

DigitalPersona, Inc.

U.are.U SDK

Version 2.0

Developer Guide



digitalPersona.

DigitalPersona, Inc.

© 2011 DigitalPersona, Inc. All Rights Reserved.

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

U.are.U® and DigitalPersona® are trademarks of DigitalPersona, Inc. registered in the United States and other countries. Windows, Windows Server 2003/2008, Windows Vista, Windows 7 and Windows XP are registered trademarks of Microsoft Corporation. All other trademarks are the property of their respective owners.

This DigitalPersona Pro Enterprise Administrator Guide and the software it describes are furnished under license as set forth in the "License Agreement" screen that is shown during the installation process.

Except as permitted by such license, no part of this document may be reproduced, stored, transmitted and translated, in any form and by any means, without the prior written consent of DigitalPersona. The contents of this manual are furnished for informational use only and are subject to change without notice.

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

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

Technical Support

The DigitalPersona web site provides online technical support at <http://www.digitalpersona.com/support>. You can also access our free support forum at any time at <http://www.digitalpersona.com/webforums>.

Feedback

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

TechPubs@digitalpersona.com

or

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

Table of Contents

1	Introduction	1
	Getting Updated Documentation	1
	Chapter Overview	1
	Standards Compliance	2
2	Background	3
	Biometrics	3
	Features of Biometric Technology	3
	The Basics of Fingerprint Identification	4
	Fingerprint Characteristics	4
	Issues in Fingerprint Recognition Technology	4
3	Developing Applications	6
	How Fingerprint Recognition Works	6
	Understanding the Data Flow	7
	Workflow - Enrollment Application	8
	Workflow - Identifying/Verifying	9
	Design Decisions	10
	Distributed Processing and Data Flow	10
	Determining an Acceptable Level of Error	11
	Defining the Data Retention Policy	12
	Specifying the Fingers to Be Scanned	12
	Optimizing Fingerprint Applications	13
4	Working with Fingerprint Readers	14
5	Converting from Gold and One Touch SDKs	15
	Converting Applications from One Touch SDK	15
	Working with Legacy Data and Deprecated Data Formats	15
	Changes in U.are.U Terminology from Legacy Usage	17
6	Using the SDK	18
	What's In the SDK?	18
	FingerJet Engine	18
	API Documentation	18
	DP Capture API	18
	FingerJet Engine API	21

7	Application Notes	24
8	Glossary	25
	General Terms	25
	Fingerprint Data	25
	Fingerprint Devices	26
	Fingerprint Recognition Terms	26
	Recognition Accuracy	27

The U.are.U SDK allows you to add fingerprint recognition to your application.

Significant new features include:

- Performing 1-to-many fingerprint identification automatically
- Support for ANSI and ISO fingerprint images and minutiae data
- Redesigned API for easier programming
- New FingerJet Engine that has met the PIV performance thresholds for fingerprint minutiae data generation required by NIST, with faster Matcher.

This manual describes the architecture and organization of the SDK. For information about specific device platforms (installation and sample applications), there are Platform Guides for Linux, Windows CE and Windows.

Getting Updated Documentation

If you are viewing this developer guide from the download package for the U.are.U SDK, you may want to check online at our website at

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

for the latest version of this document.

Chapter Overview

Chapter 1, Introduction (this chapter), provides an overview of the features and standards compliance of the U.are.U SDK.

Chapter 2, Background, contains a brief introduction to fingerprint recognition and fingerprint biometrics.

Chapter 3, Developing Applications describes the context and issues for developing applications using the U.are.U SDK. This chapter shows how data flows among the U.are.U SDK components, and includes typical workflows.

Chapter 4, Working with Fingerprint Readers contains information about working with fingerprint reader hardware.

Chapter 5, Converting from Gold and One Touch SDKs discusses how to upgrade your application to use the current API functions and data formats, as well as a mapping between the One Touch terminology and the terminology used in the U.are.U SDK.

Chapter 6, Using the SDK provides an overview of the functions in the APIs and describes how to generate API documentation using Doxygen.

Chapter 7, Application Notes provides suggestions for getting your application to work.

Chapter 8, Glossary lists the terminology specific to fingerprint recognition applications and to the U.are.U SDK.

Standards Compliance

DigitalPersona U.are.U SDK Version 2.0 Fingerprint Image Data (FID) format is compliant with

- ANSI INSITS 381-2004
- ISO/IEC 19794-4:2005

DigitalPersona U.are.U SDK Version 2.0 Fingerprint Minutiae Data (FMD) format is compliant with

- ANSI INSITS 378-2004
- ISO/IEC 19794-2:2005

DigitalPersona U.are.U SDK Version 2.0 can process FIDs and raw images generated by other systems as long as they are compliant with the aforementioned standards and:

- Are Uncompressed
- Have horizontal and vertical resolutions that are the same (square pixels)
- Have 8 bit per pixel

For multiview image records, we also require:

- No more than 16 views per finger
- All views in a multiview image must have the resolution and same pixel dimensions (Width x Height)
- We do not support unknown finger positions (finger position = 0).

DigitalPersona U.are.U SDK Version 2.0 can process minutiae data generated by other systems as long as they are compliant with the aforementioned standards and meet this criterion:

- Horizontal and Vertical resolutions are the same (square pixels)

The APIs are provided as C libraries that conforms to ANSI.C99 (<http://en.wikipedia.org/wiki/C99>).

In this chapter, we discuss the basics of fingerprint recognition. This chapter is not intended to be exhaustive, rather we're going to give you enough background knowledge to develop your own application more effectively.

For a more detailed overview, we recommend **Handbook of Fingerprint Recognition** by D. Maltoni, M. Maio, A. Jain, and S. Prabhakar, published by Springer, 2nd edition, 2009.

Biometrics

Identifying individuals based on their distinctive anatomical (fingerprint, face, iris, hand geometry) and behavioral (signature, voice) characteristics is called **biometrics**. Because biometric identifiers cannot be shared or misplaced, they intrinsically represent an individual's identity. Biometrics is quickly becoming an essential component of effective identification solutions. Recognition of a person by their body, then linking that body to an externally established "identity", forms a powerful authentication tool.

