

IT WORKS
IDWORKS Integrator SDK
Vietnamese Version

Programmer's Guide & API References

V2.1
Updated: 28 July 2009

Product Support

IT WORKS expects that every user of *ID-WORKS Integrator™ SDK* can utilize this SDK in any project as designed.

For further information or if you would like to contact our engineer please contact us via:

E-mail (*Preferred*) ***vietnam@itworks.co.th***

Comments

It is always our goal at IT WORKS to supply accurate and reliable documentation. If you discover a discrepancy in this document, please email your comments to us.

E-mail (*Preferred*) *vietnam@itworks.com.th*

Or

*IT WORKS Ltd. (Vietnam Satellite Office)
157- 159 Nguyen Dinh Chieu, Dist.3, HCMC., Vietnam
TEL: 08 3 9308925*

*IT WORKS Ltd. (HQ)
100/103 Wongwanich Complex B Tower, 30th Fl, Rama IX
Road, Huaykwang Subdistrict, Huaykwang District, Bangkok
Thailand TEL: (+662) 645-1200 FAX: (+662) 645-1233*

Disclaimer

IT WORKS makes no representations or warranties, either expressed or implied, with respect to the content hereof and specifically disclaims any warranties, merchantability or fitness for any particular purpose. Any software described in this document is sold or licensed "as is".

IT WORKS reserves the right to revise this document and to make changes from time to time in the contents hereof without obligation to notify any person of such revision or changes.

Trademark Disclosures

IT WORKS LTD. has made every effort to provide disclosures when using trademarks owned by another companies. Trademarked designations appear throughout this publication. The publisher states that it is using the designations only for editorial purposes, and to the benefits of the trademark owner with no intent to infringe upon that trademark. The following trademarks are found in this manual:

Microsoft™, Windows 7™, Windows Vista™, Windows XP™, Windows 98™, Windows 2000™, Windows Millennium™ and Windows NT™ are trademarks of Microsoft Corporation

DigitalPersona™, U.are.U™ are trademarks of Digital Persona Inc.

ID-WORKS™, ID-WORKS Integrator™, TimeWORKS Integrator™, TimeWORKS™ **และ** are the trademark of IT WORKS LTD.

Index

PRODUCT SUPPORT	II
COMMENTS	II
DISCLAIMER	II
TRADEMARK DISCLOSURES	II
INDEX	III
INTRODUCTION	1
SYSTEM REQUIREMENTS	2
IDWORKS INTEGRATOR API	3
IDWORKS INTEGRATOR FUNCTION LIST 1- FINGERPRINT PROCESSING	3
Initiate and finalize the library	3
Create/Close context and Activation.....	3
Control the Fingerprint Scanner	3
Adding and Deleting the fingerprint	3
IDWORKS INTEGRATOR FUNCTION LIST 2- FINGERPRINT ENROLLMENT	3
Initiate and finalize the library	3
Create/Close context and Activation.....	4
Fingerprint Enrollment	4
IDWORKS INTEGRATOR FUNCTION LIST 3- FINGERPRINT SCANNER.....	4
Initiate and finalize the library	4
Fingerprint Scanner Data	4
IDWORKS INTEGRATOR – FINGERPRINT PROCESSING	5
Function Definitions	5
idi_InitModule	5
idi_FinalizeModule.....	5
idi_CreateContext	6
idi_CloseContext.....	6
idi_ActivateSensor	7
idi_DeactivateSensor.....	8
idi_StartCapturing.....	9
idi_StopCapturing	10
idi_RegisterSensorEventProc.....	11
idi_UnregisterSensorEventProc	11
idi_RegisterIdentEventProc.....	12
idi_UnregisterIdentEventProc.....	13
idi_fp_Add	14
idi_fp_ClearAll	14
Data Types	16
IDI_CONTEXT	16
IDI_IDENTEVENTMSG.....	16
IDI_IDENTEVENTTYPE.....	17
IDI_IDENTEVENTPARAM.....	17
IDI_REFPARAM	17

IDI_IDENTEVENTPROC	18
IDI_IDENTEVENTMATCHEDSTRUCT.....	18
IDI_SENSOREVENTMSG	19
IDI_SENSOREVENTPROC	20
IDI_SENSOREVENTIMAGEACQUIREDSTRUCT	20
FP_FPINFO.....	21
Constants	23
Constants for IDI_RCODE.....	23
Constants for IDI_IDENTEVENTTYPE	23
Constants for IDI_SENSORTYPE	23
Constants for IDI_SENSOREVENTTYPE.....	23
ID-WORKS INTEGRATOR – FINGERPRINT SCANNER.....	24
Function Definitions	24
isl_IsModuleInited	24
isl_InitModule.....	24
isl_FinalizeModule	25
isl_GetDeviceCount.....	25
isl_GetDeviceList	26
Data Types	27
ISL_RCODE	27
ISL_DEVICETYPE.....	27
ISL_DEVICEID.....	27
ISL_CHAR	28
ISL_SENSORINFO.....	28
Constants	30
General Constants	30
Constants for ISL_RCODE.....	30
Constants for ISL_DEVICETYPE	30
ID-WORKS INTEGRATOR – FINGERPRINT ENROLLMENT.....	31
Function Definitions	31
fpe_InitModule	31
fpe_FinalizeModule	31
fpe_CreateContext.....	32
fpe_CloseContext.....	32
fpe_ActivateSensor	33
fpe_StartEnrolment.....	34
Data Types	36
FPE_RCODE	36
FPE_CONTEXT.....	36
FPE_SENSORTYPE	36
Constants	37
General Constants	37
Constants for ISL_RCODE.....	37
Constants for ISL_DEVICETYPE	37

Introduction

IT WORKS develops ID-WORKS Integrator for developers to integrate the advance 1:N fingerprint matching technology with your applications. Following are some applications that can benefited greatly by including our ID WORKS Integrator:

- Time & Attendance Applications
- Healthcare Applications
- Membership Management Application
- Childcare Security Applications
- Point of Sale Applications
- Library Management Systems
- Secure Data Storage Card/Smart Card Applications
- Physical/Logical Access Control Applications
- Internet/Financial Transaction Authentication

ID-WORKS Integrator SDK was designed to work on Windows Platform and bundled with DigitalPersona U.are.U 4000 fingerprint scanner. The standard version will only work with U.are.U 4000 scanner. However, this SDK can be modified to use with any scanners if necessary by request.

