Advanced Card Systems Ltd.
Card & Reader Technologies

AET62
NFC 读写器
带指纹传感器

应用程序编程接口 V1.01

如有任何更改，恕不另行通知

info@acs.com.hk
www.acs.com.hk

目录

图目录

1.0. 简介

AET62 是一款复合设备，由非接触式智能卡读写器和滑动式指纹传感器组成。非接触式智能卡读写器和指纹传感器既能各自单独使用，也可一起使用提高应用安全级别。AET62 的系统结构如下图所示：

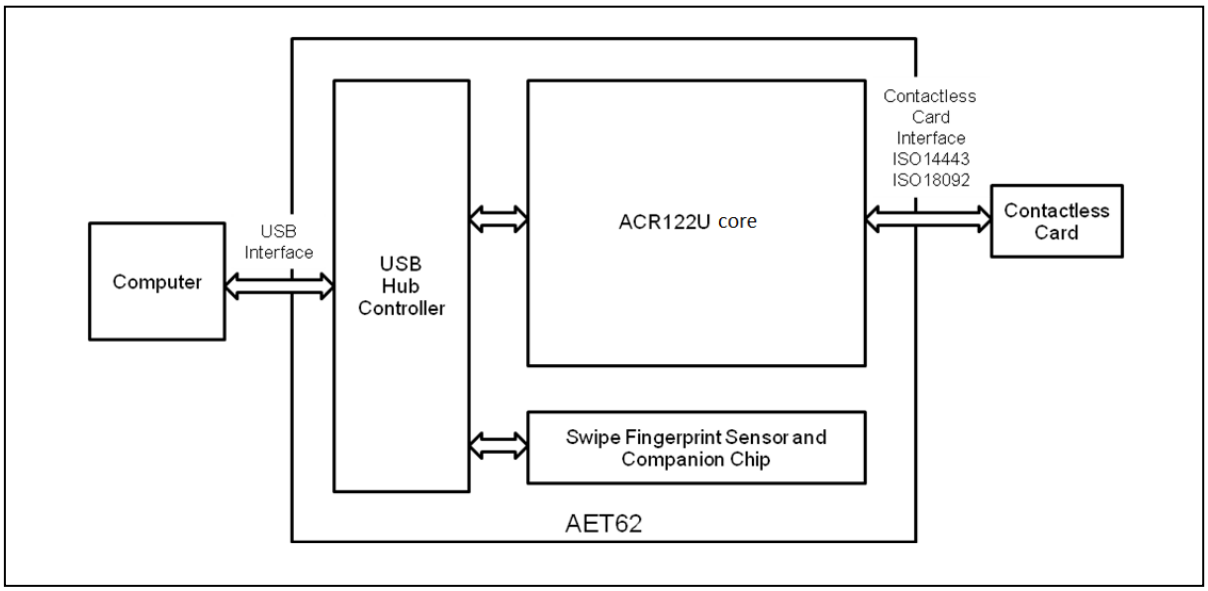

图1 : AET62 系统框图

基于 ACR122U NFC 读卡器内核的智能卡读写器模块符合 PC/SC API 标准。更多关于该智能卡读写器模块的信息，请参考 AET62 的参考手册。

本手册的目的是描述架构和 BSAPI（生物识别服务 API）。在 UPEK 的 BSAPI 参考手册的基础上，只涵盖与指纹传感器相关的应用程序编程接口。本手册描述了 BSAPI 及如何使用不同的库编写 AET62 的指纹模块程序。库负责处理通讯细节，参数转换及校错，为程序员提供了一个简单统一的接口，适用于所有可能涉及的硬件。BSAPI 架构如下图所示：

