



IT WORKS
IDWORKS Integrator SDK (DLLs)
API References

V2.1
Updated: 05 August 2008

การขั้พพอร์ทผลิตภัณฑ์

ไอที เวิร์คส์ ตั้งใจที่จะให้ผู้พัฒนาที่ใช้งาน *IDWORKS Integrator™ SDK* สามารถใช้งานชุด SDK ดังกล่าวได้ตามความต้องการและได้รับความพึงพอใจสูงสุดในการใช้งานผลิตภัณฑ์

สำหรับผู้ใช้งานที่มีข้อสงสัยในการใช้งานหรือต้องการติดต่อกับทีมวิศวกรของไอที เวิร์คส์ กรุณาติดต่อผ่านทาง

E-mail (Preferred) biosupport@itworksolutions.com

MSN Messenger [idisupp001@hotmail.com](msn:idisupp001@hotmail.com)

การแจ้งข้อผิดพลาดในเอกสาร

สำหรับ “*IDWORKS Integrator SDK - API References*” ฉบับนี้ทางทีมงาน ไอที เวิร์คส์ ตั้งใจที่จะทำให้รายละเอียดในเอกสารมีความถูกต้องและครบถ้วนที่สุด ไอที เวิร์คส์ ขอสงวนสิทธิ์ในการตรวจสอบและแก้ไขเนื้อหาในเอกสารฉบับนี้เป็นระยะๆ โดยไม่จำเป็นต้องแจ้งต่อผู้ใดทั้งสิ้น ถ้าหากท่านพบข้อผิดพลาดใดๆในเอกสารฉบับนี้ กรุณาส่งอีเมลคำชี้แจงมายัง

E-mail (Preferred) info@itworksolutions.com

หรือติดต่อเราได้ที่
IT WORKS Ltd.
100/103 Wongwanich Complex B Tower, 30th Fl, Rama IX Road. Huaykwang Subdistrict, Huaykwang District, Bangkok 10320
TEL: (+662) 645-1200-7 FAX: (+662) 645-1233

คำชี้แจง

ไอที เวิร์คส์ ตั้งใจที่จะให้ผู้พัฒนาที่ใช้งาน *IDWORKS Integrator™ SDK* สามารถใช้งานผลิตภัณฑ์ ดังกล่าวได้อย่างที่ต้องการและได้รับความพึงพอใจสูงสุด อย่างไรก็ตาม ไอที เวิร์คส์ ขอสงวนสิทธิ์ไม่รับผิดชอบ หรือรับประกันไม่ว่าในกรณีใดๆทั้งสิ้นถึงความเหมาะสมและความเข้ากันได้ของ *IDWORKS Integrator™ SDK* กับระบบที่ท่านพัฒนา

IDWORKS Integrator™ SDK และ Software ทุกตัวของไอที เวิร์คส์ ที่แสดงในเอกสารฉบับนี้จำหน่าย ในรูปแบบของผลิตภัณฑ์สำเร็จรูป เว้นแต่กรณีที่มีการตกลงกันกับไอทีเวิร์คส์เป็นกรณีไป

เครื่องหมายการค้า

ไอทีเวิร์คส์ ตั้งใจที่จะปกป้องถึงเจ้าของผลิตภัณฑ์ทุกครั้งที่มีการกล่าวถึงเครื่องหมายการค้าของบริษัทอื่น เครื่องหมายการค้าจะมีการกล่าวถึงอยู่ในจุดต่างๆของเอกสารฉบับนี้ โดยไอที เวิร์คส์ใช้เครื่องหมายการค้าต่างๆ เพื่อผลในการอธิบายและเพื่อผลประโยชน์ของเจ้าของเครื่องหมายการค้าเท่านั้น ไอที เวิร์คส์ไม่มีความตั้งใจใดๆ ในการลอกเลียนแบบใดๆทั้งสิ้น

Microsoft™, Windows XP™, Windows 98™, Windows 2000™, Windows Millennium™, Windows NT™ เป็นเครื่องหมายการค้าของ บริษัท ไมโครซอฟท์ คอร์ปอเรชัน

DigitalPersona™, U.are.U™ เป็นเครื่องหมายการค้าของ บริษัท Digital Persona Inc.

IDWORKS™, IDWORKS Integrator™, TimeWORKS Integrator™, TimeWORKS™ ระบบลงเวลาด้วยลายนิ้วมือ เป็นเครื่องหมายการค้าของ บริษัท ไอที เวิร์คส์ จำกัด



สารบัญ

การซัพพอร์ตผลิตภัณฑ์	2
การแจ้งข้อผิดพลาดในเอกสาร	2
คำชี้แจง	2
เครื่องหมายการค้า	2
สารบัญ	3
บทนำ	6
ความต้องการระบบ	7
ความต้องการต่ำสุด	7
ความต้องการของระบบที่แนะนำ	7
IDWORKS INTEGRATOR API	8
สรุปฟังก์ชันทั้งหมดใน IDWORKS INTEGRATOR - ส่วนการประมวลผลลายนิ้วมือ	8
การเริ่มและเลิกใช้งานไลบรารี	8
การสร้างและปิด context และการ Activate	8
การควบคุมเครื่องอ่านลายนิ้วมือ	8
การเพิ่มและลบลายนิ้วมือ	8
สรุปฟังก์ชันทั้งหมดใน IDWORKS INTEGRATOR - ส่วนการลงทะเบียนลายนิ้วมือ	8
การเริ่มและเลิกใช้งานไลบรารี	8
การสร้างและปิด context และการ Activate	8
การลงทะเบียนลายนิ้วมือ	9
สรุปฟังก์ชันทั้งหมดใน IDWORKS INTEGRATOR - ส่วนข้อมูลเครื่องอ่านลายนิ้วมือ	9
การเริ่มและเลิกใช้งานไลบรารี	9
การดึงข้อมูลเครื่องอ่านลายนิ้วมือ	9
IDWORKS INTEGRATOR - ส่วนการประมวลผลลายนิ้วมือ	10
รายละเอียดฟังก์ชัน (Function Definitions)	10
idi_InitModule	10
idi_FinalizeModule	10
idi_CreateContext	11
idi_CloseContext	11
idi_ActivateSensor	12
idi_DeactivateSensor	13
idi_StartCapturing	14
idi_StopCapturing	15
idi_RegisterSensorEventProc	16
idi_UnregisterSensorEventProc	17
idi_RegisterIdentEventProc	18
idi_UnregisterIdentEventProc	19
idi_fp_Add	20
idi_fp_ClearAll	21



ประเภทข้อมูล (Data Types)	22
IDI_RCODE	22
IDI_CONTEXT	22
IDI_IDENTEVENTMSG.....	22
IDI_IDENTEVENTTYPE.....	23
IDI_IDENTEVENTPARAM.....	23
IDI_REFPARAM	24
IDI_IDENTEVENTPROC.....	24
IDI_IDENTEVENTMATCHEDSTRUCT.....	25
IDI_SENSOREVENTMSG	25
IDI_SENSOREVENTPROC	26
IDI_SENSOREVENTIMAGEACQUIREDSTRUCT	27
FP_FPINFO.....	28
ค่าคงที่ (Constants)	30
ค่าคงที่สำหรับ IDI_RCODE.....	30
ค่าคงที่สำหรับ IDI_IDENTEVENTTYPE.....	30
ค่าคงที่สำหรับ IDI_SENSORTYPE.....	30
ค่าคงที่สำหรับ IDI_SENSOREVENTTYPE	30
ID-WORKS INTEGRATOR – ส่วนข้อมูลเครื่องอ่านลายนิ้วมือ	31
รายละเอียดฟังก์ชัน (Function Definitions).....	31
isl_IsModuleInited	31
isl_InitModule.....	31
isl_FinalizeModule	32
isl_GetDeviceCount.....	32
isl_GetDeviceList	33
ประเภทข้อมูล (Data Types)	35
ISL_RCODE.....	35
ISL_DEVICETYPE.....	35
ISL_DEVICEID.....	35
ISL_CHAR	36
ISL_SENSORINFO.....	36
ค่าคงที่ (Constants)	38
ค่าคงที่ทั่วไป.....	38
ค่าคงที่สำหรับ ISL_RCODE.....	38
ค่าคงที่สำหรับ ISL_DEVICETYPE.....	38
ID-WORKS INTEGRATOR – ส่วนช่วยการบันทึกลายนิ้วมือ.....	39
รายละเอียดฟังก์ชัน (Function Definitions).....	39
fpe_InitModule	39
fpe_FinalizeModule	39
fpe_CreateContext.....	40
fpe_CloseContext.....	40
fpe_ActivateSensor	41
fpe_StartEnrolment.....	42
ประเภทข้อมูล (Data Types)	44
FPE_RCODE	44
FPE_CONTEXT.....	44
FPE_SENSORTYPE	44
ค่าคงที่ (Constants)	45