ID WORKS Integrator has superior searching performance compared to most of the scanner's SDK in the market. i.e. DigitalPersona's SDK-DLL Gold Version and DigitalPersona's SDK-DCOM Platinum. ID WORKS Integrator supports Optimized 1:N (Identification) matching (> 3,000 fingerprints/sec on P4 1.8 GHz) this is at least 20-30 times faster than most of the SDK in the market. Most of the time, developers can develop the application that can identify people using just the fingerprint without introducing any PIN or ID.

ID WORKS Integrator can be used with the language that support DLL i.e. C, Delphi, VB. The SDK include sample applications in each language, API and source codes.

System Requirements

Minimum Requirements

- Pentium III PC or better
- Processor Clock 800 MHz or better
- HDD space 20 MB
- RAM 64 MB
- 1 USB Port
- DigialPersona U.are.U 4000
- DigitalPersona U.are.U Integrator Gold (included with IDWORKS Integrator SDK)

Recommend System

- Pentium 4 PC or better
- Processor clock 1.6 GHz หรือเร็วกว่า
- HDD space 500 MB
- RAM 256 MB
- 1 USB Port
- DigialPersona U.are.U 4000
- DigitalPersona U.are.U Integrator Gold (included with IDWORKS Integrator SDK)

IDWORKS Integrator API

IDWORKS Integrator Function List 1- Fingerprint Processing

Initiate and finalize the library

Function Name	Description
<code>idi_InitModule</code>	Initiate the library
<code>idi_FinalizeModule</code>	Finalize the library

Create/Close context and Activation

Function name	Description
<code>idi_CreateContext</code>	Create IDWORKS processor context
<code>idi_CloseContext</code>	Close IDWORKS processor context
<code>idi_ActivateSensor</code>	Register the U.are.U fingerprint scanner before using it

Control the Fingerprint Scanner

Function name	Description
<code>idi_StartCapturing</code>	Start capturing the fingerprint
<code>idi_StopCapturing</code>	Stop capturing the fingerprint
<code>idi_RegisterIdentEventProc</code>	register callback function to return the fingerprint searching result back to the application
<code>idi_UnregisterIdentEventProc</code>	unregister callback function that was registered by <code>idi_RegisterIdentEventProc</code> function

Adding and Deleting the fingerprint

Function name	Description
<code>idi_fp_Add</code>	Add fingerprint into the temporary database in context
<code>idi_fp_ClearAll</code>	Delete all fingerprints inside the temporary database in context

IDWORKS Integrator Function List 2- Fingerprint Enrollment

Initiate and finalize the library

Function name	Description
<code>fpe_InitModule</code>	Initiate the library
<code>fpe_FinalizeModule</code>	Finalize the library

Create/Close context and Activation

Function name	Description
<code>fpe_CreateContext</code>	Create context for fingerprint enrollment
<code>fpe_CloseContext</code>	Close context for fingerprint enrollment
<code>fpe_ActivateSensor</code>	Register the U.are.U fingerprint scanner before using it

Fingerprint Enrollment

Function name	Description
<code>fpe_StartEnrolment</code>	Start the Enrollment process

IDWORKS Integrator Function List 3- Fingerprint Scanner

Initiate and finalize the library

Function name	Description
<code>isl_InitModule</code>	Initiate the library
<code>isl_FinalizeModule</code>	Finalize the library
<code>isl_IsModuleInited</code>	Check the Initialize status

Fingerprint Scanner Data

Function name	Description
<code>isl_GetDeviceCount</code>	Count the number of fingerprints scanners in the system
<code>isl_GetDeviceList</code>	Get all data of Fingerprint Scanners in the system

IDWORKS Integrator – Fingerprint Processing

DLL Name : TMWIDW.DLL

Unless there are other remarks in the manual, all functions will return value IDI_R_OK after complete.

Function Definitions

idi_InitModule

IDWORKS Integrator Module: TMWIDW.DLL

Before using this module, this function must be called.

Function Definition

C/C++-style function definition:

```
void __stdcall idi_InitModule(void);
```

Pascal-style function definition:

```
procedure idi_InitModule; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Sub idi_InitModule Lib "TMWIDW.DLL" ()
```

Parameters

No parameter

Return Value

No return value

idi_FinalizeModule

IDWORKS Integrator Module: TMWIDW.DLL

After using this module, this function must be called to clear all remaining data

Function Definition

C/C++-style function definition:

```
void __stdcall idi_FinalizeModule(void);
```

Pascal-style function definition:

```
procedure idi_FinalizeModule; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Sub idi_FinalizeModule Lib "TMWIDW.DLL" ()
```

Parameters

No parameter

Return Value

No return value

idi_CreateContext

IDWORKS Integrator Module: TMWIDW.DLL

Function Definition

C/C++-style function definition:

```
void __stdcall idi_CreateContext(
    IDI_CONTEXT *AProcessorContext
);
```

Pascal-style function definition:

```
function idi_CreateContext(
    out AProcessorContext : IDI_CONTEXT
): IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_CreateContext Lib "TMWIDW.DLL" _
    (ByRef AProcessorContext As Long) _
    As Long
```

Parameters

AProcessorContext

If context creation is success, AProcessorContext will return Context value out

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

idi_CloseContext

IDWORKS Integrator Module: TMWIDW.DLL

idi_CloseContext function is used to close the context

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_CloseContext(
    IDI_CONTEXT AProcessorContext
);
```

Pascal-style function definition:

```
function idi_CloseContext(
    AProcessorContext: IDI_CONTEXT
): IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_CloseContext Lib "TMWIDW.DLL" _
    (ByVal AProcessorContext As Long) _
    As Long
```

Parameters

AProcessorContext
Context that will be closed

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

idi_ActivateSensor

IDWORKS Integrator Module: TMWIDW.DLL

This is the function to activate the Sensor by sending required parameter in.

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_ActivateSensor(
    IDI_CONTEXT: ProcessorContext,
    IDI_SENSORTYPE: SensorType,
    char *SensorSerialNo,
    char *AppSerial,
    char *AppKey
);
```