Biometric identification helps to reduce fraud, and enhance user convenience. Among the different biometric identification methods, fingerprint recognition technology has a good balance of qualities including accuracy, throughput, size and cost of readers, maturity of technology and convenience of use, making it the dominant biometric technology in commercial applications.

Features of Biometric Technology

Biometric solutions offer many advantages that other technologies cannot provide.

- Uniqueness - Fingerprints from each one of our ten fingers is distinctive, different from one another and from those of other persons. Even identical twins have different fingerprints.
- Convenience - Users no longer have to remember multiple, long and complex, frequently changing passwords or carry multiple keys.
- Non-repudication - Ensures the user is present at the point and time of recognition and later cannot deny having accessed the system.
- Non-transferable - Cannot be shared, lost, stolen, copied, distributed or forgotten unlike passwords, PINs, and smart cards.
- Proven - Long history of successful use in identification tasks - the U.S. and other countries have extensive real-world experience with fingerprint recognition. Fingerprints have been used in forensics for well over a century and there is a substantial body of scientific studies and real world data supporting the distinctiveness and permanence of fingerprints.

The Basics of Fingerprint Identification

The skin on the inside surfaces of our hands, fingers, feet, and toes is “ridged” or covered with concentric raised patterns. These ridges are called friction ridges and they provide friction making it easier for us to grasp and hold onto objects and surfaces without slippage. The many differences in the way friction ridges are patterned, broken, and forked make ridged skin areas, including fingerprints, distinctive.

The distinctiveness of fingerprints is well established. The underlying biological persistence of fingerprint characteristics is also a well established fact reported in various fingerprint studies conducted in different scientific fields over the past century.

Fingerprint Characteristics

Fingerprint ridges are not continuous, straight ridges. Instead, they are broken, forked, interrupted or changed directionally. The points at which ridges end, fork, and change are called **minutiae points** which provide distinctive, identifying information.

The most common properties of fingerprint minutiae points are:

1. Type

There are several types of minutiae points, some of which are listed below. The most common are ridge endings and ridge bifurcations.

- Ridge ending – the abrupt end of a ridge
- Ridge bifurcation – a single ridge that divides into two ridges
- Short ridge, or independent ridge – a ridge that commences, travels a short distance and then ends
- Island – a single small ridge inside a short ridge or ridge ending that is not connected to all other ridges
- Ridge enclosure – a single ridge that bifurcates and reunites shortly afterward to continue as a single ridge
- Spur – a bifurcation with a short ridge branching off a longer ridge
- Crossover or bridge – a short ridge that runs between two parallel ridges
- Delta – a Y-shaped ridge meeting
- Core – a U-turn in the ridge pattern.

2. Direction

3. Position

Issues in Fingerprint Recognition Technology

In a perfect world, it would be a simple matter to determine whether two fingerprints were from the same finger-- the images would be identical or they would not. However,

- Even though our fingerprints do not change over time, the fingerprint *images* can vary a lot, especially for some people. For example, certain skin conditions and wear due to manual labour can affect fingerprint images. This makes fingerprint recognition a very challenging problem that does not have a perfect solution. As the result, captured fingerprint images are often not a perfect match to the stored image from the same finger.
- Fingerprint images from two different fingers of two different people can look similar, especially when, because of worn fingerprints or temporary creases, there is very little information left about the actual fingerprint. The larger the population your are working with, the more likelihood of similar fingerprint images.

Fingerprint recognition software needs to address these issues. We'll discuss that more in later sections of this guide.

This chapter describes the process of developing fingerprint recognition applications, including:

- An overview of how fingerprint recognition works
- Data structures and data flows among components
- Typical workflows
- Design issues and tradeoffs.

How Fingerprint Recognition Works

Fingerprint recognition works in two stages:

1. First, users are enrolled with the system--their fingerprints are captured and stored in a database.
2. Next, when a person needs to be given access (e.g., to open a door or to log in to a computer), they simply scan their finger on the fingerprint reader.

In terms of application development, this typically requires the developer to build the following components:

1. An application for people to **enroll**:
 - Captures multiple fingerprints for at least two fingers from a fingerprint reader.
 - Checks image quality to ensure that a good quality scan is obtained.
 - Extracts the fingerprint minutiae.
 - Saves the fingerprint in a database.
2. A service(s)/application(s) that **identifies/verifies** people:
 - Captures a fingerprint from a fingerprint reader.
 - Extracts the fingerprint minutiae.
 - Compares fingerprint with enrolled fingerprints to **identify** a user from a list or **verify** a specific user.

This SDK provides fingerprint capture, extraction, enrollment and identification/verification functions to help you develop these components.

Understanding the Data Flow

When building a fingerprint recognition application, the data flow consists of:

1. Capture a **Fingerprint Image** (scan) from the fingerprint reader. The resulting **Fingerprint Image Data (FID)** contains one or more fingerprint images, called a **Fingerprint Image Views (FIVs)**. A typical FID for fingerprint recognition applications contains only one FIV but we also support multiple views (e.g., if there are multiple fingers from one individual or multiple images from a single finger stored in a single FID).

Each FIV (fingerprint) is approximately 140K in size.

2. Extract the fingerprint features. During extraction, **Fingerprint Minutiae Data (FMD)** is created, with each fingerprint stored in a **Fingerprint Minutiae View (FMV)** in the FMD. A FMV in a FMD takes no more than 1.5K (maybe less depending on the fingerprint). FMDs are used for identifying users in a collection and verifying specific users.

The ANSI and ISO standards permit multiple views but the U.are.U SDK creates only single-view FIDs and FMDs.

This data flow is shown in *Figure 1* below.

Note that in previous SDKs, DigitalPersona products used a proprietary format where enrollment fingerprints were stored as *templates* and fingerprint to be identified/verified were created as *feature sets*. ANSI and ISO standard data formats are the same for templates and feature sets, so while the current SDK supports the DigitalPersona proprietary data format for backward compatibility, new applications should be developed using ANSI and ISO standard data formats and the data flow as shown below. (The data flow for legacy data may be different, as described in *Working with Legacy Data and Deprecated Data Formats* on page 20.)