ค่าคงที่ทั่วไป.....	45
ค่าคงที่สำหรับ ISL_RCODE.....	45
ค่าคงที่สำหรับ ISL_DEVICETYPE.....	45
APPENDIX: A – การแก้ไขข้อผิดพลาดระหว่างการติดตั้ง.....	46
การแก้ไขข้อผิดพลาดระหว่างการติดตั้ง.....	46
APPENDIX: B – การใช้งานและการดูแลรักษาเครื่องอ่านลายนิ้วมือ.....	47
การใช้งานเครื่องอ่านลายนิ้วมืออย่างถูกต้อง.....	47
พยายามแตะให้กว้างครอบคลุมพื้นที่มากที่สุด.....	47
วางนิ้วให้ขนานกับแนวราบ.....	47
ไม่ควรกดแรงเกินไป.....	47
เล็บยาว?.....	47
อย่าหมุนนิ้ว.....	47
มือแห้ง.....	47
การป้องกันปัญหาที่อาจเกิดขึ้น.....	48
มือสกปรก?.....	48
แสงแดด.....	48
แอลกอฮอล์และน้ำยาทำความสะอาด.....	48
น้ำตาล.....	48
การทำความสะอาดเครื่องอ่านลายนิ้วมือ.....	48
วิธีที่แนะนำในการทำความสะอาด.....	48
ข้อควรระวังในการทำความสะอาด.....	48
คำถามและคำตอบเกี่ยวกับลายนิ้วมือ.....	49



บทนำ

ไอที เวิร์คส์ คือผู้นำในการนำเทคโนโลยีไบโอเมตริกส์มาสู่การพัฒนาและประยุกต์ใช้อย่างแพร่หลายใน แอปพลิเคชันต่างๆ เพื่อผลักดันการนำเทคโนโลยีการตรวจสอบลายนิ้วมือมาประยุกต์ใช้ในวงกว้าง เราได้พัฒนา ID-WORKS Integrator ขึ้นให้คุณพัฒนาแอปพลิเคชันด้วยเทคโนโลยีนี้ได้อย่างสะดวก รวดเร็วและมีประสิทธิภาพสูงสุด

นักพัฒนาสามารถนำ ID-WORKS Integrator SDK มาประยุกต์ใช้กับแอปพลิเคชันต่างๆ ของตนได้ในหลายๆกรณี เช่น การใช้ลายนิ้วมือแทนรหัสผ่าน การใช้ลายนิ้วมือแทนบัตรพนักงาน บัตรคนไข้ บัตรนักเรียน บัตรห้องสมุดและบัตรอื่นๆ เพื่อความสะดวกถูกต้องในการยืนยันตัวตนบุคคลและความประหยัดจากการที่ไม่ต้องจัดทำบัตร มีแอปพลิเคชันหลายประเภทที่สามารถนำเทคโนโลยีการยืนยันลายนิ้วมือของ ID-WORKS Integrator มาใช้งานได้เหมาะสม ยกตัวอย่างเช่น

- 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

ชุดพัฒนา ID-WORKS Integrator ได้ถูกออกแบบมาให้ทำงานบน Windows Platform โดยชุดที่จำหน่ายทั่วไปจะมาพร้อมกับเครื่องอ่าน DigitalPersona U.are.U 4000 – 4000B และจะทำงานกับเครื่องอ่าน U.are.U 4000 เท่านั้น อย่างไรก็ตามชุด SDK นี้ อาจถูกปรับปรุงให้ทำงานร่วมกับเครื่องอ่านลายนิ้วมือจากผู้ผลิตอื่นๆ ได้ในกรณีที่ต้องการ

ID-WORKS Integrator มีความสามารถเหนือกว่า SDK ของผู้ผลิตเครื่องอ่านลายนิ้วมือยี่ห้อต่างๆ เช่น ชุดพัฒนา DigitalPersona's SDK-DLL Gold Version และ DigitalPersona's SDK-DCOM Platinum โดย ID-WORKS Integrator รองรับการค้นหาลายนิ้วมือแบบ Optimized 1:N (Identification) (มีความเร็วในการค้นหาเฉลี่ยมากกว่า 2,000 ลายนิ้วมือ/วินาที (P4, 1.8 GHz) เทียบกับความเร็วประมาณ 100 นิ้ว/วินาทีที่ได้จากการนำ 1:1 verify module จ ำ ก SDK ของผู้ผลิตเครื่องอ่านลายนิ้วมือมาทำการค้นหา) ในกรณีทั่วไปนักพัฒนาจึงสามารถพัฒนาโปรแกรมค้นหาลายนิ้วมือให้ใช้งานได้โดยมีประสิทธิภาพโดยไม่จำเป็นต้องมีการใช้งานร่วมกับรหัส ID ใดๆ

ชุดพัฒนา ID-WORKS Integrator ใช้ได้กับภาษาที่สนับสนุนการเรียก DLL เช่น C/C++, Delphi, VisualBasic ฯลฯ โดยจะประกอบด้วย ไลบรารีสำหรับประมวลผลลายนิ้วมือในรูปของ DLL และ คู่มืออธิบายการใช้งาน และ Code ตัวอย่างสำหรับการ implement เพื่อใช้งานไลบรารี

ข้อมูลลายนิ้วมือที่ได้จากชุด SDK จะได้มาจากการดึงข้อมูลรายละเอียดต่างๆ ออกมาจากรูปลายนิ้วมือ ซึ่งเมื่อจัดเก็บจะอยู่ในรูปข้อมูล binary ซึ่งมีขนาดประมาณ 400-650 bytes และสามารถนำไปจัดเก็บตามที่ผู้พัฒนาต้องการได้ ซึ่งอาจจะจัดเก็บในฐานข้อมูล ในไฟล์ หรือในบัตรสมาร์ทการ์ด



ความต้องการระบบ

ความต้องการต่ำสุด

- เครื่อง PC แบบ Pentium III หรือดีกว่า
- หน่วยประมวลผลความเร็ว 800 MHz หรือเร็วกว่า
- พื้นที่ว่างในฮาร์ดดิสก์ อย่างน้อย 20 MB
- หน่วยความจำอย่างต่ำ 64 MB
- มีช่อง USB Port ว่างอย่างน้อย 1 ช่อง
- DigitalPersona U.are.U 4000
- DigitalPersona U.are.U Integrator Gold (จะมาพร้อมกับชุด IDWORKS Integrator SDK)

ความต้องการของระบบที่แนะนำ

- เครื่อง PC แบบ Pentium 4 หรือดีกว่า
- หน่วยประมวลผลความเร็ว 1.6 GHz หรือเร็วกว่า
- พื้นที่ว่างในฮาร์ดดิสก์ อย่างน้อย 500 MB
- หน่วยความจำอย่างต่ำ 256 MB
- มีช่อง USB Port ว่างอย่างน้อย 1 ช่อง
- DigitalPersona U.are.U 4000
- DigitalPersona U.are.U Integrator Gold (จะมาพร้อมกับชุด IDWORKS Integrator SDK)



IDWORKS Integrator API

สรุปฟังก์ชันทั้งหมดใน IDWORKS Integrator - ส่วนการประมวลผลลายนิ้วมือ

การเริ่มและเลิกใช้งานไลบรารี

ชื่อฟังก์ชัน	คำอธิบาย
<code>idi_InitModule</code>	เริ่มการทำงานของไลบรารี
<code>idi_FinalizeModule</code>	จบการทำงานของไลบรารี

การสร้างและปิด context และการ Activate

ชื่อฟังก์ชัน	คำอธิบาย
<code>idi_CreateContext</code>	สร้าง IDWORKS processor context
<code>idi_CloseContext</code>	ปิด IDWORKS processor context
<code>idi_ActivateSensor</code>	ลงทะเบียนเครื่องอ่านลายนิ้วมือ ก่อนเริ่มการใช้งาน

การควบคุมเครื่องอ่านลายนิ้วมือ

ชื่อฟังก์ชัน	คำอธิบาย
<code>idi_StartCapturing</code>	เริ่มการรอรับการแตะนิ้วมือ
<code>idi_StopCapturing</code>	หยุดรอรับการแตะนิ้วมือ
<code>idi_RegisterIdentEventProc</code>	register callback function สำหรับเรียกกลับ เพื่อแจ้งผลการค้นหาลายนิ้วมือกับ application
<code>idi_UnregisterIdentEventProc</code>	unregister callback function ที่ register ไว้โดยฟังก์ชัน <code>idi_RegisterIdentEventProc</code>

การเพิ่มและลบลายนิ้วมือ

ชื่อฟังก์ชัน	คำอธิบาย
<code>idi_fp_Add</code>	เพิ่มลายนิ้วมือเข้าไปในฐานข้อมูลชั่วคราวใน context
<code>idi_fp_ClearAll</code>	ลบลายนิ้วมือทั้งหมดในฐานข้อมูลชั่วคราวใน context

สรุปฟังก์ชันทั้งหมดใน IDWORKS Integrator - ส่วนการลงทะเบียนลายนิ้วมือ

การเริ่มและเลิกใช้งานไลบรารี

ชื่อฟังก์ชัน	คำอธิบาย
<code>fpe_InitModule</code>	เริ่มการทำงานของไลบรารี
<code>fpe_FinalizeModule</code>	จบการทำงานของไลบรารี

การสร้างและปิด context และการ Activate

ชื่อฟังก์ชัน	คำอธิบาย
<code>fpe_CreateContext</code>	สร้าง context สำหรับการลงทะเบียนลายนิ้วมือ
<code>fpe_CloseContext</code>	ปิด context สำหรับการลงทะเบียนลายนิ้วมือ
<code>fpe_ActivateSensor</code>	ลงทะเบียนเครื่องอ่านลายนิ้วมือ ก่อนเริ่มการใช้งาน



การลงทะเบียนลายนิ้วมือ

ชื่อฟังก์ชัน	คำอธิบาย
fpe_StartEnrolment	เริ่มการลงทะเบียนลายนิ้วมือ

สรุปฟังก์ชันทั้งหมดใน IDWORKS Integrator – ส่วนข้อมูลเครื่องอ่านลายนิ้วมือ

การเริ่มและเลิกใช้งานไลบรารี

ชื่อฟังก์ชัน	คำอธิบาย
isl_InitModule	เริ่มการทำงานของไลบรารี
isl_FinalizeModule	จบการทำงานของไลบรารี
isl_IsModuleInited	ตรวจสอบว่าได้ทำการ Initialize แล้วหรือไม่