Pascal-style function definition:

```
function idi_ActivateSensor(
    ProcessorContext: IDI_CONTEXT;
    SensorType : IDI_SENSORTYPE;
    SensorSerialNo : PChar;
    AppSerial : PChar;
    AppKey : PChar
) : IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_ActivateSensor Lib "TMWIDW.DLL" _
    (ByVal AProcessorContext As Long, _
    ByVal ASensorType As Integer, _
    ByVal ASensorSerialNo As String, _
    ByVal AAppSerial As String, _
    ByVal AAppKey As String _
    ) As Long
```

Parameters

AProcessorContext

IDWORKS Processor Context

ASensorType

Type of Sensor

(Can be checked by looking at the constant starting with IDI_ST_)

ASensorSerialNo

Sensor Serial Number (null-terminated)

AAppSerial

IDWORKS Integrator Serial (null-terminated)

AAppKey

IDWORKS Integrator Key (null-terminated)

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

idi_DeactivateSensor

IDWORKS Integrator Module: TMWIDW.DLL

Use to stop using the Scanner according to the input serial no.

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_DeactivateSensor(
    IDI_CONTEXT AProcessorContext,
    IDI_SENSORTYPE ASensorType,
    char* ASensorSerialNo
);
```

Pascal-style function definition:

```
function idi_DeactivateSensor(
    AProcessorContext: IDI_CONTEXT;
    ASensorType: IDI_SENSORTYPE;
    ASensorSerialNo: PChar
): IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_DeactivateSensor Lib "TMWIDW.DLL" _
    (ByVal AProcessorContext As Long, _
    ByVal ASensorType As Long, _
    ByVal ASensorSerialNo As String _
    ) As Long
```

Parameters

AProcessorContext

IDWORKS Processor Context

ASensorType

Type of Sensor

(Can be checked by looking at the constant starting with IDI_ST_)

ASensorSerialNo

Sensor Serial Number (null-terminated)

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

idi_StartCapturing

IDWORKS Integrator Module: TMWIDW.DLL

When the function is called, the sensor will start the capturing process by waiting for the fingerprint. When the user touch the sensor the fingerprint will be processed automatically and the result will (ie. Invalid fingerprint, unrecognized fingerprint etc.) will be sent as a message via Callback Function that was registered by `idi_RegisterIdentEventProc` function.

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_StartCapturing(
    IDI_CONTEXT AProcessorContext,
    char *ASensorSerialNo
): IDI_RCODE;
```

Pascal-style function definition:

```
function idi_StartCapturing(
    AProcessorContext: IDI_CONTEXT;
    ASensorSerialNo : PChar
) : IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_StartCapturing Lib "TMWIDW.DLL" _
    (ByVal AProcessorContext As Long, _
    ByVal ASensorSerialNo As String _
    ) As Long
```

Parameters

AProcessorContext

IDWORKS Processor Context

ASensorSerialNo

Sensor Serial No of the sensor that will start the capturing process

Remark : Sensor can be used only after it was Activated by `idi_ActivateSensor` function

Return Value

The returned value will be `IDI_RCODE` meaning as follow

- `IDI_R_OK` : context is created successfully
- `IDI_R_MODULENOTINITED` : Module is not initialized
- `IDI_R_INVALIDCONTEXT` : Context is invalid

idi_StopCapturing

IDWORKS Integrator Module: `TMWIDW.DLL`

This function will stop the capturing process of the specified sensor.

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_StopCapturing(
    IDI_CONTEXT AProcessorHanle,
    char *ASensorSerialNo
);
```

Pascal-style function definition:

```
function idi_StopCapturing(
    AProcessorHanle: IDI_CONTEXT;
    ASensorSerialNo: PChar
): IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_StopCapturing Lib "TMWIDW.DLL" _
    (ByVal AProcessorHandle As Long, _
    ByVal ASensorSerialNo As String _
    ) As Long
```

Parameters

AProcessorContext

IDWORKS Processor Context

ASensorSerialNo

Sensor Serial No of the sensor that will stop the capturing process

Return Value

The returned value will be `IDI_RCODE` meaning as follow

- `IDI_R_OK` : context is created successfully
- `IDI_R_MODULENOTINITED` : Module is not initialized
- `IDI_R_INVALIDCONTEXT` : Context is invalid

idi_RegisterSensorEventProc

IDWORKS Integrator Module: TMWIDW.DLL

Register the callback function to receive event of fingerprint scanner

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_RegisterSensorEventProc(
    IDI_CONTEXT AProcessorContext,
    IDI_SENSOREVENTPROC AEventProc,
    IDI_REFPARAM ARefParam
);
```

Pascal-style function definition:

```
idi_RegisterSensorEventProc : function(
    AProcessorContext: IDI_CONTEXT;
    AEventProc: IDI_SENSOREVENTPROC;
    ARefParam: IDI_REFPARAM
): IDI_RCODE; stdcall
```

Visual Basic-style function definition:

```
Public Declare Function idi_RegisterSensorEventProc Lib "TMWIDW.DLL" _
    (ByVal AProcessorContext As Long, _
    ByVal AEventProc As Long, _
    ByRef ARefParam As Any _
    ) As Long
```

Parameters

AProcessorContext

IDWORKS Processor Context

AEventProc

Pointer or Address of Callback function that will be called when there is any event with the fingerprint scanner

ARefParam

Is the return value with Message in IDI_SensorEventMsg.RefParam (can be use as a reference ie. Can be use as Pointer of a specified object)

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

idi_UnregisterSensorEventProc

IDWORKS Integrator Module: TMWIDW.DLL

Unregistered the Callback function

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_UnregisterSensorEventProc(
    IDI_CONTEXT AProcessorContext,
    IDI_SENSOREVENTPROC AEventProc,
);
```

Pascal-style function definition:

```
idi_UnregisterSensorEventProc : function(
    AProcessorContext: IDI_CONTEXT;
    AEventProc: idi_SENSOREVENTPROC
): IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_UnregisterSensorEventProc Lib "TMWIDW.DLL" _
    (ByVal AProcessorContext As Long, _
    ByVal AEventProc As Long, _
    ) As Long
```

Parameters

AProcessorContext

IDWORKS Processor Context

AEventProc

Callback function that will be unregistered

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

idi_RegisterIdentEventProc

IDWORKS Integrator Module: TMWIDW.DLL

Register the Callback Function for fingerprint processing events.

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_RegisterIdentEventProc(
    IDI_CONTEXT AProcessorContext,
    IDI_IDENTEVENTPROC AEventProc,
    IDI_REFPARAM ARefParam
);
```

Pascal-style function definition:

```
function idi_RegisterIdentEventProc(
    AProcessorContext: IDI_CONTEXT;
    AEventProc: IDI_IDENTEVENTPROC;
    ARefParam: IDI_REFPARAM
) : IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_RegisterIdentEventProc Lib "TMWIDW.DLL" _
    (ByVal AProcessorContext As Long, _
    ByVal AEventProc As Long, _
    ByRef ARefParam As Any _
    ) As Long
```

Parameters

AProcessorContext

IDWORKS Processor Context

AEventProc

Callback function that will be called when there is a processing

ARefParam

Is the return value with Message in IDI_SensorEventMsg.RefParam (can be use as a reference ie. Can be use as Pointer of a specified object)

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

idi_UnregisterIdentEventProc

IDWORKS Integrator Module: TMWIDW.DLL

Unregistered the function that was previously registered with `idi_RegisterIdentEventProc`

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_UnregisterIdentEventProc(
    IDI_CONTEXT AProcessorContext,
    IDI_IDENTEVENTPROC AEventProc
);
```

Pascal-style function definition:

```
function idi_UnregisterIdentEventProc(
    AProcessorContext: IDI_CONTEXT;
    AEventProc: IDI_IDENTEVENTPROC
): IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_UnregisterIdentEventProc Lib "TMWIDW.DLL" _
    (ByVal AProcessorContext As Long, _
    ByVal AEventProc As Long _
    ) As Long
```

Parameters

AProcessorContext

IDWORKS Processor Context

AEventProc

Callback function that will be unregistered

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

idi_fp_Add

IDWORKS Integrator Module: TMWIDW.DLL

Add fingerprint into a temporary database . This will make that fingerprint available for processing

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_fp_Add(
    IDI_CONTEXT AProcessorContext,
    FP_FPINFO *AFPInfoPtr
);
```

Pascal-style function definition:

```
function idi_fp_Add(
    AProcessorContext: IDI_CONTEXT;
    AFPInfoPtr: FP_FPINFOPTR
): IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_fp_Add Lib "TMWIDW.DLL" _
    (ByVal AProcessContext As Long, _
    ByRef AFPInfo As FP_FPInfo _
    ) As Long
```

Parameters

AProcessorContext

IDWORKS Processor Context ;

AFPInfoPtr

Pointer that point to fingerprint data (FP_FPInfo) that will be added to the

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

idi_fp_ClearAll

IDWORKS Integrator Module: TMWIDW.DLL

Clear all data in the temporary database

Function Definition

C/C++-style function definition:

```
IDI_RCODE __stdcall idi_fp_ClearAll(  
    IDI_CONTEXT AProcessorContext  
);
```

Pascal-style function definition:

```
function idi_fp_ClearAll(  
    AProcessorContext: IDI_CONTEXT  
): IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function idi_fp_ClearAll Lib "TMWIDW.DLL" _  
    (ByVal AProcessorContext As Long _  
    ) As Long
```

Parameters

AProcessorContext
IDWORKS Processor Context

Return Value

The returned value will be IDI_RCODE meaning as follow

- IDI_R_OK : context is created successfully
- IDI_R_MODULENOTINITED : Module is not initialized
- IDI_R_INVALIDCONTEXT : Context is invalid

Data Types

IDI_CONTEXT

IDI_CONTEXT is a 32-bit data type for Processor Context that acts like a processing unit. More than one context can be created separately at the same time.

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long IDI_CONTEXT;
```

Pascal-style data type definition:

```
IDI_CONTEXT = Cardinal;
```

Visual Basic-style data type definition:

Use Long

IDI_IDENTEVENTMSG

IDI_IDENTEVENTMSG is a structure data type that will be returned to callback function registered with `idi_RegisterIdentEventProc`

Data-type Definition

C/C++-style data type definition:

```
struct IDI_IDENTEVENTMSG {
    IDI_CONTEXT ProcessorContext;
    IDI_IDENTEVENTTYPE MsgID;
    IDI_IDENTEVENTPARAM MsgParam;
    IDI_REFPARAM RefParam;
};
```

Pascal-style data type definition:

```
IDI_IDENTEVENTMSG = record
    ProcessorContext: IDI_CONTEXT;
    MsgID: IDI_IdentEventType;
    MsgParam: IDI_IdentEventParam;
    RefParam: IDI_REFPARAM;
End;
```

Visual Basic-style data type definition:

```
Public Type IDI_IdentEventMsg
    ProcessorContext As Long
    MsgID As Long
    MsgParam As Long
    RefParam As Long
End Type
```

Remark

Return value in MsgParam will be depended on MsgID that will be explained in the following table

MsgID value	Description of value in MsgParam
IDI_ID_FINGERPRINTMATCHED	Pointer ที่ชี้ไปยัง IDI_IDENTEVENTMATCHSTRUCT
IDI_ID_FINGERPRINTNOTMATCHED	0

IDI_IDENTEVENTTYPE

IDI_IDENTEVENTTYPE is a 32-bit data type that represents type of identification message events of IDI_IDENTEVENTMSG, which returned to registered callback function that previously registered by using RegisterIdentEventProc.

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long IDI_IDENTEVENTTYPE;
```