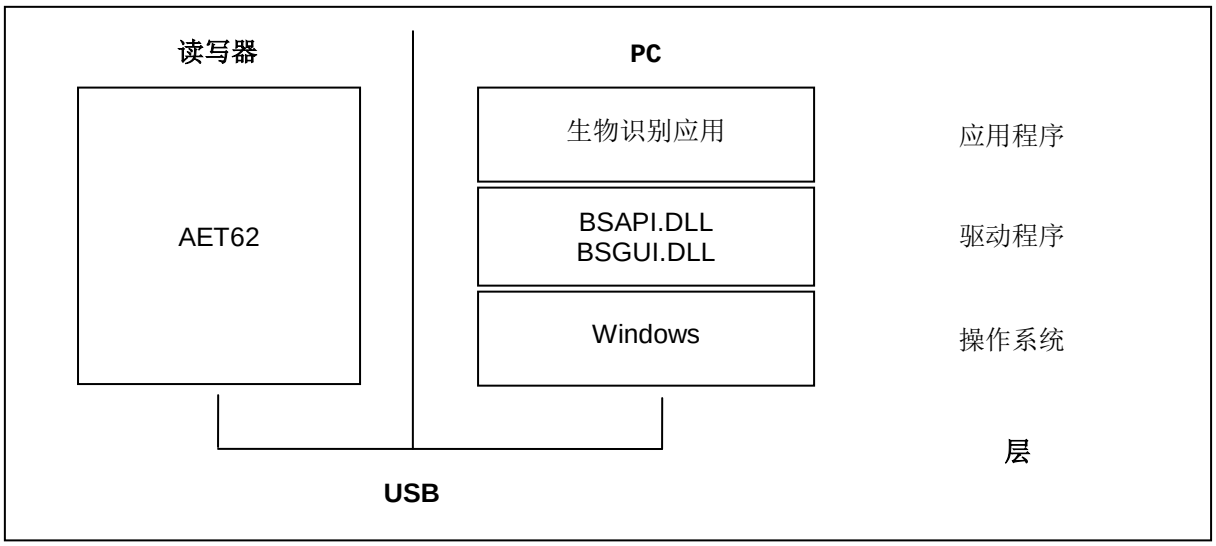

图2 : AET62 连接

2.0. BSAPI

2.1. 术语

术语	说明
BSAPI	生物识别服务 API
FM	主机连接的指纹模块（指纹读卡器）
主机	连接 FM 的计算机
BioAPI	生物识别 API 的行业标准是由 BioAPI Consortium（www.bioapi.org）。本文中涉及 BioAPI 的内容以 BioAPI 1.1 为准。
BioAPI 框架	参考 BioAPI Consortium 制定的 BioAPI 标准相关部分。
(BioAPI) BSP	生物识别服务提供商，第三方模块，可插入 BioAPI 框架。BSP 提供特定设备（例，指纹读写器）。
BSP API	BioAPI 框架下的 API 表示 BioAPI 框架和 BSP 之间的接口。
TFMESSBSP	UPEK BSP 实施，支持 TFM 和 ESS 设备。

2.2. 概述

BSAPI 专为满足以下要求而设计：

- 与 BioAPI 类似：熟悉 BioAPI 的研发团队能够在短时间内掌握 BSAPI 的用法。
- 兼容 TFMESSBSP：BSAPI 功能与 TFMESSBSP 的相应功能完全一致。（TFMESBSP 更新版本能在 BSAPI 上轻松运行。）
- 简单的 DLL 文件即可实现 BSAPI 功能：不同于 BioAPI，使用 BSAPI 不需要任何框架。
- 与 BioAPI 相比，BSAPI 功能更多，为设备提供更多生物识别及其他特性。
- 无需使用低级 API（例，PTAPI）。

如上所述，请勿混淆 BSAPI 与 BioAPI 之间的“兼容性”。BSAPI 与 BioAPI 之间没有二进制兼容性，二者的源层面也不兼容。BSAPI 提供的一组函数，能将所有 BioAPI 代码转换成 BSAPI 适用的代码，同时功能不变。

实际上，如下文所述，TFMESSBSP 更新版本将成为 BSAPI 上的薄层。

2.3. 架构

BSAPI 包括若干动态库（Windows 系统上显示为.DLL 文件，其他系统上后缀名不同）：

- **BSAPI.DLL**，包含所有核心功能。该库的源码通常与平台无关。如果底层库移植正常，移植 BSAPI 也能简单快捷地移植到新平台（例，Linux、Mac OS X）。
- **BSGUI.DLL 提供 GUI 回调函数的默认执行程序。**该库从独立的 ZIP 文件（窗口装饰和反馈图像）加载图像，用户可以自定义外观和感觉。该库支持多种语言（本地化），一般软件支持的语言该库都支持。这类情况下，应用自行回调，不需要调用库
  注：*目前，BSGUI.DLL 只在 Windows 系统平台上可用。*