การดึงข้อมูลเครื่องอ่านลายนิ้วมือ

ชื่อฟังก์ชัน	คำอธิบาย
isl_GetDeviceCount	ใช้นับจำนวนเครื่องอ่านลายนิ้วมือในระบบ
isl_GetDeviceList	ดึงข้อมูลเครื่องอ่านลายนิ้วมือทั้งหมดในระบบ



IDWORKS Integrator – ส่วนการประมวลผลลายนิ้วมือ

DLL Name : tmwidw.dll

ทุกฟังก์ชันจะ return ค่าเป็น IDI_R_OK หากทำงานได้สำเร็จ ยกเว้นหากได้หมายเหตุไว้ว่าเป็นอย่างอื่น

รายละเอียดฟังก์ชัน (Function Definitions)

idi_InitModule

IDWORKS Integrator Module: tmwitw.dll

ก่อนการใช้งานโมดูล จะต้องเรียกฟังก์ชันนี้ก่อนที่จะมีการใช้งานฟังก์ชันอื่นๆ เสมอ

Function Definition

C/C++-style function definition:

```
typedef void (__stdcall *idi_InitModule)(void);
```

Pascal-style function definition:

```
idi_InitModule: procedure; stdcall;
```

Visual Basic-style function definition:

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

Parameters

ไม่มี parameter

Return Value

ไม่มี return value

idi_FinalizeModule

IDWORKS Integrator Module: tmwitw.dll

เมื่อเลิกใช้งานโมดูล ต้องเรียกฟังก์ชันนี้เพื่อเคลียร์ข้อมูลต่างๆ ที่อาจค้างอยู่ในโมดูล

Function Definition

C/C++-style function definition:

```
typedef void (__stdcall *idi_FinalizeModule)(void);
```

Pascal-style function definition:

```
idi_FinalizeModule: procedure; stdcall;
```

Visual Basic-style function definition:

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



Parameters

ไม่มี parameter

Return Value

ไม่มี return value

idi_CreateContext

IDWORKS Integrator Module: tmwitw.dll

Context เป็นเสมือนตัวในการเข้าใช้ฟังก์ชัน หลังจากได้ตัวหรือ context แล้ว เมื่อต้องการเรียกใช้ฟังก์ชันอื่นๆ ต้องส่งตัวหรือ context นี้ไปเป็น parameter ในการเรียกฟังก์ชันเสมอ

Function Definition

C/C++-style function definition:

```
typedef IDI_RCODE (__stdcall *idi_CreateContext)(
    IDI_CONTEXT * AProcessorContext
);
```

Pascal-style function definition:

```
idi_CreateContext: function(
    var 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

ถ้าการสร้าง context สำเร็จ AProcessorContext จะ return ค่า Context ออกมา

Return Value

ค่าที่ส่งกลับมากจะเป็น IDI_RCODE โดยมีความหมายดังนี้

- IDI_R_OK : การทำงานเสร็จสมบูรณ์
- IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
- IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_CloseContext

IDWORKS Integrator Module: tmwitw.dll

ฟังก์ชัน **idi_CloseContext** ใช้สำหรับปิดการทำงานของ context



Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

```
idi_CloseContext: function(
    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 ที่ต้องการเลิกการใช้งาน

Return Value

- ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้
- IDI_R_OK : การทำงานเสร็จสมบูรณ์
 - IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
 - IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_ActivateSensor

IDWORKS Integrator Module: tmwitw.dll

ฟังก์ชันนี้ใช้สำหรับเริ่มการใช้งาน Sensor โดยการส่งค่า parameter ของ sensor ที่ต้องการใช้งานเข้าไป

Function Definition

C/C++-style function definition:

```
typedef IDI_RCODE (__stdcall *idi_ActivateSensor)(
    IDI_CONTEXT AProcessorContext,
    IDI_SENSORTYPE ASensorType,
    char *ASensorSerialNo,
    char *AAppSerial,
    char *AAppKey
);
```



Pascal-style function definition:

```
idi_ActivateSensor: function(
    AProcessorContext: IDI_CONTEXT;
    ASensorType: IDI_SENSORTYPE;
    ASensorSerialNo: PChar;
    AAppSerial: PChar;
    AAppKey: PChar
): IDI_RCODE; stdcall;
```

Visual Basic-style function definition:

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

Parameters

AProcessorContext

IDWORKS Processor Context

ASensorType

ประเภทของ Sensor

(สามารถดูได้จากค่าคงที่ที่ขึ้นต้นด้วย IDI_ST_)

ASensorSerialNo

เลข Serial ของ Sensor (null-terminated)

AAppSerial

เลข IDWORKS Integrator Serial (null-terminated)

AAppKey

เลข IDWORKS Integrator Key (null-terminated)

Return Value

ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้

- IDI_R_OK : การทำงานเสร็จสมบูรณ์
- IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
- IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_DeactivateSensor

IDWORKS Integrator Module: tmwitw.dll

ใช้สำหรับยกเลิกการใช้งานหมายเลขเครื่องอ่านลายนิ้วมือ ตามหมายเลขเครื่องอ่านลายนิ้วมือที่กำหนด



Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

```
idi_DeactivateSensor: function(
    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

ประเภทของ Sensor
(สามารถดูได้จากค่าคงที่ที่ขึ้นต้นด้วย IDI_ST_)

ASensorSerialNo

เลข Serial ของ Sensor (null-terminated)

Return Value

ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้

- IDI_R_OK : การทำงานเสร็จสมบูรณ์
- IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
- IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_StartCapturing

IDWORKS Integrator Module: tmwitw.dll

เมื่อเรียกฟังก์ชันนี้ โปรแกรมจะเริ่มการทำงานของ sensor เพื่อรับลายนิ้วมือจากผู้ใช้ หากมีการแตะลายนิ้วมือ ลายนิ้วมือจะถูกประมวลผลโดยอัตโนมัติ ผลการประมวลผลของลายนิ้วมือ (เช่น "รู้จักลายนิ้วมือ" หรือ "ไม่รู้จัก") จะถูกส่งในรูปแบบของ Message ผ่านไปทาง callback function ที่ได้ register ไว้โดยฟังก์ชัน `idi_RegisterIdentEventProc`



Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

```
idi_StartCapturing : function(
    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

หมายเลข Sensor ที่ต้องการให้เริ่มการทำงาน

หมายเหตุ : Sensor ที่สามารถใช้ได้ จะต้องผ่านการ Activate โดยใช้

ฟังก์ชัน `idi_ActivateSensor` ก่อน

Return Value

ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้

- IDI_R_OK : การทำงานเสร็จสมบูรณ์
- IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
- IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_StopCapturing

IDWORKS Integrator Module: tmwitw.dll

ฟังก์ชันนี้จะหยุดการทำงานของ Sensor ตัวที่ระบุ

Function Definition

C/C++-style function definition:

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



Pascal-style function definition:

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

Visual Basic-style function definition:

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

Parameters

AProcessorContext

IDWORKS Processor Context

ASensorSerialNo

หมายเลข Sensor ที่ต้องการหยุดการทำงาน

Return Value

ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้

- IDI_R_OK : การทำงานเสร็จสมบูรณ์
- IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
- IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_RegisterSensorEventProc

IDWORKS Integrator Module: tmwitw.dll

ทำการรีจิสเตอร์ callback function เพื่อรับ event ของเครื่องอ่านลายนิ้วมือ

Function Definition

C/C++-style function definition:

```
typedef 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 หรือ Address ของ Callback function ที่จะถูกเรียกเมื่อมี event เกี่ยวกับเครื่องอ่านลายนิ้วมือ

ARefParam

เป็นค่าที่จะถูกส่งกลับมากับ Message ใน IDI_SensorEventMsg.RefParam (มีประโยชน์เพื่อใช้อ้างอิง เช่น สามารถตั้งค่าเป็น Pointer ของ object ที่ต้องการได้)

Return Value

ค่าที่ส่งกลับมากจะเป็น IDI_RCODE โดยมีความหมายดังนี้

- IDI_R_OK : การทำงานเสร็จสมบูรณ์
- IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
- IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_UnregisterSensorEventProc

IDWORKS Integrator Module: tmwitw.dll

ยกเลิกการรีจิสเตอร์ callback function ที่ระบุ

Function Definition

C/C++-style function definition:

```
typedef 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 ที่ต้องการยกเลิกการรับ event

Return Value

ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้

- IDI_R_OK : การทำงานเสร็จสมบูรณ์
- IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
- IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_RegisterIdentEventProc

IDWORKS Integrator Module: tmwitw.dll

Register ฟังก์ชัน callback สำหรับรับ event การประมวลผลลายนิ้วมือ

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

```
idi_RegisterIdentEventProc: function(
    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, _
        ByVal ARefParam As Long _
    ) As Long
```



Parameters

AProcessorContext

IDWORKS Processor Context

AEventProc

Callback function ที่จะถูกเรียกเมื่อมีการประมวลผลลายนี้นี้

ARefParam

เป็นค่าที่จะถูกส่งกลับมากับ Message ใน IDI_IdentEventMsg.RefParam (มีประโยชน์เพื่อใช้อ้างอิง เช่น สามารถตั้งค่าเป็น Pointer ของ object ที่ต้องการได้)