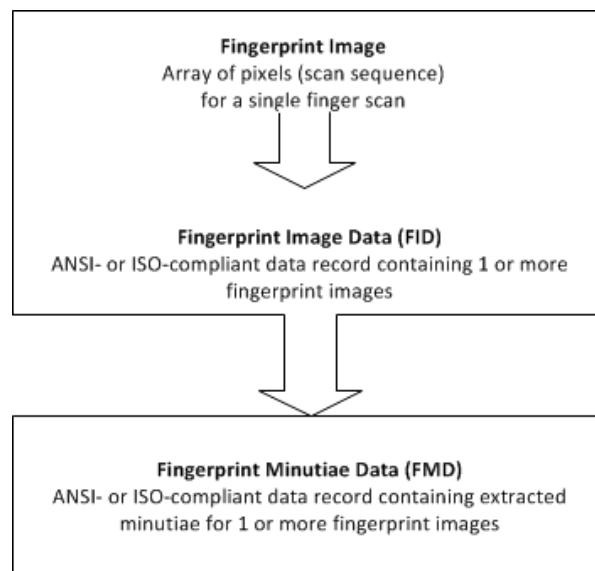


Figure 1. Data flow used by U.are.U SDK in fingerprint recognition

Workflow - Enrollment Application

During the enrollment process, one or more fingers are scanned for each person. We recommend that you enroll at least two fingers (more is recommended) because in the event of an accident or injury to one finger, another enrolled finger can be used to identify the individual.

The enrollment application needs to perform the following steps to enroll a single finger from a user:

Step One - Initialization

Initialize the library. Discover the available devices and open a connection to the device.

Step Two - Capture, Extract and Enroll

1. Begin the enrollment process.
2. Capture a series of fingerprint scan(s); for each scan,
 - Create a FID,
 - Extract fingerprint minutiae and create the FMD,
 - Add the FMD to the pool of FMDs for enrollment.
3. Continue to capture fingerprints until the enrollment process has enough FMDs to complete the enrollment. (The enrollment functions evaluate the FMDs and select the best image -- typically several scans are required.)
4. Create the enrollment FMD and release resources.

Step Three - Store Data

Store the enrollment FMD. Many applications keep only the enrollment FMD because of space constraints or policy decisions. You cannot use FIDs for identification, so even if you choose to keep the FIDs, you must also store the FMD for each individual.

Notes on Enrollment

Before storing, you may want to check for existing entries that match the new entry -- applications like law enforcement, banking or voting registration, may not allow duplicate enrollments.

The capture/extract minutiae part of the enrollment process is the same as for capturing/extracting minutiae for the purpose of verifying/identifying users. If you wish, you can enroll users without using the enrollment functions (by simply capturing, extracting minutiae and storing the resulting FMD). However we recommend that you use the enrollment functions to create the best quality enrollment FMDs.

The enrollment process is described in more detail on *page 22*.

Workflow - Identifying/Verifying

Fingerprint recognition involves two types of operation:

- **Identification** - Comparing a fingerprint against the database of enrolled fingerprints and confirming that the fingerprint is enrolled (e.g., to open a door there may be many authorized users).
- **Verification** - Comparing a fingerprint against a specific user's enrolled fingerprint(s) to verify a specific person's identity (e.g., when the user types their name and then uses a fingerprint rather than a password).

To perform these operations, your application needs to do the following steps:

Step One - Initialization

Initialize the library. Discover the available devices and open a connection to the fingerprint reader device.

Step Two - Capture and Extract

1. Wait for a fingerprint. When a fingerprint is detected, capture the image and create an FID.
2. Extract fingerprint minutiae and create an FMD.

This sequence is exactly the same as for the capture/extraction process during enrollment.

Step Three - Identify/Verify

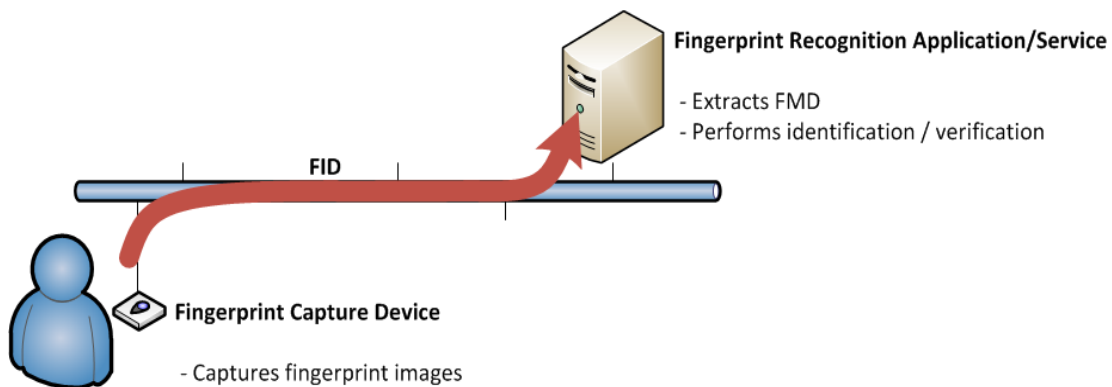
Call the appropriate function to **verify** a specific person OR to **identify** a valid user.