应用研发人员可以从以下几个方面处理 GUI：

- 原封不动地使用 BSGUI.DLL，确保应用外观与自己的应用程序保持一致。
- 使用定制 ZIP 文件的 BSGUI.DLL（可修改或替换 ZIP 文件里的部分或全部图像）。
- 利用回调函数。这种方法定制的自由度最大，包括增加 BSGUI.DLL 不支持的语言。

BSAPI 在 BioFrame 顶层执行。BioFrame 高级管理库负责管理设备，质量保障和其他策略。该库使 BSAPI 用户专注应用逻辑，不必理会底层细节。

未来版本的 BioAPI BSP 将被设计成 BSAPI 顶部简单的一层，避免重复的 BSAPI 代码和 TFMESSBSP 执行程序.BSAPI 将静态链接到 TFMESSBSP，这样可以避免从 DLL 导出 BSAPI 符号，使 BioAPI 用户不会混淆同一个项目里的两个 API。静态库的特性也导致 TFMESSBSP 只包括必需的 BSAPI 库的子集。

2.4. 命名规则

所有标识（常量、类型和函数的名字）都带有前缀“ABS”，格式如下：

- `ABS_MACRO_IDENTIFIER`
- `ABS_TYPE_IDENTIFIER`
- `ABSFunctionIdentifier()`

函数原型可以使用特殊字符 IN、OUT、和 INOUT，分别表明是否用参数将数据传入函数，返回部分结果数据或两者都有。

3.0. BSAPI.DLL 函数

3.1. 总述

BSAPI.DLL 提供的函数可以从支持的指纹传感器读取数据，并执行各种生物识别算法。

bsapi.h 是声明 BSAPI 函数的主要头文件。头部包括 bstypes.h 和 bserror.h，用于声明类别和错误状态码。BSAPI 包含的其他库共享后两个头部。

3.1.1. 错误处理

几乎所有 BSAPI.DLL 函数都返回状态码 **ABS_STATUS**。代码 **ABS_STATUS_OK(0)**表示成功。所有其他值表示错误状态。

可以调用 **ABSGetLastErrorInfo 来获取错误状态信息。**获取的信息仅用于帮助应用程序和库开发人员，不开放给终端用户。

如果 BSAPI.DLL 函数执行失败，将释放自己能分配的所有资源。只有函数执行成功，即返回 ABS_STATUS_OK 的情况下，才会定义输出函数值。

3.1.2. 存储器管理

部分 BSAPI.DLL 函数可以通过输出函数为调用者程序分配存储。使用完存储后，调用者必须用 **ABSFree 释放存储。**

3.1.3. 交互操作

部分 BSAPI.DLL 函数需要用户与 FM 交互,本文将所有此类函数被称为交互操作类函数。交互操作函数采用一样的交互方法。

它们的第二个参数（紧随会话句柄）都是一个指向结构体 **ABS_OPERATION 的指针。**调用任意交互操作函数时，不能执行其他操作，直到交互操作完成或取消。处理操作时，将反复调用 **ABS_OPERATION 指定的回调函数，为用户提供反馈信息。**

注：执行回调有限制，成功的回调要求：

- 通过 C++或其他支持语言使用 BSAPI 时，回调函数不能抛出异常。
- 回调函数不能调用大部分 BSAPI 函数，仅能调用 **ABSFree**、**ABSCancelOperation** 和 **ABSGetLastErrorInfo。**

BSAPI 3.5 及以后版本，都是从已经调用了交互操作函数的线程上下文调用回调函数。但是，对于 BSAPI 3.5 以前的版本，这一点并不确定。

必要情况下可以用 **ABSCancelOperation 取消交互操作。**如果已经为待取消的交互操作关联了唯一的操作 ID，可以直接从回调函数自身或其他任意线程调用 ABSCancelOperation。