Return Value

ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้

- IDI_R_OK : การทำงานเสร็จสมบูรณ์
- IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
- IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_UnregisterIdentEventProc

IDWORKS Integrator Module: tmwitw.dll

ยกเลิกการ Register ที่ได้ทำไปแล้วจากการเรียกฟังก์ชัน `idi_RegisterIdentEventProc`

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

```
idi_UnregisterIdentEventProc: function(
    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 ที่ต้องการยกเลิกการ register



Return Value

- ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้
- IDI_R_OK : การทำงานเสร็จสมบูรณ์
 - IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
 - IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง

idi_fp_Add

IDWORKS Integrator Module: tmwitw.dll

เพิ่มลายนิ้วมือเข้าไปในฐานข้อมูลชั่วคราวภายในของโมดูลลายนิ้วมือที่เพิ่มเข้าไปจะทำให้โมดูลรู้จักลายนิ้วมือนั้นเมื่อมีการประมวลผลลายนิ้วมือ

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

```
idi_fp_Add: function(
    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 ที่ชี้ไปยังข้อมูลลายนิ้วมือ (FP_FPInfo) ที่ต้องการเพิ่มเข้าไปในฐานข้อมูลภายใน

Return Value

- ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้
- IDI_R_OK : การทำงานเสร็จสมบูรณ์
 - IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
 - IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง



idi_fp_ClearAll

IDWORKS Integrator Module: tmwitw.dll

เป็นการล้างฐานข้อมูลภายในของโมดูล

Function Definition

C/C++-style function definition:

```
typedef IDI_RCODE (__stdcall *idi_fp_ClearAll)(
    IDI_CONTEXT processorContext
);
```

Pascal-style function definition:

```
idi_fp_ClearAll : function(
    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

- ค่าที่ส่งกลับมาจะเป็น IDI_RCODE โดยมีความหมายดังนี้
- IDI_R_OK : การทำงานเสร็จสมบูรณ์
 - IDI_R_MODULENOTINITED : ยังไม่มีการ Initialize Module
 - IDI_R_INVALIDCONTEXT : ค่า Context ไม่ถูกต้อง



ประเภทข้อมูล (Data Types)

IDI_RCODE

IDI_RCODE เป็นประเภทข้อมูล 32 บิต ที่เป็นค่าที่ส่งกลับมาเมื่อเรียกฟังก์ชัน เพื่อบอกถึงผลลัพธ์ของฟังก์ชันนั้นๆ

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long IDI_RCODE;
```

Pascal-style data type definition:

```
IDI_RCODE = Integer;
```

Visual Basic-style data type definition:

ใช้ประเภท long

IDI_CONTEXT

IDI_CONTEXT เป็นประเภทข้อมูล 32-bit สำหรับ Processor Context ที่เหมือนกับเป็นหน่วยในการประมวลผลข้อมูล ซึ่งสามารถสร้างได้หลาย context แยกกัน ในการใช้งานแต่ละครั้ง

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long IDI_CONTEXT;
```

Pascal-style data type definition:

```
IDI_CONTEXT = Longword;
```

Visual Basic-style data type definition:

ใช้ประเภท Long

IDI_IDENTEVENTMSG

IDI_IDENTEVENTMSG เป็นประเภทข้อมูลแบบ structure ที่จะถูกส่งกลับมาให้กับ callback function ที่ได้ทำการ register ไว้โดยฟังก์ชัน `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;
IDI_IDENTEVENTMSG_PTR = ^IDI_IDENTEVENTMSG;
```

Visual Basic-style data type definition:

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

หมายเหตุ

ค่าที่ถูกส่งมาใน MsgParam จะขึ้นกับค่า MsgID ซึ่งจะมีความหมายดังตารางต่อไปนี้

ค่า MsgID	ความหมายของค่าใน MsgParam
IDI_ID_FINGERPRINTMATCHED	Pointer ที่ชี้ไปยัง IDI_IDENTEVENTMATCHSTRUCT
IDI_ID_FINGERPRINTNOTMATCHED	0

IDI_IDENTEVENTTYPE

IDI_IDENTEVENTTYPE เป็นประเภทข้อมูลขนาด 32-bit สำหรับเก็บชนิดของ Identification Message ที่จะถูกส่งกลับมายังฟังก์ชันที่ได้ทำการ register ไว้โดยใช้ฟังก์ชัน RegisterIdentEventProc ซึ่งประเภทข้อมูลนี้จะใช้ใน IDI_IDENTEVENTMSG.MsgID

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long IDI_IDENTEVENTTYPE;
```

Pascal-style data type definition:

```
IDI_IDENTEVENTTYPE = Longword;
```

Visual Basic-style data type definition:

ใช้ประเภท Long

IDI_IDENTEVENTPARAM

IDI_IDENTEVENTPARAM เป็นประเภทข้อมูลขนาด 32-bit ที่จะถูกส่งมากับ Identification Message IDI_IDENTEVENTMSG ซึ่งความหมายจะแตกต่างกันสำหรับแต่ละ message ขึ้นอยู่กับค่า IDI_IDENTEVENTMSG.MsgID



Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long IDI_IDENTEVENTPARAM;
```

Pascal-style data type definition:

```
IDI_IDENTEVENTPARAM = Longword;
```

Visual Basic-style data type definition:

```
ใช้ประเภท Long
```

IDI_REFPARAM

IDI_REFPARAM เป็นข้อมูล Pointer ที่ไว้ใช้อ้างอิงกับการเรียกใช้ฟังก์ชัน `idi_RegisterIdentEventProc` ได้ โดยค่าที่ส่งกลับมาจะเป็นค่าเดียวกันกับค่า `ARefParam` ที่ส่งเข้าไปเมื่อเรียกฟังก์ชัน `idi_RegisterIdentEventProc`

Data-type Definition

C/C++-style data type definition:

```
typedef void *IDI_REFPARAM;
```

Pascal-style data type definition:

```
IDI_REFPARAM = Pointer;
```

Visual Basic-style data type definition:

```
ใช้ประเภท Long
```

IDI_IDENTEVENTPROC

IDI_IDENTEVENTPROC เป็นการประกาศประเภทของฟังก์ชัน ที่ใช้สำหรับการ callback เมื่อมีการแตะนิ้วมือ โดยpointer ของฟังก์ชันที่ใช้ส่งเข้าไปเป็นพารามิเตอร์ `AEventProc` ของ `idi_RegisterIdentEventProc` จะต้องมี arguments เหมือนกับที่ประกาศใน 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_IDENTEVENTMSG_PTR
);stdcall;
```

Visual Basic-style data type definition:

```
Public Sub IdentEventProc(
    ByRef audtIdentEventMsg As IDI_IDENTEVENTMSG
)
```



Parameters

AldentEventMsgPtr

Pointer ที่ชี้ไปยัง IDI_IDENTEVENTMSG ซึ่งบอกรายละเอียดของ Message

IDI_IDENTEVENTMATCHEDSTRUCT

ในกรณีที่มีการแตะนิ้วมือและพบข้อมูล IDI_IDENTEVENTMATCHEDSTRUCT จะเป็นข้อมูลที่ถูกส่งกลับมาจาก IDI_IDENTEVENTMSG ซึ่ง IDI_IDENTEVENTMSG.MsgParam จะเป็น pointer ที่ชี้ไปยังข้อมูลดังกล่าว

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_FPINFO_PTR;
    MatchingScore : Integer;
end;

IDI_IDENTEVENTMATCHEDSTRUCT_PTR = ^IDI_IDENTEVENTMATCHEDSTRUCT;
```

Visual Basic-style data type definition:

```
Public Type IDI_IDENTEVENTMATCHEDSTRUCT
    FPInfoPtr As Long
    MatchingScore As Long
End Type
```

Members

FPInfoPtr

เป็น Pointer ที่ชี้ไปยังข้อมูล FP_FPINFO ที่บอกรายละเอียดของลายนิ้วมือ

MatchingScore

เป็นค่าคะแนนความเหมือนกันของลายนิ้วมือ

IDI_SENSOREVENTMSG

IDI_SENSOREVENTMSG เป็นประเภทข้อมูลแบบ structure ที่จะถูกส่งกลับมาให้กับ callback function ที่ได้ทำการ register ไว้โดยฟังก์ชัน `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;
IDI_SENSOREVENTMSG_PTR = ^IDI_SENSOREVENTMSG;

```

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 ที่เป็นผู้ส่ง message มาให้

MsgID
ประเภทของ Message

MsgParam
ข้อมูลของ Message นั้นๆ

RefParam
เลขอ้างอิงที่ถูกส่งเข้ามากับการเรียก `idi_RegisterSensorEventProc` ก่อนหน้านี้

หมายเหตุ

ค่าที่ถูกส่งมาใน `MsgParam` จะขึ้นกับค่า `MsgID` ซึ่งจะมีความหมายดังตารางต่อไปนี้

ค่า MsgID	ความหมายของค่าใน MsgParam
IDI_SE_IMAGEACQUIRED	Pointer ที่ชี้ไปยัง IDI_SENSORIMAGEACQUIREDSTRUCT
IDI_SE_IMAGEPROCESSED	Pointer ที่ชี้ไปยัง IDI_SENSORIMAGEACQUIREDSTRUCT

IDI_SENSOREVENTPROC

`IDI_SENSOREVENTPROC` เป็นการประกาศประเภทของฟังก์ชัน ที่ใช้สำหรับการ callback เมื่อมี event เกี่ยวกับเครื่องอ่านลายนิ้วมือ เช่น ได้รับรูปลายนิ้วมือ ประเภทฟังก์ชันนี้จะใช้กับฟังก์ชัน `IDI_RegisterSensorEventProc` โดย pointer หรือ address ของฟังก์ชันที่ใช้ส่งเข้าไปเป็นพารามิเตอร์ `AEventProc` จะต้องมี arguments เหมือนกับที่ประกาศใน `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_SENSOREVENTMSG_PTR;
); stdcall;
```