Design Decisions

Distributed Processing and Data Flow

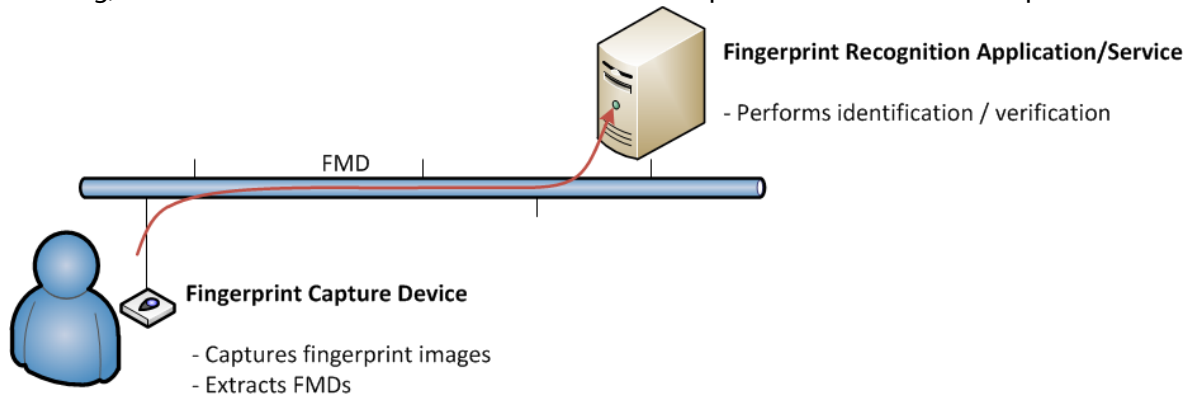
Depending on the capabilities of your fingerprint reader, you can capture FIDs and send them to another machine for processing OR the fingerprint reader can extract the FMD and transmit only the much smaller FMD files. Thus the application can be designed in these two ways:

1. The fingerprint capture device can simply capture a fingerprint image and transmit the image to a server for processing as shown in the image below. Since FIDs are large (around 100K - 140K), this means that you need a faster connection, but there is less computing power required by the fingerprint reader.



Optionally, the fingerprint capture device can compress the image before transmitting it. **If you use compression to save bandwidth, do NOT use lossy data formats like JPEG.** If you need to compress fingerprint images, we recommend JPEG2000 as a compression format. For 500dpi images, you can use JPEG2000 with a compression ratio of 12 to reduce images to approximately 12K. **The U.are.U SDK does not support compressed images directly**, so you would have to capture the fingerprint as a raw image and compress it to JPEG2000. After transmission, you would convert the JPEG2000 file back into a raw image for minutiae extraction and identification/verification.

2. Another alternative is to develop software for the fingerprint capture device to capture the fingerprint image AND extract the fingerprint features to create a FMD. The FMD is then transmitted to the server for processing, as shown below. FMDs are 1.5K or less and so require less bandwidth and speed.



Determining an Acceptable Level of Error

When identifying fingerprints, you want to identify strictly enough that you do not let unauthorized people have access (false positives) but also do not inconvenience legitimate users by rejecting their fingerprints (false negatives). Note that some people will always experience more false rejections -- the rate of false negatives is a statistical measure, but *individuals* may experience higher rejection rates, based on their specific fingerprint characteristics.

There is a trade-off between the frequencies of false positive and false negative errors. Applications have control over this trade-off by specifying the threshold for the required degree of similarity between two fingerprint images in order to call it a match. **When choosing the identification threshold, note that increasing the false positive error rate by a factor of 100 will reduce the false negative error rate only approximately by a factor of 2.**

There will always be some false negatives. As the result, every practical fingerprint recognition system should have an alternative means to establish and prove identity, without using fingerprints.

Setting the Error Threshold when Identifying a Fingerprint in a Collection

When a fingerprint is scanned, the first step is to identify the fingerprint against a set of stored FMDs.

- If you are trying to confirm that a user is allowed access, you will want to identify the fingerprint against all the valid FMDs that you have stored.
- If you are trying to confirm the identity of a specific person, you must identify against all FMDs for that individual (typically at least two fingers are stored for each user).

The `dpfj_identify` function compares a FMV against a collection of FMDs to produce the candidate list. You can specify the maximum desired number of candidates: a smaller number can make the execution faster. The most similar candidates are returned closer to the beginning of the candidate list.

Your **threshold** determines the trade-off between false positive and false negative error rates where:

- 0 = no false positives
- `maxint` (#7FFFFFFF or 2147483647) = fingerprints do not match at all
- Values close to 0 allow very few false positives; values closer to `maxint` allow very poor matches (a lot of false positives) in the candidate list. The table below shows the relationship between the threshold values and the false positive identification error rates observed in our test. Note: the actual false positive identification error rates in your deployment may vary.

Your Threshold	Corresponding False Positive Identification Rate	Expected number of False Positive Identifications	Numeric Value of Threshold
.001 * <code>maxint</code>	.1%	1 in 1,000	2147483
.0001 * <code>maxint</code>	.01%	1 in 10,000	214748

Your Threshold	Corresponding False Positive Identification Rate	Expected number of False Positive Identifications	Numeric Value of Threshold
.00001 * maxint	.001%	1 in 100,000	21474
1.0e-6 * maxint	.0001%	1 in 1,000,000	2147

For many applications, a good starting point for testing is a threshold of 1 in 100,000. If you want to be conservative, then you will want to set the threshold lower than the desired error rate (e.g., if you want an error rate that does not exceed 1 in 100,000, you might set the threshold to 1 in 1,000,000).

Defining the Data Retention Policy

After a fingerprint scan, a FID is created, which contains the actual image. Each fingerprint image takes roughly 140K of storage. With modern computers, that is not a huge amount, but each enrolled user may have several fingerprints scanned. Multiplied by the number of potential users, this can add up to a fair amount of data.

To identify fingerprints, you must extract the fingerprint characteristics to create a FMD, which is < 1.5K per fingerprint.

Some applications choose to retain the full image records, but other companies discard the image record and retain only the FMD. Note that if you discard the image record, you cannot reconstruct the original fingerprint image from the FMD, the FMD is only useful for identifying/verifying fingerprints. From time to time DigitalPersona may release a new version of the U.are.U SDK that will provide improved accuracy in the feature extraction process. If you do not retain the fingerprint images, you will not be able to redo the feature extraction using the new version of the SDK/runtime in order to take advantage of these improvements.

Specifying the Fingers to Be Scanned

When you enroll people into your system, you will usually want more than one finger enrolled. This allows for injury and makes it easier for people who have fingerprints that are difficult to recognize. For some applications you may want to scan all ten fingers or take multiple fingerprint impressions for individual fingers.

