The JAVA API

THIS CHAPTER DESCRIBES HOW TO USE THE JAVA API.

The Java API is built as a wrapper to the C/C++ API. The Java API is available for Linux, Android and Windows. This chapter provides an overview of the API. For details of using the API on a specific reader platform, consult the appropriate Platform Guide.

The Java API is considerably simpler to use than the C/C++ APIs and therefore generally results in:

- Easier data management
- Easier enrollment
- Faster development

Importing the U.are.U Java package

The U.are.U Java library classes and interfaces are aggregated into `dpuaru.jar`. To use the U.are.U Java library functionality import the `com.Crossmatch.uareu.*` package, and make sure to include `dpuareu.jar` into your classpath.

Getting Detailed Documentation

This chapter provides an overview of the main methods in the Java API. For a complete description of method parameters, the Javadoc documentation for the Java libraries is provided. Consult the platform guide for details of where the Javadoc files are located.

Using the Package

Main Access Point

The main access point to the U.are.U Java library is the `UareUGlobal` class. This is a static class, which allows you to acquire references to the classes related to fingerprint readers and to the FingerJet Engine:

- **To acquire a reference to ReaderCollection** use the `GetReaderCollection()` method. To destroy `ReaderCollection`, (release all system resources associated with readers and make readers available for other processes) use the `DestroyReaderCollection()` method.
- **To acquire a reference(s) to individual readers**, use the `ReaderCollection` object, which is a collection of objects of type `Reader`.
- **To acquire a reference to the FingerJet Engine** use the `GetEngine()` method. The engine does not use or allocate any system resources except memory and does not have to be destroyed explicitly.

UareUException

The `UareUException` interface describes exceptions specific to the U.are.U SDK.

Getting a List of Available Readers

The `ReaderCollection` interface provides a list of the readers connected to the machine. A list of available readers can be acquired any time with `GetReaders()` method.

Working with Readers

Each attached reader is represented with a `Reader` object. The `Reader` interface allows:

- Querying reader description and capabilities,
- Acquiring status of the reader,
- Capturing fingerprints,
- Starting and stopping video stream, and
- Resetting and calibrating the reader.

The main methods are:

Function	Description
GetDescription	Get the description of a reader. The description is available at any time (even if the device is not open). This is the only method that can be called before the `open()` method. Returns an object of type `Description` holding information about the reader hardware.
Open	Open a device and return the device capabilities. This method establishes an exclusive link to the device; no other processes will be able to use the device until you close it. The application *must* open the device before use.
GetStatus	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. Returns an object of type `ReaderStatus` which describes the current status of the reader.
GetCapabilities	Get the capabilities of a device. Returns an object of type `Capabilities` which describes what the reader can do.
Calibrate	Calibrate a device. Some devices are self-calibrating. Ambient light or temperature can affect calibration, for some devices. Calibration can take several seconds.
Reset	Do a hardware reset on the reader. 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.
Close	Close a reader and release the resources associated with the reader.

Capturing Fingerprints

The `Capture` function captures a fingerprint image for

- Enrollment (as part of the process described on page *12)*
- Identifying users with `Identify`
- Verifying a specific user identity with `Compare`

The primary fingerprint capture methods are:

Function	Description
Capture	Captures a fingerprint image from the open reader. This function signals the reader that a fingerprint is expected, and blocks until a fingerprint is received, capture fails or the reader times out.

CancelCapture	Cancel a pending capture

Streaming Fingerprints

Not all readers support streaming mode. To determine if a specific reader supports this feature, get the reader capabilities with the `GetCapabilities` method and check the value of the field `can_stream`.

The streaming methods are:

Function	Description
StartStreaming	Put the reader into streaming mode. In this mode, the application must call `GetStreamImage()` to acquire images from the stream.
GetStreamImage	Capture a fingerprint image from the streaming data. After this function returns, the reader remains in streaming mode. Frame selection, scoring and other image processing are not performed by this function.
StopStreaming	End streaming mode

Accessing the FingerJet Engine

The `Engine` interface provides functionality to

- Extract fingerprint features (create FMDs),
- Identify and compare FMDs, and
- Create enrollment FMDs.

See Understanding the Data Flow on page 11 for details on enrollment and comparison terminology and data flow.

Creating FMDs from images

The `CreateFmd()` method can be used in two ways:

1 Extract fingerprint minutiae from a raw image and create an FMD.

2 Extract fingerprint minutiae from an FID and create an FMD.

The following limitations are applied to the raw images and FIDs:

- 8 bits per pixel
- no padding
- square pixels (horizontal and vertical dpi are the same)

The size of the resulting FMD will vary depending on the minutiae in a specific fingerprint.

Identification and Comparison

`Identify()` identifies a single FMD against an array of FMDs. This function takes as inputs:

- A single view in an FMD
- An array of FMDs (each FMD can contain up to 16 views) to compare
- The desired number of candidates to return
- The threshold for False Positive Identification Rate (FPIR) that is permitted

Each time a view has a score lower than the threshold FPIR, that view is marked as a possible candidate. Then when all possible candidates are identified (i.e., they meet the threshold), they are ranked by their score. Finally, the function returns as many candidates as requested, based on the candidates with the lowest dissimilarity score. For a discussion

of setting the threshold as well as the statistical validity of the dissimilarity score and error rates, consult NIST Fingerprint Image Quality (NFIQ) on page 14.

`Compare()` takes two single views from two FMDs and returns a dissimilarity score indicating the quality of the match.

The majority of applications should use the `Identify` method to implement both identification and verification. However, in a few special cases, e.g., using multi-modal biometrics, or doing statistical risk assessment, the `Compare` method 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 `Compare` method 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 `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%

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

Enrollment

`CreateEnrollmentFmd()` creates and returns an enrollment FMD. It takes as input a reference to an object of type `EnrollmentCallback`. The client must implement `EnrollmentCallback.GetFmd()`. This method acquires and returns an FMD to add to the enrollment. The engine calls `EnrollmentCallback.GetFmd()` as many times as needed in order to create an enrollment FMD.

Normally, the client application will implement `EnrollmentCallback.GetFmd()` to provide the onscreen UI to capture fingerprints from the reader, extract features using `CreateFmd()`, and return an enrollment FMD. If the user wants to cancel the enrollment, `EnrollmentCallback.GetFmd()` should return `null`.