Visual Basic-style data type definition:

```
Public Sub SensorEventProc(ByRef ASensorEventMsg As IDI_SENSOREVENTMSG)
```

Parameters

ASensorEventMsgPtr

Pointer ที่ชี้ไปยัง IDI_SENSOREVENTMSG ซึ่งบอกรายละเอียดของ Message

IDI_SENSOREVENTIMAGEACQUIREDSTRUCT

เมื่อมีการแตะลายนิ้วมือ และเครื่องอ่านลายนิ้วมืออ่านรูปลายนิ้วมือสำเร็จ โมดูลจะส่ง event ให้กับ function ที่ได้ทำการรีจิสเตอร์ไว้โดย `idi_RegisterSensorEvent` ซึ่งใน `MsgParam` ของ event ดังกล่าว จะเป็นค่า pointer ของ `IDI_SENSOREVENTIMAGEACQUIREDSTRUCT`

Data-type Definition

C/C++-style data type definition:

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

Pascal-style data type definition:

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

IDI_SENSOREVENTIMAGEACQUIREDSTRUCT_PTR = ^IDI_SENSOREVENTIMAGEACQUIREDSTRUCT;
```

Visual Basic-style data type definition:

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

Members

ImageWidth

ความกว้างของรูปลายนิ้วมือ

ImageHeight

ความสูงของรูปลายนิ้วมือ



ImageBitmapPtr

Pointer ที่ชี้ไปยังข้อมูล grayscale bitmap ของลายนิ้วมือ โดยมีขนาดเท่ากับ ImageWidth x ImageHeight และมีรูปแบบเป็น 8-bit grayscale

FP_FPINFO

FP_FPINFO เป็นประเภทข้อมูลแบบ structure ที่ใช้สำหรับส่งผ่านข้อมูลลายนิ้วมือเข้าไปยังฐานข้อมูลชั่วคราวใน Processor Context

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: Longword;
    Tag : Integer;
end;

FP_FPINFO_PTR = ^FP_FPINFO;
```

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

เป็นค่า Indexing ของลายนิ้วมือ ที่ใช้สำหรับ optimize การค้นหาลายนิ้วมือ

FingerIndex

เป็นหมายเลขบอกนิ้ว ซึ่งจะไล่ตั้งแต่ 0 คือนิ้วก้อยมือซ้าย ถึง 9 คือนิ้วก้อยมือขวา

FingerprintRawDataPtr

เป็น Pointer ที่ชี้ไปยัง memory buffer ที่เก็บข้อมูลลายนิ้วมือ

FingerprintRawDataSize

เป็นขนาดของ memory buffer ที่ชี้โดย FingerprintRawDataPtr

Tag

เป็นเลขกำกับลายนิ้วมือ ซึ่งผู้พัฒนาสามารถใช้อ้างอิงได้ เช่น สามารถเก็บเลข ID ของบุคคลในฐานข้อมูล





ค่าคงที่ (Constants)

ค่าคงที่สำหรับ IDI_RCODE

ชื่อค่าคงที่	ค่า	ความหมาย
IDI_R_OK	0	การทำงานเสร็จสิ้น ไม่มีข้อผิดพลาด
IDI_R_GENERALFAILURE	-1	มีข้อผิดพลาดทั่วไป
IDI_R_INIT_FAILED	-2	ไม่สามารถเริ่มการใช้งานได้
IDI_R_MODULENOTINITED	-3	เรียกฟังก์ชันโดยไม่มี การ Initialize Module ก่อน
IDI_R_SENSORNOTFOUND	-4	ไม่พบเครื่องอ่านลายนิ้วมือตามที่กำหนด
IDI_R_SENSORALREADYACTIVATED	-5	เครื่องอ่านลายนิ้วมือได้ถูก Activate ไปก่อนหน้านั้นแล้ว
IDI_R_INVALIDSENSORTYPE	-6	ประเภทเครื่องอ่านลายนิ้วมือไม่ถูกต้อง
IDI_R_INVALIDREGISTRATION	-7	การ Activate ไม่ถูกต้อง
IDI_R_INVALIDCONTEXT	-16	Context ไม่ถูกต้อง

ค่าคงที่สำหรับ IDI_IDENTEVENTTYPE

ชื่อค่าคงที่	ค่า	ความหมาย
IDI_ID_FINGERPRINTMATCHED	0	พบลายนิ้วมือ
IDI_ID_FINGERPRINTNOTMATCHED	1	ไม่พบลายนิ้วมือ

ค่าคงที่สำหรับ IDI_SENSORTYPE

ชื่อค่าคงที่	ค่า	ความหมาย
IDI_ST_UAREU	1	เครื่องอ่านลายนิ้วมือ DigitalPersona U.are.U 4000

ค่าคงที่สำหรับ IDI_SENSOREVENTTYPE

ชื่อค่าคงที่	ค่า	ความหมาย
IDI_SE_IMAGEACQUIRED	1	Event รับรูปจากเครื่องอ่านลายนิ้วมือ
IDI_SE_IMAGEPROCESSED	2	Event ประมวลผลรูปลายนิ้วมือเสร็จสิ้น



ID-WORKS Integrator – ส่วนข้อมูลเครื่องอ่านลายนิ้วมือ

DLL Name: ITWSSL.DLL

เป็นไลบรารีสำหรับดึงข้อมูลของเครื่องอ่านลายนิ้วมือที่ต่ออยู่ทั้งหมด

รายละเอียดฟังก์ชัน (Function Definitions)

isl_IsModuleInited

IDWORKS Integrator Module: ITWSSL.DLL

ตรวจสอบว่า module ได้ถูก initialize แล้วหรือไม่

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

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

Visual Basic-style function definition:

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

Parameters

ไม่มี parameters

Return Value

- ISL_R_OK : โมดูลได้ initialize แล้ว
- ISL_R_MODULENOTINITED : โมดูลยังไม่ได้ initialize

isl_InitModule

IDWORKS Integrator Module: ITWSSL.DLL

เริ่มการทำงานของโมดูล

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

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

Visual Basic-style function definition:

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



Parameters

ไม่มี parameters

Return Value

- ISL_R_OK : การเริ่มต้น
- ISL_R_MODULENOTINITED : ไม่สามารถเริ่มต้นโมดูลได้

isl_FinalizeModule

IDWORKS Integrator Module: ITWSSL.DLL

เลิกการทำงานของโมดูล เมื่อเลิกการใช้งานโมดูล (เช่น ในช่วงก่อนปิดโปรแกรม)จะต้องเรียกฟังก์ชันนี้เพื่อเคลียร์ข้อมูลต่างๆที่อาจค้างอยู่ในโมดูล

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

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

Visual Basic-style function definition:

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

Parameters

ไม่มี parameters

Return Value

- ISL_R_OK : สำเร็จ
- ISL_R_MODULENOTINITED : ไม่สามารถเริ่มต้นโมดูลได้

isl_GetDeviceCount

IDWORKS Integrator Module: ITWSSL.DLL

ใช้สำหรับนับจำนวน sensor ทั้งหมดที่ต่ออยู่

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

```
isl_GetDeviceCount: function(
    var 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

เป็นจำนวน sensor ที่ส่งกลับมา

Return Value

- ISL_R_OK : สำเร็จ จำนวน sensor จะถูกใส่ค่ามาใน ASensorCount
- ISL_R_MODULENOTINITED : ฟังก์ชัน isl_InitModule ยังไม่ถูกเรียก

isl_GetDeviceList

IDWORKS Integrator Module: ITWSSL.DLL

ใช้สำหรับดึงข้อมูลของ sensor ทั้งหมดที่ต่ออยู่

Function Definition

C/C++-style function definition:

```
typedef ISL_RCODE ( __stdcall *isl_GetDeviceList)(
    ISL_SENSORINFO *ASensorList,
    unsigned int ASensorCount
);
```

Pascal-style function definition:

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

Visual Basic-style function definition:

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

Parameters

ASensorList

เป็น pointer ของ array (แบบ C/C++) ของ FP_FPInfo ซึ่งต้องมีความยาวอย่างน้อยเท่ากับ ASensorCount

ASensorCount

คือจำนวน sensor ที่ต้องการดึงข้อมูล สามารถหาได้จากฟังก์ชัน isl_GetDeviceCount



Return Value

- ISL_R_OK : สำเร็จ จำนวน sensor จะถูกใส่ค่ามาใน ASensorCount
- ISL_R_MODULENOTINITED : ฟังก์ชัน isl_InitModule ยังไม่ถูกเรียก



ประเภทข้อมูล (Data Types)

ISL_RCODE

ISL_RCODE เป็นประเภทข้อมูล 32 บิต ที่เป็นค่าที่ส่งกลับมาเมื่อเรียกฟังก์ชัน เพื่อบอกถึงผลลัพธ์ของฟังก์ชันนั้นๆ

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long ISL_RCODE;
```

Pascal-style data type definition:

```
ISL_RCODE = Integer;
```

Visual Basic-style data type definition:

ใช้ประเภท Long

ISL_DEVICETYPE

ISL_DEVICETYPE เป็นประเภทข้อมูล 32 บิตที่ใช้สำหรับระบุประเภทของเครื่องอ่านลายนิ้วมือ