A typical policy would be to require both index fingers or both thumbs to be scanned. Thumbs are typically the worst in terms of recognition accuracy and thumbs require different capture devices with larger area and different ergonomics. Where possible, avoid any industrial design forcing users to use thumbs.

The preferred approach is to enroll, at a minimum, both index and both middle fingers. Middle fingers are usually the best, probably because people have almost as good dexterity with middle fingers as with index fingers, and yet middle fingers have fingerprints that are typically less worn than index fingers.

Some solutions are ergonomically designed in a way that only the right or only the left hand can be used conveniently. In this case the right hand is better (probably because the majority of people are right handed), and at least three fingers need to be enrolled: index, middle and ring fingers.

Ergonomics and the correct finger placement are extremely important. Poor ergonomics can easily increase the false negative identification rate by a factor of 5 to 10. The system users need to be aware of correct finger placement for best results.

Optimizing Fingerprint Applications

To identify a fingerprint against a large database can take a considerable amount of time and create unacceptable delays between the fingerprint scan and the user authorization. Identifying fingerprints will be faster if the database of fingerprints is in memory rather than retrieved from disk. If you have a large database of users, you may need to provision your server with an appropriate amount of RAM to handle the searches.

This SDK is not optimized for large scale identification. If you are developing such an application, you may want to contact DigitalPersona to get help selecting the technology that will best fit your needs.

To ensure the fastest response time, you must weigh whether it will be faster to transmit the image record to a server for FMD extraction or whether it is faster to extract the FMD on the device and transmit only the FMD to the server for identification/verification.

For devices that are used by a limited number of people (e.g., kiosks or pharmacy cabinets), you may have the device identify fingerprints against a limited set of FMDs. However this requires that you keep the FMDs in the device in sync with your central database, to ensure that new employees are able to gain access and departing employees' privileges are revoked quickly.

The U.are.U SDK works with the following fingerprint readers:

- U.are.U 4000B Fingerprint Reader Rev. 100 and Rev. 101 48Mhz
- U.are.U 4500 Fingerprint Reader Rev. 103

Neither reader provides support for streaming.

Fingerprint readers can lose their calibration as a result of changes in temperature, humidity and ambient light. Humidity is the most important environmental factor affecting calibration. The U.are.U 4000B and 4500 Fingerprint Readers are self-calibrating, but you still might want to set up your application to check periodically that no additional calibration is needed.

If the fingerprint reader is not giving clear readings:

- Try cleaning it with sticky tape. Gently dab it with the sticky side of the tape. Do not rub it with paper and do not get it wet.
- Make sure that you are touching the fingerprint reader with the pad of your finger, not the tip. The most detail (minutiae) occur roughly midway between the first joint and the tip.
- If your fingers are very dry, try touching your forehead with the pad of the finger you are trying to scan and then rescanning your fingerprint.

Good ergonomics in the mounting of your fingerprint reader can significantly improve the quality of fingerprint scans. Be sure that the reader is mounted in a way that is convenient for users to touch properly, such that the large area of the pad of the finger is captured. The angle at which the reader is mounted as well as the rim around the sensing area can make it difficult for people with large fingers or long fingernails to have proper contact between the pad of the finger and the sensing area.

If your fingerprint reader becomes non-responsive from an electrostatic shock you may need to perform a hardware reset.

Your application should check the device status between fingerprint scans to ensure that the hardware has not experienced an error condition.

The U.are.U SDK can exchange data with applications developed using the following DigitalPersona products:

- DigitalPersona Gold SDK/Fingerprint Recognition Software
- DigitalPersona One Touch SDKs

This SDK does not support the custom encryption keys that are supported by other DigitalPersona products.

Converting Applications from One Touch SDK

To convert an application from One Touch SDK, install the U.are.U SDK as described above and use this documentation to modify your applications to use the new API.

Be sure to install U.are.U SDK in a new folder on your development machine so that your existing files do not get overwritten.

Note that when you install the U.are.U SDK on the target device, **the One Touch drivers will be overwritten with new drivers that are compatible with both your existing applications and new applications based on U.are.U SDK.**

The files installed on the target device include both drivers and SDK files. If you retain the One Touch SDK files on the device, you will need up to an additional 120K for the new U.are.U SDK files. If you retain the old SDK files on the device, you can run applications based on either the old or the new SDK. The two SDKs can coexist and your existing applications can run using the older run-time environment while you update your programs or develop new applications based on the new SDK. Applications based on the old and new SDKs can run on the same hardware, but not simultaneously.

Note that the **data** from existing applications based on One Touch SDK is fully compatible with applications based on U.are.U SDK. For more information, see *Working with Legacy Data and Deprecated Data Formats* on page 15.

Working with Legacy Data and Deprecated Data Formats

The DigitalPersona Gold SDK and One Touch SDK products used a different format for minutiae data (which were called feature sets). This data format has been deprecated. In this older format, there were three different data types for minutiae data which reflected the intended use of the data:

- **Pre-registration features** (Gold SDK), or **Pre-registration Feature Set** to be used for Enrollment (OneTouch SDK); Pre-registration features were intended only for creation of Registration features. An application needed to collect four fingerprints of the finger to enroll, extract pre-registration features, and pass it to the SDK to produce registration features.

- **Registration features** (Gold SDK), or **Fingerprint Template** (OneTouch SDK); Registration features were intended to be stored in a database as an enrolled finger.
- **Verification features** (Gold SDK), or **Feature Set** (One Touch SDK). Verification features are what is compared to the Registration features when a user swipes a finger.

In the previous SDKs, you could not compare two Feature Sets, you could only compare a Feature Set against an Fingerprint Template. The U.are.U SDK removes that distinction and when you create data with ANSI/ISO data formats, all minutiae data is stored in an FMD, whether produced by the feature extraction functions or the enrollment functions.

You may also choose to use the legacy DigitalPersona format for your data. For new applications, we recommend a standardized (ANSI or ISO) format.

When passing an array of fingerprint templates into the identification function, all the templates in the array should be in the same format. If you have existing data in the legacy format, you must either convert the data and work in a new format or continue to use the legacy format for all of your data. The conversion functions can convert legacy data.