注：结构体 ABS_OPERATION 含有能影响回调函数的调用方式的成员标识位。更多信息，请参考 ABS_OPERATION 中的描述。

3.1.4. 多线程

BSAPI.DLL 一般是线程安全的。多线程（包括同一个 FM 上的多个生物识别操作）可同时调用 BSAPI.DLL 函数。如果多线程调用 BSAPI.DLL 函数时，BSAPI.DLL 正在与 FM 通信，则其中一个调用操作会被锁定。此间，对相关调用者来说，调用操作似乎仍在进行。BSAPI.DLL 与 FM 通信完毕后，将恢复锁定的调用操作。

如果被挂起的是交互操作函数，ABS_CALLBACK 会通知应用。更多信息，请参考描述 ABS_MSG_PROCESS_SUSPEND 和 ABS_MSG_PROCESS_RESUME 消息的文档。

只有函数 **ABSInitialize** 和 **ABSTerminate** **不是**线程安全的，但它们不影响应用的正常使用，因为调用这两个函数分别只是应用初始化和终止时操作的一部分。

3.1.5. 潜指纹检测

潜指纹检测的目标是将区域传感器表面残留指纹的消极影响最小化。消极影响有两种形式–安全性（风险或认假）和便利性（图像质量低导致拒真风险）。

注：潜指纹检测不适用于带状传感器，因为带状传感器没有潜指纹的问题，API 接口会自动上报带状传感器的上一次扫描结果没有潜指纹。

潜指纹检测有两种方法：1. 利用内置的高级生物识别操作：登记和验证。即，调用 ABSEnroll()或 ABSVerify()将自动引发潜指纹检测，函数不会返回被评估含有潜指纹的指纹模板。相反，函数将发送合适的回调消息要求用户清洗传感器并再次扫描指纹。这种隐式潜指纹检测是默认启用的，可以通过参数 ABS_PARAM_LATENT_CHECK 停用或再次启用。

2. 手动调用 ABSCheckLatent。BSAPI 自动记忆一段会话上下文里最后扫描的图像，并在 ABSCheckLatent 被调用时检查扫描图像是否含有潜指纹。

注：每次检测（隐式检测或调用 ABSCheckLatent 引发的显式检测）的逻辑如下：对比最后扫描图像和最新的有效图像。根据签指纹检测算法，如果两者一致，通过检测。如果不一致，最后扫描图像自动更新为最新有效图像，成为后续对比操作的参照图像。

不足之处：避免一次扫描后执行多次潜指纹检测（包括隐式潜指纹检测），因为只有第一次潜指纹检测会返回有效数据。所以，调用 ABSEnroll 或 ABSVerify 后不能再手动调用 ABSCheckLatent，除非已经用全局参数 ABS_PARAM_LATENT_CHECK 停止隐式检测。

3.2. 应用通用函数

应用通用函数用于 BSAPI 库初始化，打开和关闭主机与 FM 的连接，执行其他各种任务。

3.2.1. ABSInitialize

此函数用于 BSAPI 库的初始化。BSAPI 必须初始化后才能调用其他函数。典型地，应用启动时调用此函数。

```
ABS_STATUS ABSInitialize(
   void
)
```

返回值	说明
ABS_STATUS	结果码：ABS_STATUS_OK (0) 表示成功。

3.2.2. ABSInitializeEx

此函数用于 BSAPI 库的初始化。BSAPI 必须初始化后才能调用其他函数。典型地，应用启动时调用此函数。

```
ABS_STATUS ABSInitializeEx(
    IN ABS_DWORD dwFlags
)
```

参数	说明
dwFlags	Windows 系统仅支持 ABS_INIT_FLAG_NT_SERVICE 标识位。 其他系统不支持任何标识位。
返回值	ABS_STATUS　结果码：ABS_STATUS_OK (0) 表示成功。

3.2.3. ABSTerminate

此函数用于 BSAPI 库的去初始化。主机与 FM 有任何连接时，不得调用此函数。如果程序退出前 BSAPI 库要保持初始化，不强制但仍建议调用此函数。

```
ABS_STATUS ABSTerminate(
   void
)
```