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long ISL_DEVICETYPE;
```

Pascal-style data type definition:

```
ISL_DEVICETYPE = Longword;
```

Visual Basic-style data type definition:

ใช้ประเภท Long

ISL_DEVICEID

ISL_DEVICEID เป็นประเภทข้อมูล 64 บิต ที่ใช้เป็นเลขประจำตัวของเครื่องอ่านลายนิ้วมือ

Data-type Definition

C/C++-style data type definition:

```
typedef __int64 ISL_DEVICEID;
```

Pascal-style data type definition:

```
ISL_DEVICEID = int64;
```

Visual Basic-style data type definition:

ใน Visual Basic จะไม่มี Integer ขนาด 64 bit แต่ใน header file ของ Visual Basic ที่มาพร้อมกับ SDK จะแก้ปัญหาใน structure ที่ใช้ประเภทข้อมูลแบบนี้แล้ว



ISL_CHAR

ISL_CHAR เป็นประเภทข้อมูล 8 บิต ที่ใช้เก็บตัวอักษร จะใช้เก็บข้อความทั่วไปในโมดูล

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:

ใช้ประเภท Byte

ISL_SENSORINFO

ISL_SENSORINFO เป็นประเภทข้อมูลแบบ structure ที่ใช้เก็บข้อมูลเกี่ยวกับรายละเอียดต่างๆ ของเครื่องอ่านลายนิ้วมือ

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;

ISL_SENSORINFO_PTR = ^ISL_SENSORINFO
```

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

ประเภทของเครื่องอ่านลายนิ้วมือ

DeviceID

หมายเลขประจำเครื่องอ่านลายนิ้วมือ



DeviceName

ชื่อเครื่องอ่านลายนิ้วมือ เช่น DigitalPersona U.are.U 4000

DeviceSerial

S/N ประจำเครื่องอ่านลายนิ้วมือ



ค่าคงที่ (Constants)

ค่าคงที่ทั่วไป

ชื่อค่าคงที่	ค่า	ความหมาย
ISL_MAXDEVICENAMELEN	64	เรียกฟังก์ชันโดยไม่มี Initialize Module ก่อน
ISL_MAXDEVICESSERIALLEN	64	ไม่พบเครื่องอ่านลายนิ้วมือตามที่กำหนด

ค่าคงที่สำหรับ ISL_RCODE

ชื่อค่าคงที่	ค่า	ความหมาย
ISL_R_OK	0	การทำงานเสร็จสิ้น ไม่มีข้อผิดพลาด
ISL_R_MODULENOTINITED	-1	มีข้อผิดพลาดทั่วไป
ISL_R_ALREADYINITED	-2	ไม่สามารถเริ่มการใช้งานได้

ค่าคงที่สำหรับ ISL_DEVICETYPE

ชื่อค่าคงที่	ค่า	ความหมาย
ISL_DT_UAREU	0	การทำงานเสร็จสิ้น ไม่มีข้อผิดพลาด



ID-WORKS Integrator – ส่วนช่วยการบันทึกถายนิ้วมือ

DLL Name: FPENROL.DLL

- ทุกฟังก์ชันจะ return ค่าเป็น FPE_R_OK หากทำงานได้สำเร็จ ยกเว้นหากได้หมายเหตุไว้ว่าเป็นอย่างอื่น

รายละเอียดฟังก์ชัน (Function Definitions)

fpe_InitModule

IDWORKS Integrator Module: FPENROL.DLL

เริ่มการใช้งานไลบรารี

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

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

Visual Basic-style function definition:

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

Parameters

ไม่มี Parameter

Return Value

- FPE_R_OK : การทำงานสำเร็จ
- FPE_R_MODULENOTINITED : ไม่สามารถเริ่มการทำงานได้

fpe_FinalizeModule

IDWORKS Integrator Module: FPENROL.DLL

เรียกเมื่อเลิกใช้งานไลบรารี

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

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

Visual Basic-style function definition:

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



Parameters

ไม่มี Parameter

Return Value

- FPE_R_OK : การทำงานสำเร็จ
- FPE_R_MODULENOTINITED : ไม่ได้มีการ Initialize Module ก่อนหน้านี้

fpe_CreateContext

IDWORKS Integrator Module: FPENROL.DLL

สร้าง context สำหรับ fingerprint enrolment

Function Definition

C/C++-style function definition:

```
typedef FPE_RCODE (__stdcall *LP_FUNC_FPE_CREATE_CONTEXT)(
    FPE_CONTEXT* AEnrolmentContext
);
```

Pascal-style function definition:

```
fpe_CreateContext: function(
    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

ถ้าการสร้าง Context สำเร็จ ค่าดังกล่าวจะเป็น Context สำหรับ Enrolment

Return Value

- FPE_R_OK : การทำงานสำเร็จ
- FPE_R_MODULENOTINITED : ไม่ได้มีการ Initialize Module ก่อนหน้านี้

fpe_CloseContext

IDWORKS Integrator Module: FPENROL.DLL

ปิด context ของ fingerprint enrolment

Function Definition

C/C++-style function definition:

```
typedef FPE_RCODE (__stdcall *fpe_CloseContext)(
```




```
FPE_CONTEXT AEnrolmentContext
);
```

Pascal-style function definition:

```
fpe_CloseContext: function(
    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

หากการสร้าง context สำเร็จ AEnrolmentContext จะถูกตั้งค่าเป็นค่า context ที่ใช้ในการลงทะเบียนลายนิ้วมือ

Return Value

- FPE_R_OK : การทำงานสำเร็จ
- FPE_R_MODULENOTINITED : ไม่ได้มีการ Initialize Module ก่อนหน้านี้

fpe_ActivateSensor

IDWORKS Integrator Module: FPENROL.DLL

เริ่มการใช้งาน sensor ที่ต้องการ

Function Definition

C/C++-style function definition:

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

Pascal-style function definition:

```
fpe_ActivateSensor: function(
    AEnrolmentContext: FPE_CONTEXT;
    ASensorType: FPE_SENSORTYPE;
    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 ที่ใช้ในการลงทะเบียนลายนิ้วมือ

ASensorType

ประเภทของ Sensor (สามารถดูได้จากค่าคงที่ที่ขึ้นต้นด้วย FPE_ST_)

ASensorSerialNo

เลข Serial ของ Sensor (null-terminated)

AAppSerial

เลข IDWORKS Integrator Serial (null-terminated)

AAppKey

เลข IDWORKS Integrator Key (null-terminated)

Return Value

- FPE_R_OK : การทำงานสำเร็จ และ Activate เครื่องอ่านลายนิ้วมือเรียบร้อยแล้ว
- FPE_R_MODULENOTINITED : ไม่ได้มีการ Initialize Module ก่อนหน้านี้

fpe_StartEnrolment

IDWORKS Integrator Module: FPENROL.DLL

เริ่มการ Enroll ลายนิ้วมือ

Function Definition

C/C++-style function definition:

```

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

```



Pascal-style function definition:

```
fpe_StartEnrolment : function(
    AEnrolmentContext: FPE_CONTEXT;
    AFingerIndex: Integer;
    ARawDataPtr: PByte;
    AMaxRawDataSize: Cardinal;
    var 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 Long, _
    ByRef ARawDataPtr As Any, _
    ByVal AMaxRawDataSize As Long, _
    ByRef AOutputRawDataSize As Long, _
    ByRef AIndexingValue As Long _
) As Long
```

Parameters

AEnrolmentContext

Fingerprint Enrolment Context

AFingerIndex

ขณะ Start Enrolment ให้เลือกนิ้วใด

ARawDataPtr

Pointer ที่ชี้ไปที่ buffer สำหรับเก็บข้อมูล binary ของลายนิ้วมือ

AMaxRawDataSize

ขนาดสูงสุดที่สามารถจัดเก็บข้อมูล binary ลงใน ARawDatPtr ได้

AOutputRawDataSize

จะเป็นขนาดที่ฟังก์ชันส่งออกมาเพื่อบอกขนาดของ buffer ที่ได้ใช้ไป หากเป็น 0 หมายถึง ผู้ใช้กดยกเลิก

AIndexingValue

ถ้าไม่ใช่ null ฟังก์ชันจะส่งค่า FingerprintIndexingValue ออกมาให้ เพื่อนำไปจัดเก็บ

Return Value

- FPE_R_OK : การทำงานสำเร็จ
- FPE_R_MODULENOTINITED : ไม่ได้มีการ Initialize Module ก่อนหน้านี้

หมายเหตุ

ถ้าผู้ใช้กดยกเลิก ค่าที่ return มากี่ยังเป็น FPE_R_OK แต่ AOutputRawDataSize จะมีค่าเป็น 0 ซึ่งหมายถึงการทำงานเสร็จสมบูรณ์ แต่ไม่มีข้อมูลลายนิ้วมือ เนื่องจากผู้ใช้กดยกเลิก



ประเภทข้อมูล (Data Types)

FPE_RCODE

FPE_RCODE เป็นประเภทข้อมูล 32 บิต ที่เป็นค่าที่ส่งกลับมาเมื่อเรียกฟังก์ชัน เพื่อบอกถึงผลลัพธ์ของฟังก์ชันนั้นๆ

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long FPE_RCODE;
```

Pascal-style data type definition:

```
FPE_RCODE = Integer;
```

Visual Basic-style data type definition:

ใช้ประเภท Long

FPE_CONTEXT

FPE_CONTEXT เป็นประเภทข้อมูล 32-bit สำหรับ Context ที่เสมือนกับเป็นหน่วยย่อยในการลงทะเบียนลายนิ้วมือ

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long FPE_CONTEXT;
```

Pascal-style data type definition:

```
FPE_CONTEXT = Longword;
```

Visual Basic-style data type definition:

ใช้ประเภท Long

FPE_SENSORTYPE

FPE_SENSORTYPE เป็นประเภทข้อมูล 32 บิตที่ใช้สำหรับระบุประเภทของเครื่องอ่านลายนิ้วมือ

Data-type Definition

C/C++-style data type definition:

```
typedef unsigned long FPE_SENSORTYPE;
```

Pascal-style data type definition:

```
FPE_SENSORTYPE = Integer;
```

Visual Basic-style data type definition:

ใช้ประเภท Long



ค่าคงที่ (Constants)

ค่าคงที่ทั่วไป

ชื่อค่าคงที่	ค่า	ความหมาย
FPC_MAXRAWDATASIZE	4096	ขนาด memory buffer สูงสุดที่ต้องการในการลงทะเบียนลายนิ้วมือ

ค่าคงที่สำหรับ ISL_RCODE

ชื่อค่าคงที่	ค่า	ความหมาย
FPE_R_OK	0	การทำงานเสร็จสิ้น ไม่มีข้อผิดพลาด
FPE_R_INVALIDCONTEXT	-1	Context ไม่ถูกต้อง
FPE_R_GENERALFAILURE	-2	มีข้อผิดพลาดทั่วไป
FPE_R_MODULENOTINITED	-3	เรียกฟังก์ชันโดยไม่มี Initialize Module
FPE_R_SENSORNOTFOUND	-4	ไม่พบเครื่องอ่านลายนิ้วมือที่ระบุ
FPE_R_INTERNALERROR	-4096	มีข้อผิดพลาดภายใน module

ค่าคงที่สำหรับ ISL_DEVICETYPE

ชื่อค่าคงที่	ค่า	ความหมาย
FPE_ST_UAREU	1	ประเภทของเครื่องอ่านลายนิ้วมือ



Appendix: A – การแก้ไขข้อผิดพลาดระหว่างการติดตั้ง

การแก้ไขข้อผิดพลาดระหว่างการติดตั้ง

มีสาเหตุสองสามประการที่อาจทำให้เกิดปัญหาในการติดตั้งเครื่องอ่านลายนิ้วมือ เช่น

- ในบางครั้งคุณอาจเกิดปัญหาถ้าติดตั้งเครื่องอ่านลายนิ้วมือก่อนติดตั้งไดรเวอร์ หรือ
- คุณอาจเกิดปัญหาในกรณีที่เคยติดตั้งไดรเวอร์รุ่นเก่าๆมาก่อนที่ไม่สามารถทำงานร่วมกันได้ หรือ
- คุณอาจติดตั้งโปรแกรมอื่นๆที่มีไดรเวอร์ที่ไม่สามารถทำงานร่วมกันได้ การอัปเดตเวอร์ชันของ Windows ก็อาจทำให้เกิดปัญหาได้

ปัญหาทั้งหมดที่กล่าวมาสามารถแก้ไขได้โดยวิธีเดียวกันคือ

1. ถอดสายเครื่องอ่านลายนิ้วมือออกจากเครื่องคอมพิวเตอร์ของคุณ
2. ถอดโปรแกรมและไดรเวอร์ทั้งหมดของเครื่องอ่านลายนิ้วมือออกจากเครื่องคอมพิวเตอร์ของคุณ
 - I. เลือกที่ปุ่ม “Start” (มุมซ้ายล่างของจอ)
 - II. เลือก “Setting Menu”
 - III. คลิกที่ “Control Panel”
 - IV. ดับเบิลคลิกที่ “Add Remove Programs”
 - V. เลือกถอดโปรแกรมทั้งหมดที่มีคำว่า “IDWORKS”, “digitalPersona” หรือ “U.are.U” ทั้งหมด.
3. รีสตาร์ทเครื่องคอมพิวเตอร์ของคุณ
4. ติดตั้งโปรแกรม IDWORKS ใหม่ทั้งหมดตั้งแต่เริ่ม รวมทั้งไดรเวอร์เครื่องอ่านลายนิ้วมือ
5. เสียบสายเครื่องอ่านลายนิ้วมืออีกครั้ง

สิ่งอื่นๆที่อาจทำให้เกิดปัญหาอาจอยู่ที่สาย USB หรือ USB Hubs ที่ใช้งาน ถ้าหากว่าคุณสามารถใช้งานเครื่องอ่านลายนิ้วมือ ได้โดยการเสียบสายโดยตรงกับเครื่อง นั้นแสดงว่ามีปัญหาบางอย่างจากการต่อสาย สิ่งที่ต้องสังเกตได้แก่

- สายอาจเสียหายหรือหลวม
- พลังงานไฟฟ้าไม่พอ ในกรณีที่มีการใช้งานอุปกรณ์ USB หลายๆชิ้นบน Hub ตัวเดียวกันอาจทำให้พลังงานไฟฟ้าไม่พอ
- การต่อสายพ่วงที่ยาวมากๆอาจทำให้เครื่องอ่านลายนิ้วมือไม่สามารถทำงานได้



Appendix: B – การใช้งานและการดูแลรักษาเครื่องอ่านลายนิ้วมือ

การใช้งานเครื่องอ่านลายนิ้วมืออย่างถูกวิธี

พยายามแตะให้กว้างครอบคลุมพื้นที่มากที่สุด

ในการแตะอุปกรณ์เพื่อลงทะเบียนหรือลงเวลา ผู้ใช้ควรวางปลายนิ้วไว้ที่ส่วนบนสุดของกระจกรับภาพของอุปกรณ์ และแตะนิ้วลงไปให้สัมผัสกระจกให้ได้ครอบคลุมพื้นที่มากที่สุด เพราะอุปกรณ์จะอ่านลายนิ้วมือจากส่วนที่สัมผัสกับกระจกเท่านั้น ดังนั้นยังผู้ใช้แตะพื้นผิวได้กว้างเท่าใดก็ยิ่งเป็นการดีสำหรับการประมวลผลข้อมูลลายนิ้วมือได้อย่างแม่นยำ

วางนิ้วให้ขนานกับแนวราบ

บ่อยครั้งที่ผู้ใช้อุปกรณ์จะแตะอุปกรณ์ในลักษณะเดียวกับการกดปุ่ม เหมือนกับการชี้นิ้วลงไปแตะอุปกรณ์ ซึ่งเป็นลักษณะการใช้งานที่ไม่ถูกต้อง เพราะในลักษณะนั้น ส่วนปลายนิ้วเท่านั้นที่สัมผัสกับกระจก ซึ่งจะทำให้ลายนิ้วมือที่อ่านได้มีคุณภาพต่ำ (โดยปกติอุปกรณ์จะไม่อ่านลายนิ้วมือจากการสัมผัสในลักษณะนั้น เนื่องจากมีพื้นที่สัมผัสน้อยกว่าที่กำหนด) ซึ่งลักษณะที่ถูกต้องในการสัมผัสอุปกรณ์ จะต้องวางนิ้วขนานกับแนวราบลงไปกับกระจกรับภาพของอุปกรณ์

ไม่ควรกดแรงเกินไป

ลายนิ้วมือที่อุปกรณ์อ่านได้นั้น มีที่มาจากรายละเอียดและความดันลึกบนพื้นผิวของลายนิ้วมือของบุคคลนั้นๆ การกดอุปกรณ์แรงเกินไปจะทำให้ลายนิ้วมือที่อ่านได้ไม่แม่นยำ และถูกต้องน้อยลง ในทางกลับกัน หากแตะเบาเกินไป จะทำให้นิ้วสัมผัสกับอุปกรณ์น้อยกว่าจะอ่านข้อมูลได้ การใช้งานที่ถูกต้องคือ ผู้ใช้จะต้องวางนิ้วลงกับอุปกรณ์ให้พอดี และกดลงไปเบาๆ

เล็บยาว?

บ่อยครั้งที่ผู้ใช้ที่มีเล็บยาว จะมีปัญหากับการใช้เครื่องอ่านลายนิ้วมือ เนื่องจากผู้ใช้มักจะใช้ปลายเล็บวางไว้ที่ส่วนบนสุดของกระจกรับภาพ แทนที่จะเป็นปลายนิ้ว ซึ่งวิธีการที่ถูกต้องคือ จะต้องวางนิ้วให้เล็บเลยกระจกรับภาพไปทางด้านบน และให้ปลายนิ้วส่วนที่เป็นเนื้อ วางอยู่ที่ขอบบนของกระจกพอดี

อย่าหมุนนิ้ว