The `dpfj_identify` and `dpfj_compare` functions can work with the legacy data formats listed above.

The U.are.U SDK supports the legacy data formats to allow for continued support of old applications, but we encourage you to convert your data to a newer format for new applications.

Changes in U.are.U Terminology from Legacy Usage

This section describes changes in terminology from the One Touch documentation and other DigitalPersona products.

Old Term	New Term	Explanation
Fingerprint authentication	Fingerprint verification	Change in industry standard terminology.
Fingerprint registration	Fingerprint enrollment	Change in industry standard terminology.
Fingerprint sensor (referring to a fingerprint capture device)	Fingerprint reader	Erroneous usage: Sensors are a <i>component</i> of certain types of <i>fingerprint capture devices</i> , they are not themselves <i>fingerprint capture devices</i> .
Match	Compare	Change in industry standard terminology.
Matching	Comparison	
Matching score	Comparison score	
Performance	Recognition accuracy	More accurate terminology.
Fingerprint feature set Fingerprint template	Fingerprint minutiae data	<i>Fingerprint templates</i> (<i>fingerprint features</i> stored for enrolled fingers) and <i>fingerprint feature sets</i> (images used for verification and identification) have been replaced with <i>fingerprint minutiae data</i> in the U.are.U family of the SDKs only. With standards-based data formats in the U.are.U SDKs, all <i>fingerprints</i> are stored in the same format.
ROC curve	DET curve	More accurate terminology. We have never used ROC curves, however our DET curves were sometimes erroneously called ROC curves in the past.
FAR	FMR or FNMR as appropriate	The definition of <i>FAR</i> is application-specific. In some applications <i>FAR</i> is similar to <i>FMR</i> while in other applications, <i>FAR</i> is similar to <i>FNMR</i> .
FRR	FMR or FNMR as appropriate	The definition of <i>FRR</i> is application-specific. In some applications <i>FRR</i> is similar to <i>FNMR</i> , while in the others <i>FRR</i> is similar to <i>FMR</i> .

What's In the SDK?

The SDK consists of:

1. APIs -- C libraries that conform to ANSI.C99 (<http://en.wikipedia.org/wiki/C99>):
 - **DP Capture API** - for capturing fingerprints
 - **FingerJet Engine API** - for extracting fingerprint characteristics and identifying/verifying fingerprints
2. Run-time components:
 - Capture driver and SDK layer
 - FingerJet Engine run-time

FingerJet Engine

FingerJet Engine is a module that extracts fingerprint characteristics from image records to create FMDs and compares FMDs to confirm identity.

FingerJet Engine has met the PIV performance thresholds for fingerprint minutiae data generation required by NIST.

We support a maximum of 16 views in a single FID or FMD. All views in a single record must have the same resolution. We do not support unknown finger positions (finger position = 0).

API Documentation

Detailed documentation for the API is contained in the header files. You can simply read the header files or you can use Doxygen (<http://doxygen.org>). To generate the documentation, use Doxywizard, set to be optimized for C/PHP and output only plain HTML.

DP Capture API

The DP Capture API consists of library management, device management, capture & stream.

Library Management

Table 1. Library management functions

Function	Description
dpfpdd_init	Initialize the library (allocate system resources and initialize data). This must be the first function called.
dpfpdd_exit	Release the library and its resources.
dpfpdd_version	Query the library version. This is the only function that can be called before dpfpdd_init or after dpfpdd_exit. This returns the DP Capture API library version (not the U.are.U SDK version). This is analogous to the dpfj_version function which returns the version of the FingerJet library file.

Device Management

Table 2. Device hardware management functions

Function	Description
dpfpdd_query_devices	Discover connected devices.
dpfpdd_open	Open a device. This function establishes an exclusive link to the device; no other processes will be able to use the device until you close it. The application <i>must</i> open the device before use.
dpfpdd_close	Close a device.
dpfpdd_get_device_status	Get the status for a device. You would normally check the device status between captures to ensure that the device is functioning and there are no error conditions.
dpfpdd_get_device_capabilities	Query a device for information on capabilities.
dpfpdd_set_parameter	Change a device or driver parameter
dpfpdd_get_parameter	Query a device or driver parameter
dpfpdd_calibrate	Calibrate a device. Some devices are self-calibrating. Ambient light or temperature can affect calibration, for some devices. Calibration can take several seconds.
dpfpdd_reset	Do a hardware reset on the device. Hardware resets are typically needed only after a hardware problem (e.g., the device is unplugged or receives an electrostatic shock). Hardware resets typically only take a few milliseconds.

Capturing Fingerprints

The `dpfpdd_capture` function captures a fingerprint image for

- Enrollment (as part of the process described on page 22)
- Identifying users with `dpfj_identify`
- Verifying a specific user identity with `dpfj_compare`

Table 3. Fingerprint capture functions

Function	Description
<code>dpfpdd_capture</code>	Capture a fingerprint image from the device. This function signals the device that a fingerprint is expected, and waits til a fingerprint is received.
<code>dpfpdd_cancel</code>	Cancel a pending capture

Streaming Fingerprints

Streaming features are not available on the U.are.U 4000B and 4500 Fingerprint Readers.

Table 4. Streaming Functions

Function	Description
<code>dpfpdd_start_stream</code>	Start streaming mode
<code>dpfpdd_get_stream_image</code>	Capture a fingerprint image from the streaming data
<code>dpfpdd_stop_stream</code>	End streaming mode

FingerJet Engine API

The FingerJet API contains functions that extract features from FIDs to create FMDs, identify/verify FMDs and convert FMDs to different formats.

Library Management

Table 5. Library version verification function

Function	Description
dpfj_version	Query the library version. This returns the FingerJet library version (not the U.are.U SDK version). This is analogous to the dpfpdd_version function which returns the version of the DP Capture API library file.

Extract FMD

Table 6. Feature extraction functions

Function	Description
dpfj_create_fmd_from_raw	Creates FMD from raw image
dpfj_create_fmd_from_fid	Creates FMD from ANSI, ISO or DigitalPersona legacy format FID

Identify Fingerprint

This function is described in detail on page 12.