Pascal-style data type definition:

```
IDI_IDENTEVENTTYPE = Cardinal;
```

Visual Basic-style data type definition:

```
Long
```

IDI_IDENTEVENTPARAM

IDI_IDENTEVENTPARAM is a 32-bit size data type that represents parameter that returned together with identification message IDI_IDENTEVENTMSG. The meaning of this parameter is different for each message depends on IDI_IDENTEVENTMSG.MsgID

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long IDI_IDENTEVENTPARAM;
```

Pascal-style data type definition:

```
IDI_IDENTEVENTPARAM = Cardinal;
```

Visual Basic-style data type definition:

```
Long
```

IDI_REFPARAM

IDI_REFPARAM is a pointer that can be used as custom reference when using `idi_RegisterIdentEventProc`. This value will be returned in `IDI_IDENTEVENTMSG.RefParam`, and will be the same value as `ARefParam` that was passed to the `idi_RegisterIdentEventProc` function.

Data-type Definition

C/C++-style data type definition:

```
typedef void* IDI_IDENTREFPARAM;
```

Pascal-style data type definition:

```
IDI_IDENTREFPARAM = Pointer;
```

Visual Basic-style data type definition:

```
Long
```

IDI_IDENTEVENTPROC

IDI_IDENTEVENTPROC is procedure-type declaration that use for callback when the sensor was touched. The pointer of function that passed as AEventProc parameter to `idi_RegisterIdentEventProc` must have the same arguments as those declared in IDI_IDENTEVENTPROC.

Data-type Definition

C/C++-style data type definition:

```
typedef void __stdcall (*IDI_IDENTEVENTPROC)(
    IDI_IDENTEVENTMSG *AIdentEventMsgPtr
);
```

Pascal-style data type definition:

```
IDI_IDENTEVENTPROC = procedure(
    AIdentEventMsgPtr: IDI_IdentEventMsgPtr
); stdcall;
```

Visual Basic-style data type definition:

```
Public Sub IdentProc(ByRef AIdentEventMsg As IDI_IdentEventMsg)
```

Parameters

AIdentEventMsgPtr

Pointer that point to IDI_IDENTEVENTMSG that tells the Message details.

IDI_IDENTEVENTMATCHEDSTRUCT

In case that the fingerprint was found when the user touches the sensor, IDI_IDENTEVENTMATCHEDSTRUCT is the data return with IDI_IDENTEVENTMSG in IDI_IDENTEVENTMSG.MsgParam member.

Data-type Definition

C/C++-style data type definition:

```
struct IDI_IDENTEVENTMATCHEDSTRUCT {
    FP_FPInfo *FPInfoPtr;
    int MatchingScore;
};
```

Pascal-style data type definition:

```
IDI_IDENTEVENTMATCHEDSTRUCT = record
    FPInfoPtr: FP_FPInfoPtr;
    MatchingScore: Integer;
end;
```

Visual Basic-style data type definition:

```
Public Type IDI_IdentEventMatchedStruct
    FPInfoPtr As Long
    MatchingScore As Integer
End Type
```

Members

FPInfoPtr

Is the Pointer that point to FP_FPINFO that tells the details of the fingerprint

MatchingScore

is the Matching score telling the similarity.

IDI_SENSOREVENTMSG

IDI_SENSOREVENTMSG is a structure data type that will be sent back to callback function that was registered using `idi_RegisterSensorEventProc`

Data-type Definition

C/C++-style data type definition:

```
struct IDI_SENSOREVENTMSG {
    IDI_CONTEXT ProcessorContext;
    IDI_SENSOREVENTTYPE MsgID;
    IDI_SENSOREVENTPARAM MsgParam;
    IDI_REFPARAM RefParam;
};
```

Pascal-style data type definition:

```
IDI_SENSOREVENTMSG = record
    ProcessorContext: IDI_CONTEXT;
    MsgID: IDI_SENSOREVENTTYPE;
    MsgParam: IDI_SENSOREVENTPARAM;
    RefParam: IDI_REFPARAM;
End;
```

Visual Basic-style data type definition:

```
Public Type IDI_SensorEventMsg
    ProcessorContext As Long
    MsgID As Long
    MsgParam As Long
    RefParam As Long
End Type
```

Members

ProcessorContext

Context that send message back

MsgID

Type of Message

MsgParam

Message parameter

RefParam

Reference number that was previously sent in when calling `idi_RegisterSensorEventProc`

Remark

Return value in `MsgParam` will depend on `MsgID` that will be explained in the following table

MsgID value	Description of value in <code>MsgParam</code>
<code>IDI_SE_IMAGEACQUIRED</code>	Pointer that point to <code>IDI_SENSORIMAGEACQUIREDSTRUCT</code>
<code>IDI_SE_IMAGEPROCESSED</code>	Pointer that point to <code>IDI_SENSORIMAGEACQUIREDSTRUCT</code>

IDI_SENSOREVENTPROC

`IDI_SENSOREVENTPROC` is type declaration of function that use for callback when there is any event with the sensor ie. receiving the image. This type of function will be used with `IDI_RegisterSensorEventProc`. The pointer of function passed as `AEventProc` parameter for `IDI_RegisterSensorEventProc` must have the same arguments as those declared in `IDI_SENSOREVENTPROC`.

Data-type Definition

C/C++-style data type definition:

```
typedef void __stdcall (*IDI_SENSOREVENTPROC)(
    IDI_SENSOREVENTMSG *ASensorEventMsgPtr
);
```

Pascal-style data type definition:

```
IDI_SENSOREVENTPROC = procedure(
    ASensorEventMsgPtr: IDI_SensorEventMsgPtr
); stdcall;
```

Visual Basic-style data type definition:

```
Public Sub SensorProc(ByRef ASensorEventMsg As IDI_SensorEventMsg)
```

Parameters
ASensorEventMsgPtr

Pointer that point to `IDI_SENSOREVENTMSG` that tells the Message details

IDI_SENSOREVENTIMAGEACQUIREDSTRUCT

When the sensor is touched and the image is captured, event will be sent to callback function that previously registered with `idi_RegisterSensorEvent`. The pointer of `IDI_SENSOREVENTIMAGEACQUIREDSTRUCT` is stored in `MsgParam` of that event

Data-type Definition

C/C++-style data type definition:

```
struct IDI_IDI_SENSOREVENTIMAGEACQUIREDSTRUCT {
    int ImageWidth;
    int ImageHeight;
    void* ImageBitmapPtr;
};
```

Pascal-style data type definition:

```
IDI_SENSOREVENTIMAGEACQUIREDSTRUCT = record
  ImageWidth : Integer;
  ImageHeight : Integer;
  ImageBitmapPtr : Pointer;
end;
```

Visual Basic-style data type definition:

```
Public Type IDI_SENSOREVENTIMAGEACQUIREDSTRUCT
  ImageWidth As Long
  ImageHeight As Long
  ImageBitmapPtr As Long
End Type
```

Members

ImageWidth

Fingerprint image's width

ImageHeight

Fingerprint image's height

ImageBitmapPtr

Pointer that point to grayscale bitmap data of fingerprint that have the size equal to ImageWidth x ImageHeight in the form of 8-bit grayscale

FP_FPINFO

FP_FPINFO is structure data type that represents fingerprint data. The fingerprint data passed into the context temporary database must be passed as FP_FPINFO.

Data-type Definition

C/C++-style data type definition:

```
struct FP_FPINFO {
  int FingerprintIndexingValue;
  int FingerIndex;
  void *FingerprintRawDataPtr;
  unsigned long FingerprintRawDataSize;
  int Tag;
}
```

Pascal-style data type definition:

```
FP_FPINFO = record
  FingerprintIndexingValue : Integer;
  FingerIndex : Integer;
  FingerprintRawDataPtr: PByte;
  FingerprintRawDataSize: Cardinal;
  Tag : Integer;
end;
```

Visual Basic-style data type definition:

```
Public Type FP_FPInfo
  FingerprintIndexingValue As Long
```

```
FingerIndex As Long
FingerprintRawDataPtr As Long
FingerprintRawDataSize As Long
Tag As Long
End Type
```

Members

FingerprintIndexingValue

is the Indexing value of the fingerprint that is used to optimize the searching performance.

FingerIndex

is them number to specify the finger. 0 is equal to left pinky and 9 is equal to the right pinky.

FingerprintRawDataPtr

is the Pointer that point to memory buffer that store the fingerprint data

FingerprintRawDataSize

is the size of memory buffer that is used by FingerprintRawDataPtr

Tag

Is the reference number of the fingerprint that the developer can be used i.e. reference ID of people in the database

Constants

Constants for IDI_RCODE

Constant Name	Value	Description
IDI_R_OK	0	Finish without any error
IDI_R_GENERALFAILURE	-1	General Failure
IDI_R_INIT_FAILED	-2	Initiation fail
IDI_R_MODULENOTINITED	-3	Call function without Initialize Module
IDI_R_SENSORNOTFOUND	-4	Sensor is not found
IDI_R_SENSORALREADYACTIVATED	-5	Sensor's been activated already
IDI_R_INVALIDSENSORTYPE	-6	Invalid sensor type
IDI_R_INVALIDREGISTRATION	-7	Invalid Activation
IDI_R_INVALIDCONTEXT	-16	Invalid Context

Constants for IDI_IDENTEVENTTYPE

Constant Name	Value	Description
IDI_ID_FINGERPRINTMATCHED	0	Fingerprint matched
IDI_ID_FINGERPRINTNOTMATCHED	1	Fingerprint not matched

Constants for IDI_SENSORTYPE

Constant Name	Value	Description
IDI_ST_UAREU	1	DigitalPersona U.are.U 4000 fingerprint scanner

Constants for IDI_SENSOREVENTTYPE

Constant Name	Value	Description
IDI_SE_IMAGEACQUIRED	1	Receiving the image event
IDI_SE_IMAGEPROCESSED	2	Image was processed event

ID-WORKS Integrator – Fingerprint Scanner

DLL Name: ITWSSL.DLL

is the library to retrieve the data of all connected scanners

Function Definitions

isl_IsModuleInited

IDWORKS Integrator Module: ITWSSL.DLL

Check if the module was initialized or not

Function Definition

C/C++-style function definition:

```
ISL_RCODE __stdcall isl_IsModuleInited(void);
```

Pascal-style function definition:

```
function isl_IsModuleInited : ISL_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function isl_IsModuleInited Lib "ITWSSL.DLL" () As Long
```

Parameters

No parameters

Return Value

- ISL_R_OK : module was initialized
- ISL_R_MODULENOTINITED : module was not initialized

isl_InitModule

IDWORKS Integrator Module: ITWSSL.DLL

Start the module

Function Definition

C/C++-style function definition:

```
ISL_RCODE __stdcall isl_InitModule(void);
```

Pascal-style function definition:

```
function isl_InitModule : ISL_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function isl_InitModule Lib "ITWSSL.DLL" () As Long
```

Parameters

No parameters

Return Value

- ISL_R_OK : success
- ISL_R_MODULENOTINITED : can not initiates module

isl_FinalizeModule

IDWORKS Integrator Module: ITWSSL.DLL

Stop module (i.e. when closing the program) clear all remaining data before close

Function Definition

C/C++-style function definition:

```
ISL_RCODE __stdcall isl_FinalizeModule(void);
```

Pascal-style function definition:

```
function isl_FinalizeModule : ISL_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function isl_FinalizeModule Lib "ITWSSL.DLL" () As Long
```

Parameters

No parameters

Return Value

- ISL_R_OK : success
- ISL_R_MODULENOTINITED : can not initiates module

isl_GetDeviceCount

IDWORKS Integrator Module: ITWSSL.DLL

Count the total number of connected sensors

Function Definition

C/C++-style function definition:

```
ISL_RCODE __stdcall isl_GetDeviceCount(
    unsigned int *ASensorCount
);
```

Pascal-style function definition:

```
function isl_GetDeviceCount(
    out ASensorCount: Cardinal
): ISL_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function isl_GetDeviceCount Lib "ITWSSL.DLL" _
    (ByRef ASensorCount As Long _
    ) As Long
```

Parameters

ASensorCount
No. of sensor

Return Value

- ISL_R_OK : success, the number of sensor will be stored in ASensorCount
- ISL_R_MODULENOTINITED : function isl_InitModule is not called

isl_GetDeviceList

IDWORKS Integrator Module: ITWSSL.DLL

Use to retrieve data of all connected sensors

Function Definition

C/C++-style function definition:

```
ISL_RCODE __stdcall isl_GetDeviceList(
    FP_FPINFO[] ASensorList,
    unsigned int ASensorCount
);
```

Pascal-style function definition:

```
function isl_GetDeviceList(
    ASensorList : Pointer;
    ASensorCount : Cardinal
) : ISL_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function isl_GetDeviceList Lib "ITWSSL.DLL" _
    (ByVal ASensorList As Long, _
    ByVal ASensorCount As Long _
    ) As Long
```

Parameters

ASensorList
Is the pointer of array (C/C++) of FP_FPInfo that have to have the size to be at least equal to ASensorCount

ASensorCount
Is the no. of sensor that will be retrieve the data. Use isl_GetDeviceCount to get the no.

Return Value

- ISL_R_OK : success, the number of sensor will be stored in ASensorCount
- ISL_R_MODULENOTINITED : function isl_InitModule is not called

Data Types

ISL_RCODE

ISL_RCODE is the 32-bit data type that is the return value when function was called to tell the result.

Data-type Definition

C/C++-style data type definition:

```
typedef int ISL_RCODE;
```

Pascal-style data type definition:

```
ISL_RCODE = Integer
```

Visual Basic-style data type definition:

```
Long
```

ISL_DEVICETYPE

ISL_DEVICETYPE is the 32-bit data type that use to specify the type of scanner

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long ISL_RCODE;
```

Pascal-style data type definition:

```
ISL_DEVICETYPE = Cardinal;
```

Visual Basic-style data type definition:

```
Long
```

ISL_DEVICEID

ISL_DEVICEID is the 64-bit data type that is used as the device ID

Data-type Definition

C/C++-style data type definition:

```
typedef __int64 ISL_RCODE;
```

Pascal-style data type definition:

```
ISL_DEVICEID = int64;
```

Visual Basic-style data type definition:

```
IN Visual Basic, there is no Integer 64 bit size but the included header file of Visual Basic in this SDK 's solved this structure problem
```

ISL_CHAR

ISL_CHAR is the 8-bit data type that use to store character in the module

Data-type Definition

C/C++-style data type definition:

```
typedef char ISL_CHAR;
```

Pascal-style data type definition:

```
ISL_CHAR = Char;
```

Visual Basic-style data type definition:

```
use Byte
```

ISL_SENSORINFO

ISL_SENSORINFO is the structure data type that is used to store scanners data

Data-type Definition

C/C++-style data type definition:

```
struct ISL_SENSORINFO {
    ISL_DEVICETYPE DeviceType;
    ISL_DEVICEID DeviceId;
    ISL_CHAR DeviceName[ISL_MAXDEVICENAMELEN];
    ISL_CHAR DeviceSerial[ISL_MAXDEVICERIALLEN];
}
```

Pascal-style data type definition:

```
ISL_SENSORINFO = record
    DeviceType: ISL_DEVICETYPE;
    DeviceId: ISL_DEVICEID;
    DeviceName: Array[0..ISL_MAXDEVICENAMELEN-1] of ISL_CHAR;
    DeviceSerial: Array[0..ISL_MAXDEVICERIALLEN-1] of ISL_CHAR;
end;
```

Visual Basic-style data type definition:

```
Public Type ISL_SENSORINFO
    DeviceType As Long
    DeviceId As Long
    DeviceName(ISL_MAXDEVICENAMELEN) As Byte
    DeviceSerial(ISL_MAXDEVICERIALLEN) As Byte
End Type
```

Members

DeviceType
Type of scanner

DeviceID
Device ID

DeviceName
Device Name i.e. DigitalPersona U.are.U 4000

DeviceSerial
S/N of fingerprint scanner

Constants

General Constants

Constant Name	Value	Description
ISL_MAXDEVICENAMELEN	64	Maximum length of device name string
ISL_MAXDEVICERIALLEN	64	Maximum length of device serial string

Constants for ISL_RCODE

Constant Name	Value	Description
ISL_R_OK	0	Operation succeed
ISL_R_MODULENOTINITED	-1	Module is not initialized
ISL_R_ALREADYINITED	-2	Module has been initialized already

Constants for ISL_DEVICETYPE

Constant Name	Value	Description
ISL_DT_UAREU	0	U.are.U sensor

ID-WORKS Integrator – Fingerprint Enrollment

DLL Name: FPENROL.DLL

All functions will return Value as FPE_R_OK if success

Function Definitions

fpe_InitModule

IDWORKS Integrator Module: FPENROL.DLL

Start the library

Function Definition

C/C++-style function definition:

```
FPE_RCODE __stdcall fpe_InitModule(void);
```

Pascal-style function definition:

```
function fpe_InitModule: FPE_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function fpe_InitModule Lib "FPENROL.DLL" () As Long
```

Parameters

No Parameter

Return Value

- FPE_R_OK : success
- FPE_R_MODULENOTINITED : module is not initiated

fpe_FinalizeModule

IDWORKS Integrator Module: FPENROL.DLL

Use when stop using the module

Function Definition

C/C++-style function definition:

```
FPE_RCODE __stdcall fpe_FinalizeModule(void);
```

Pascal-style function definition:

```
function fpe_FinalizeModule: FPE_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function fpe_FinalizeModule Lib "FPENROL.DLL" () As Long
```

Parameters

No Parameter

Return Value

- FPE_R_OK : success
- FPE_R_MODULENOTINITED : module is not initiated

fpe_CreateContext

IDWORKS Integrator Module: FPENROL.DLL

create context for fingerprint enrolment

Function Definition

C/C++-style function definition:

```
FPE_RCODE __stdcall fpe_CreateContext(
    FPE_CONTEXT* AEnrolmentContext
);
```