ในบางครั้งผู้ใช้อาจจะคิดว่า การที่อุปกรณ์จะอ่านลายนิ้วมือได้ดี จะต้องกดแล้วเอียงนิ้วซ้าย-ขวา ในลักษณะของการพิมพ์ลายนิ้วมือบนกระดาษ ซึ่งเป็นวิธีการที่ไม่แนะนำอย่างยิ่งเนื่องจากการกดในลักษณะเช่นนั้นจะทำให้ข้อมูลที่ได้อาจจะไม่มีความแม่นยำเท่าที่ควร และจะไม่ได้ผลที่ดีกว่า เมื่อเทียบกับการกดนิ้วลงไปนิ่งๆ บนกระจกรับภาพ

มือแห้ง

ถึงแม้ว่าเราอาจจะพบกับปัญหานี้น้อยมากๆ อย่างไรก็ตามในบางกรณี ความชื้นที่อยู่บนนิ้วมือก็เป็นส่วนที่สำคัญอย่างมากสำหรับการอ่านลายนิ้วมือโดยเครื่องอ่านลายนิ้วมือ ผู้ใช้ที่มีมือที่แห้งมากจนเกินไป อาจจะมีปัญหาในการใช้อุปกรณ์ ซึ่งวิธีการแก้ปัญหาสำหรับผู้ใช้นี้คือให้ผู้ใช้ใช้นิ้วถูกับฝ่ามืออีกข้างสักครู่ จนรู้สึกว่ามีน้ำความชื้นอยู่บ้าง และลองกดดูอีกครั้งหนึ่ง สำหรับผู้ใช้ที่มีปัญหาผิวหนังแห้งเป็นประจำ อาจจะต้องหมั่นใช้โลชั่นเพื่อให้มือมีความชุ่มชื้นบ้าง อย่างไรก็ตามผู้ใช้ควรหลีกเลี่ยงการใช้งานอุปกรณ์ทันทีหลังจากเพิ่งทาโลชั่นเสร็จใหม่ๆ เนื่องจากจะทำให้มีคราบโลชั่นติดบนกระจกรับภาพบนอุปกรณ์



การป้องกันปัญหาที่อาจเกิดขึ้น

มือสกปรก?

กรุณาตรวจสอบให้แน่ใจว่านิ้วมือของท่านสะอาดและไม่เปียกน้ำก่อนการใช้งานอุปกรณ์ทุกครั้ง โดยนิ้วมือของท่านไม่จำเป็นต้องสะอาดถึงขั้นที่จะต้องล้างมือก่อนการใช้งานแต่ก็ไม่ควรมีฝุ่นสกปรกที่อาจหลงเหลือติดอยู่บนเครื่องอ่านลายนิ้วมือ โดยทั่วไปการปิดฝุ่นสกปรกออกจากมือก่อนการใช้งานก็เพียงพอแล้วแม้แต่ในสภาพแวดล้อมที่มีการใช้งานอย่างสมบุกสมบันอย่างเช่นตามสถานที่ก่อสร้าง อย่างไรก็ตามเครื่องอ่านลายนิ้วมือที่ใช้งานในสภาพแวดล้อมเช่นนี้อาจต้องทำความสะอาดบ่อยครั้ง (กรุณาดูวิธีการทำความสะอาดเครื่องอ่านลายนิ้วมือในส่วนถัดไป)

แสงแดด

การวางเครื่องอ่านลายนิ้วมือไว้ในที่ที่มีแสงแดดจ้าอาจรบกวนการทำงานของเครื่องอ่านลายนิ้วมือได้

แอลกอฮอล์และน้ำยาทำความสะอาด

แอลกอฮอล์และน้ำยาทำความสะอาด จะทำลายแผ่นฟิล์มบางๆบนเครื่องอ่านลายนิ้วมือได้

น้ำตาล

เช่นเดียวกับแอลกอฮอล์ น้ำตาลที่อยู่บนนิ้วมือเมื่อผสมกับเหงื่อจะกัดกร่อนแผ่นฟิล์มบางๆบนเครื่องอ่านลายนิ้วมือ ถ้าเป็นไปได้ผู้ใช้งานที่อาจมีน้ำตาลอยู่บนนิ้วมือเช่นผู้ใช้งานที่ทำงานในครัว ควรล้างมือก่อนใช้งานเครื่องอ่านลายนิ้วมือ

การทำความสะอาดเครื่องอ่านลายนิ้วมือ

การทำความสะอาดจะขึ้นกับปริมาณการใช้งาน เครื่องอ่านลายนิ้วมืออาจต้องมีการทำความสะอาดบ้างเนื่องจากฝุ่นละอองและเหงื่อที่อาจตกหล่นอยู่จนทำให้ช่องอ่านลายนิ้วมือมัวได้

วิธีที่แนะนำในการทำความสะอาด

ในการทำความสะอาดฝุ่นละอองที่อาจตกหล่นอยู่ ให้ใช้สก็อตเทปชิ้นเล็กๆด้านที่มีกาวเหนียวติดลงบนช่องอ่านลายนิ้วมือแล้วดึงออก



ถ้าต้องการเช็ดทำความสะอาด ต้องใช้ผ้านุ่มที่ไม่มีขุยหลุดออกมาตะน้ำหรือน้ำยาเช็ดกระจกที่เป็นส่วนผสมของแอมโมเนียเช็ดเบาๆ

ข้อควรระวังในการทำความสะอาด

อย่าใช้กระดาษเช็ดช่องอ่านลายนิ้วมือ อย่าจุ่มหรือเทน้ำลงบนเครื่องอ่านลายนิ้วมือ และแอลกอฮอล์ จะทำลายแผ่นฟิล์มบางๆบนเครื่องอ่านลายนิ้วมือได้



คำถามและคำตอบเกี่ยวกับลายนิ้วมือ

99% ของลายนิ้วมือทั้งหมดจะสามารถนำมาใช้ยืนยันตัวบุคคลได้ตลอดเวลา อย่างไรก็ตามอีก 1% ที่เหลือจะไม่สามารถนำมาใช้ได้ (ปัญหาดังกล่าวจะพบในการนำลายนิ้วมือมาใช้งานทุกประเภท รวมไปถึงวิธีการของทางราชการ/ตำรวจและ FBI)

เราจำเป็นต้องเข้าใจปัจจัยสองอย่างที่จะช่วยอธิบายว่าทำไมลายนิ้วมือถึงถูกตรวจสอบว่าผ่าน หรือไม่ผ่านในการทดสอบแต่ละครั้ง โดยค่าแรกคือ False Acceptance Rate (FAR) ซึ่งเป็นค่าความน่าจะเป็นที่คนคนหนึ่งจะถูกยืนยันผิดว่าเป็นอีกคนหนึ่ง อีกค่าหนึ่งคือ False Rejection Rate (FRR) ซึ่งเป็นค่าความน่าจะเป็นที่คน คนหนึ่งซึ่งเป็นเจ้าของลายนิ้วมือ จะถูกยืนยันผิดว่าไม่ใช่เจ้าของลายนิ้วมือ จะเห็นได้ว่าเป็นเรื่องสำคัญมากกว่าที่เราจะต้องไม่ยืนยันคนคนหนึ่งผิดว่าเป็นอีกคนหนึ่ง

ระบบตรวจสอบลายนิ้วมือของ IDWORKS ได้ถูกออกแบบมาให้สามารถทำงานได้กับลายนิ้วมือที่มีคุณภาพต่ำมากๆ อย่างไรก็ตามลายนิ้วมือจำนวนน้อยมากๆของคนบางคนจะเกิดการสีกหรือจากการทำงานหนัก หรืออาจมีเส้นลายนิ้วมือที่ไม่สามารถอ่านออกได้ แม้แต่เครื่องอ่านลายนิ้วมือ U.are.U ก็ไม่สามารถลงทะเบียนลายนิ้วมือของคนเหล่านี้ได้

ในการพัฒนาระบบ IDWORKS ที่ใช้กับเครื่องอ่านลายนิ้วมือ U.are.U วิศวกรของ ไอที เวิร์คส์ ได้ใช้วิธีที่เหมาะสมและเทคนิคพิเศษต่างๆที่สามารถทำให้ค่า FAR ที่ต่ำมากจนกล่าวได้ว่าจะไม่มีการยืนยันตัวบุคคลผิดเกิดขึ้นได้ไม่ว่ากรณีใดๆ และเพื่อให้ได้ความถูกต้องสูงสุดค่า FRR ที่ต่ำมากๆก็ได้ถูกนำมาพิจารณาด้วย

ถาม: ระบบที่ใช้ ID-WORKS Integrator สามารถยืนยันลายนิ้วมือของคนทุกคนได้หรือไม่?

ตอบ: ไม่ได้ เป็นเรื่องน่าเสียดายที่ไม่ใช่ทุกคนจะมีลายนิ้วมือที่สมบูรณ์เสมอไป

เราแนะนำให้ผู้ใช้งานที่มีปัญหาในการลงทะเบียนลายนิ้วมือ

ต้องสังเกตว่าลายนิ้วมือนิ้วใดที่ดูจะมีเส้นที่ละเอียดและคมชัดที่สุด เพื่อที่จะได้เลือกใช้นิ้วมือนั้นในการลงทะเบียน

ถาม: เราจะทำอย่างไรกับคนที่ปัญหาไม่มีนิ้วใดเลยที่สามารถลงทะเบียนได้?

ตอบ: เราแนะนำให้ท่านพัฒนาโปรแกรมให้มีฟังก์ชันการทำงานด้วยคีย์บอร์ดแทนสำหรับคนที่ปัญหาในการลงทะเบียนลายนิ้วมือ

ถาม: ผู้ใช้งานสามารถทำอะไรได้บ้างที่จะทำให้ลายนิ้วมือของเขายืนยันด้วยระบบได้ดีขึ้น?

ตอบ: ผู้ใช้งานอาจลองการใช้ครีมบำรุงผิวเป็นประจำ หรืออาจลองดูนิ้วมือที่ใช้กับฝ่ามือก่อนที่จะแตะเครื่องอ่านลายนิ้วมือ วิธีดังกล่าวจะช่วยเพิ่มความชุ่มชื้นบนลายนิ้วมือซึ่งจะช่วยให้ผลการอ่านลายนิ้วมือนี้อดีขึ้นมาก