Table 7. Fingerprint identify function

Function	Description
dpfj_identify	Identify a FMD: given an array of FMDs, this function returns an array of candidates that match the original fingerprint (a FMV within a FMD) within the threshold of error. Supported formats are: Gold SDK, One Touch SDK, ANSI and ISO.

Enrollment

The enrollment functions allow you to enroll a finger to create a FMD that you can store in your database. For ANSI/ISO formats, the enrollment functions create FMDs. For legacy DigitalPersona format, the enrollment functions create a fingerprint template.

The typical process would be:

1. Call `dpfj_start_enrollment`.
2. Capture a fingerprint scan and extract an FMD, using the standard functions (`dpfpdd_capture` to capture and `dpfj_create_fmd_from_fid` or `dpfj_create_fmd_from_raw` to extract).
3. Call `dpfj_add_to_enrollment` to add the fingerprint to the potential pool.
4. Repeat the previous two steps until `dpfj_add_to_enrollment` returns a flag indicating the pool of FMDs is now sufficient to create an enrollment FMD.
5. Create the enrollment FMD with `dpfj_create_enrollment_fmd` and release resources by calling `dpfj_finish_enrollment`.
6. Store the enrollment FMD in your database. Some applications like voting, banking and law enforcement require that you check for duplicate fingerprints before storing a new fingerprint in the database.

Table 8. Enrollment Functions

Function	Description
<code>dpfj_start_enrollment</code>	Begin the enrollment process and allocate resources.
<code>dpfj_add_to_enrollment</code>	Add the FMD to the pool of FMDs for enrollment and return a flag indicating that the enrollment is ready (enough FMDs have been received to create the enrollment FMD)
<code>dpfj_create_enrollment_fmd</code>	Create FMD for enrolled finger
<code>dpfj_finish_enrollment</code>	Release resources used during enrollment process

Format Conversion

Table 9. Conversion functions to convert FMDs from legacy formats or between supported formats

Function	Description
<code>dpfj_fmd_convert</code>	Convert FMDs from any supported format to any other; supported formats are: Gold SDK, One Touch SDK, ANSI and ISO.
<code>dpfj_dp_fid_convert</code>	Convert legacy DigitalPersona image (Gold SDK and One Touch SDK) to ANSI or ISO images

Advanced Diagnostics

The majority of applications should use the `dpfj_identify` function to implement both identification and verification. However, in a few special cases, e.g., using multi-modal biometrics, or doing statistical risk assessment, the `dpfj_compare` function allows you to compare two FMVs to determine their actual degree of dissimilarity. This is useful for accuracy testing and diagnostics and is not intended to be used in final applications for actual fingerprint recognition. The `dpfj_compare` function returns a **dissimilarity score** with values:

- 0 = fingerprints are NOT dissimilar (i.e., they MATCH perfectly).
- `maxint` (#7FFFFFFF or 2147483647) = fingerprints are completely dissimilar (i.e., DO NOT match).
- Values close to 0 indicate very close matches, values closer to `maxint` indicate very poor matches.

The table below shows the relationship between the scores returned from `dpfj_compare` and the false match error rates observed in our test. The dissimilarity score distribution is estimated based on our internal testing, and may not be representative of the actual rate that will be observed in deployment.

Dissimilarity Score	False Match Rate
2147483	.1%
214748	.01%
21474	.001%
2147	.0001%

Table 10.

Function	Description
<code>dpfj_compare</code>	Compare two FMDs; supported formats are: Gold SDK, One Touch SDK, ANSI and ISO.

This chapter describes some suggestions for troubleshooting your application.

It is very important that you test your application with multiple and diverse people. The readability of fingerprints is affected by many factors including wear and tear, skin dryness, and age.

The middle finger often gives better scans than the index finger because the middle finger is typically less worn. We recommend that you enroll more than one finger for each user, preferably at least the index and middle fingers.

The `dpfj_identify` function returns `DPFJ_SUCCESS` if it is able to do identification (i.e., the FMDs are valid and correctly formed). However that does not mean that a candidate was found. You must check the number of returned candidates to see if any actual matches were found.

The `dpfj_compare` function returns `DPFJ_SUCCESS` if it is able to do the requested comparison (i.e., the FMDs are valid and correctly formed). However that does not mean that the fingerprints matched. To check whether they matched, you must look at the dissimilarity score (0=no match, `maxint`=perfect match, `maxint-threshold` = acceptable match).

General Terms

Fingerprint

The impression left from the friction ridges of a human finger.

Fingerprint characteristics

The biological finger surface details that can be detected and from which distinguishing and repeatable *fingerprint features* can be extracted.

Fingerprint minutiae

The *fingerprint characteristics* commonly used in fingerprint recognition systems. *Fingerprint minutiae* include:

- Ridge ending – the abrupt end of a ridge
- Ridge bifurcation – a single ridge that divides into two ridges
- Short ridge, or independent ridge – a ridge that commences, travels a short distance and then ends
- Island – a single small ridge inside a short ridge or ridge ending that is not connected to all other ridges
- Ridge enclosure – a single ridge that bifurcates and reunites shortly afterward to continue as a single ridge
- Spur – a bifurcation with a short ridge branching off a longer ridge
- Crossover or bridge – a short ridge that runs between two parallel ridges
- Delta – a Y-shaped ridge meeting
- Core – a U-turn in the ridge pattern.

Fingerprint Data

Fingerprint sample

An analog or digital representation of a *fingerprint* obtained from a *fingerprint capture device*. See also *fingerprint image*.

Fingerprint image

A digital representation of a *fingerprint sample* encoded as a spatially mapped array of pixels. A *fingerprint image* is the only representation of a *fingerprint sample* supported by DigitalPersona products, thus the terms *fingerprint image* and *fingerprint sample* are used interchangeably.

Fingerprint features

The digital representation of *fingerprint characteristics*.

Fingerprint Image Data (FID)

The binary data containing one or more *fingerprint images* from one or multiple fingers of the same person.

Fingerprint Image View (FIV)

The part of an *FID* that contains a *fingerprint image* from a single impression of a single finger.