Pascal-style function definition:

```
function fpe_CreateContext(
    var AEnrolmentContext: FPE_CONTEXT
): FPE_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function fpe_CreateContext Lib "FPENROL.DLL" _
    (ByRef AEnrolmentContext As Long _
    ) As Long
```

Parameters

AEnrolmentContext

If success, the value is the context

Return Value

- FPE_R_OK : success
- FPE_R_MODULENOTINITED : module is not initiated

fpe_CloseContext

IDWORKS Integrator Module: FPENROL.DLL

close context of fingerprint enrolment

Function Definition

C/C++-style function definition:

```
FPE_RCODE __stdcall fpe_CloseContext(
    FPE_CONTEXT AEnrolmentContext
);
```

Pascal-style function definition:

```
function fpe_CloseContext(
    AEnrolmentContext: FPE_CONTEXT
) : FPE_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function fpe_CloseContext Lib "FPENROL.DLL" _
    (ByVal AEnrolmentContext As Long _
    ) As Long
```

Parameters

AEnrolmentContext

If success, AEnrolmentContext will be set as the value context used for fingerprint enrollment

Return Value

- FPE_R_OK : success
- FPE_R_MODULENOTINITED : module is not initiated

fpe_ActivateSensor

IDWORKS Integrator Module: FPENROL.DLL

Activate the sensor

Function Definition

C/C++-style function definition:

```
FPE_RCODE __stdcall fpe_ActivateSensor(
    FPE_CONTEXT AEnrolmentContext,
    char *ASensorSerialNo,
    char *AAppSerial,
    char *AAppKey
);
```

Pascal-style function definition:

```
function fpe_ActivateSensor(
    AEnrolmentContext: FPE_CONTEXT;
    ASensorSerialNo : PChar;
    AAppSerial : PChar;
    AAppKey : PChar
): FPE_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function fpe_ActivateSensor Lib "FPENROL.DLL" _
    (ByVal AEnrolmentContext As Long, _
    ByVal ASensorType As Long, _
    ByVal ASensorSerialNo As String, _
    ByVal AAppSerial As String, _
    ByVal AAppKey As String _
    ) As Long
```

Parameters

- AProcessorContext*
Context that used for the enrollment
- ASensorType*
Type of Sensor (check by the value that start with FPE_ST_)
- ASensorSerialNo*
Sensor Serial No. (null-terminated)
- AAppSerial*
IDWORKS Integrator Serial (null-terminated)
- AAppKey*
IDWORKS Integrator Key (null-terminated)

Return Value

- FPE_R_OK : Success and the sensor is activated
- FPE_R_MODULENOTINITED : module is not initiated

fpe_StartEnrolment

IDWORKS Integrator Module: FPENROL.DLL

Start the fingerprint enrollment process

Function Definition

C/C++-style function definition:

```
FPE_RCODE __stdcall fpe_StartEnrolment(
    FPE_CONTEXT AEnrolmentContext,
    void *ARawDataPtr,
    unsigned int AMaxRawDataSize,
    unsigned int *AOutputRawDataSize,
    int *AIndexingValue
);
```

Pascal-style function definition:

```
function fpe_StartEnrolment(
    AEnrolmentContext: FPE_CONTEXT;
    ARawDataPtr: PByte;
    AMaxRawDataSize: Cardinal;
    out AOutputRawDataSize: Cardinal;
    AIndexingValue: PInteger
) : FPE_RCODE; stdcall;
```

Visual Basic-style function definition:

```
Public Declare Function fpe_StartEnrolment Lib "FPENROL.DLL" _
    (ByVal AEnrolmentContext As Long, _
    ByVal AFingerIndex As Integer, _
    ByRef ARawDataPtr As Any, _
    ByVal AMaxRawDataSize As Long, _
    ByRef AOutputRawDataSize As Long, _
    ByRef AIndexingValue As Long _
    ) As Long
```

Parameters

AEnrolmentContext

Fingerprint Enrolment Context

ARawDataPtr

Pointer pointed to buffer for collecting the fingerprint binary

AMaxRawDataSize

Maximum raw data size that can store the binary data in ARawDatPtr

AOutputRawDataSize

the size of buffer used, 0 means the module is canceled by user

AIndexingValue

If not Null, then it's the Fingerprint Indexing Value

Return Value

- FPE_R_OK : Success and the sensor is activated
- FPE_R_MODULENOTINITED : module is not initiated

Remark

If the user press Cancel, the return valus is still FPE_R_OK but the AOutputRawDataSize is equal to 0.

Data Types

FPE_RCODE

ISL_RCODE is the 32-bit data type that is the return value when function was called to tell the result.

Data-type Definition

C/C++-style data type definition:

```
typedef int FPE_RCODE;
```

Pascal-style data type definition:

```
FPE_RCODE = Integer
```

Visual Basic-style data type definition:

```
Long
```

FPE_CONTEXT

FPE_CONTEXT is the 32-bit data type for context

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long FPE_CONTEXT;
```

Pascal-style data type definition:

```
FPE_CONTEXT = Cardinal;
```

Visual Basic-style data type definition:

```
Long
```

FPE_SENSORTYPE

FPE_SENSORTYPE is the 32-bit data type representing type of scanners

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long FPE_SENSORTYPE;
```

Pascal-style data type definition:

```
FPE_SENSORTYPE = Cardinal;
```

Visual Basic-style data type definition:

```
Long
```

Constants

General Constants

Constant Name	Value	Description
FPC_MAXRAWDATASIZE	4096	Maximum memory buffer size required for the enrollment process

Constants for ISL_RCODE

Constant Name	Value	Description
FPE_R_OK	0	Success - No error
FPE_R_INVALIDCONTEXT	-1	Invalid Context
FPE_R_GENERALFAILURE	-2	General Failure
FPE_R_MODULENOTINITED	-3	Module is not Initialized
FPE_R_SENSORNOTFOUND	-4	Sensor is not found
FPE_R_INTERNALERROR	-4096	Internal error in the module

Constants for ISL_DEVICETYPE

Constant Name	Value	Description
FPE_ST_UAREU	1	Type of the scanner