In the majority of cases, an *FID* contains only one *FIV*.

Fingerprint Minutiae Data (FMD)

The digital representation of *fingerprint minutiae*, and optionally other *fingerprint characteristics*, from one or multiple fingers of the same person.

Fingerprint Minutiae View (FMV)

The part of an *FMD* that contains *fingerprint features* from a single impression of a single finger.

Typically *FMDs* contain only one *FMV*.

Fingerprint template

The *fingerprint minutiae data* that is stored as a result of the *enrollment* process.

Fingerprint Devices

Fingerprint capture device

A device that collects a signal of *fingerprint characteristic* and outputs a *fingerprint sample*. This device can consist of one or more components, including hardware and supporting software. For example, a *fingerprint capture device* may include a camera, photographic paper, a printer, and/or a digital scanner.

Fingerprint reader

A *fingerprint capture device* that obtains a *fingerprint image* by direct interaction with a finger.

Fingerprint Recognition Terms

Capture

The process of acquiring a *fingerprint image* from a *fingerprint reader* or *fingerprint capture device*.

Enrollment

The process of *capturing a fingerprint image(s)* for an individual, extracting *fingerprint features*, optionally checking for duplicates, and storing the *fingerprint features (fingerprint template)*. See also *FMD*, *Fingerprint Template*.

Comparison

The function which, given two *fingerprints* computes a *comparison score*.

Comparison score

A *comparison score* is a numerical value resulting from the comparison of the *fingerprint features* of two *fingerprints*. *Comparison scores* are of two types: *similarity scores* and *dissimilarity scores*.

Comparison scores are calculated using algorithms that are specific to the fingerprint recognition system and, optionally, can be normalized in order to maintain a

constant score distribution of *impostor verification attempts* or *open set identification attempts*.

Similarity Score

See *comparison score*.

Dissimilarity Score

See *comparison score*.

Fingerprint Recognition

Verification or *identification*.

Verification

The process of

1. *Capturing a fingerprint image* from an individual,
2. *Extracting fingerprint features*,
3. *Comparing* the captured image's *fingerprint features* with the *fingerprint template(s)* of the enrollee this individual claims to be and
4. *Making the match/non-match decision*.

Verification Threshold

The maximum degree of dissimilarity that is allowed between a *fingerprint image* and a *fingerprint template* in order to make the *match decision* during the *verification* process.

Match decision or match

A decision that two *fingerprints* are from the same finger (meet the *verification threshold*).

Non-match decision or non-match

A decision that two *fingerprints* are not from the same finger (do not meet the *verification threshold*).

Identification

The process of

1. *Capturing a fingerprint image* from an individual,
2. *Extracting fingerprint features* and

3. *Comparing* the captured image's *fingerprint features* with the *fingerprint templates* from multiple individuals in order to create a candidate list of fingerprints that meet the *identification threshold*.

Identification Threshold

The maximum degree of dissimilarity that is allowed between a *fingerprint image* and a *fingerprint template* in order to call the *fingerprint template* a *candidate* in the *identification process*.

Candidate

A *fingerprint template* that is determined to be similar to a given *FMD* during identification.

Recognition Accuracy

The terminology below is used when discussing testing processes and reporting accuracy for fingerprint-enabled applications.

Genuine verification attempt

An attempt to compare *fingerprint features* with a *fingerprint template* where the *fingerprint features* and *fingerprint template* are from the same individual, and the *fingerprint features* and *fingerprint template* were captured at different times (recommended at least 2-3 weeks apart).

Impostor verification attempt

An attempt to compare *fingerprint features* with a *fingerprint template*, where the *fingerprint features* and *fingerprint template* are from two different people.

False match rate (FMR)

The ratio between the number of *impostor verification attempts* that produced a *match decision* and the total number of *impostor verification attempts*. See also *FAR*.

False non-match rate (FNMR)

The ratio between the number of *genuine verification attempts* that produced a *non-match decision* and the total number of *genuine verification attempts*. See also *FRR*.

False Accept Rate (FAR)

The application-specific measure of how often the application falsely grants a request.

In authentication applications, *FAR* is used in place of *FMR*.

False Reject Rate (FRR)

The application-specific measure of how often the application falsely rejects a request.

In authentication applications, *FRR* is used in place of *FNMR*.

Verification Detection Error Trade-off (DET) curve

A parametric curve that plots *FMR* on the x-axis and *FNMR* on the y-axis as a function of the *verification threshold*.

Probe

In simulation tests, a *fingerprint* used in searches (analogous to the user *fingerprint* used by a real application for searching).

Gallery

In simulation tests, a set of enrolled *fingerprints* (*fingerprint templates*) used for comparison purposes (analogous to the database of enrolled *fingerprint templates* used by a real application).

Open set identification attempt

For the purpose of testing, an attempt to compare a *probe* against a *gallery* that does not contain any impressions from any finger of the individual whose finger was used as the *probe*.

Closed set identification attempt

For the purpose of testing, an attempt to compare a *probe* against a *gallery*, where the *gallery* includes a *fingerprint template* from the same finger (from the same person) as used for the probe. The *probe* and the corresponding *fingerprint template* should be captured at different times (recommended at least 2-3 weeks apart).

False positive identification rate (FPIR)

Proportion of *identification* requests that return a *candidate* even though the person is not enrolled.

More precisely, for the purpose of recognition accuracy testing, the ratio between the number of *open set identification attempts* that returned one or more *candidates* and the total number of *open set identification attempts*.

False negative identification rate (FNIR)

Proportion of *identification* requests by enrollees that do not return the correct *candidate*.

More precisely, for the purpose of recognition accuracy testing, the ratio between the number of *closed set identification attempts* that did not return the correct *candidate* and the total number of *closed set identification attempts*.

Identification detection error trade-off (DET) curve

A parametric curve that plots *FPIR* on the x-axis and *FNIR* on the y-axis as a function of the *identification threshold*. The *Identification DET curve* depends on the *gallery* size. Typically, multiple *identification DET curves* are shown on the same plot, one for each *gallery* size.