返回值	说明
ABS_STATUS	结果码：ABS_STATUS_OK (0) 表示成功。

3.2.4. ABSOpen

此函数用于开始主机与所连接 FM 之间的新会话。

```
ABS_STATUS ABSOpen(
   IN const ABS_CHAR *pszDsn
   OUT ABS_CONNECTION *phConnection
)
```

参数	说明	
pszDsn	描述 FM 连接参数的零终止 ASCII 字符串。	
phConnection	产生的连接句柄。连接的尾部应为 ABSClose。	
返回值	ABS_STATUS	结果码。ABS_STATUS_OK (0) means success.
备注	断开连接，请调用 ABSClose。 通过 USB 打开连接时，不需要其他参数。 例： DSN = "usb" "device=X" （仅针对 USB） 如果系统附加了多个设备，可用该参数打开指定设备。X 是设备名称字符串，表示明确要求的设备，获得参数值需要调用函数 ABSEnumerateDevices（实际上，ABSEnumerateDevices 将返回整个 DSN 字符串）。注： X 与主机当前的系统配置相关。 例： DSN = "usb,device=\\?\usb#vid_0483&pid_2016#5&20890ddc&0&1#{d5620e51-8478-44bd-867e-aac02f883a00}"	

3.2.5. ABSClose

此函数用于关闭 ABSOpen 打开的连接。

```
ABS_STATUS ABSClose(
    IN ABS_CONNECTION hConnection
)
```

参数	说明
hConnection	待关闭的连接的句柄
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。
备注	每一次成功调用 ABSOpen，都要相应调用 ABSClose。

3.2.6. ABSEnumerateDevices

此函数用于列举当前连接的指纹设备。

```
ABS_STATUS ABSEnumerateDevices(
   IN const ABS_CHAR *pszEnumDsn
   OUT ABS_DEVICE_LIST **ppDeviceList
)
```

参数	说明
pszEnumDsn	零终止 ASCII 字符串,描述执行列举操作的连接接口。 例：用字符串"usb"字符串列举所有连接 USB 设备。
ppDeviceList	指针地址。指针指向被发现设备的清单。释放数据必须调用 ABSFree。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。
备注	只列举 USB 接口当前连接的设备。

3.2.7. ABSGetDeviceProperty

此函数返回的数据描述了当前打开的会话关联的设备的属性。

```
ABS_STATUS ABSGetDeviceProperty(
    IN ABS_CONNECTION hConnection
    IN ABS_DWORD dwPropertyId
    OUT ABS_DATA **ppPropertyData
)
```

参数	说明
hConnection	FM 连接的句柄。
dwPropertyId	ABS_DEVPROP_xxxx 常量，表示调用者需要的设备属性。
ppPropertyData	指针地址。指针指向数据块。数据内容与 dwPropertyId 相关。 释放数据必须调用 ABSFree。 更多信息，请参考常量相关文档。
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。

3.2.8. ABSFree

此函数用于释放其他 BSAPI.DLL 函数分配的内存。

```
void ABSFree(
   IN void *Memblock
)
```

参数	说明
Memblock	待释放内存块的地址。如果值为 NULL，参数无效。

3.3. 生物识别函数

本章描述了如何使用生物识别函数。

3.3.1. ABSEnroll

此函数用于扫描现场指纹，处理成指纹模板后返还给调用者。

```
ABS_STATUS ABSEnroll(
   IN ABS_CONNECTION hConnection
   IN ABS_OPERATION *pOperation
   OUT ABS_BIR **ppEnrolledTemplate IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pOperation	请参考 ABS_OPERATION 的描述。
Ppenrolled Template	指针地址。指针指向生成的模板（BIR）。要丢弃模板时,必须调用 ABSFree。
dwFlags	保留为将来所用。设置为 0。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

3.3.2. ABSVerify

此函数用于捕获 FM 上的样本，处理成模板后与 pTemplateArray 指定的另一个模板进行对比，确认指纹是否一致。

```
ABS_STATUS ABSVerify(
   IN ABS_CONNECTION hConnection
   IN ABS_OPERATION *pOperation
   IN ABS_DWORD dwTemplateCount
   IN ABS_BIR **pTemplateArray
   OUT ABS_LONG *pResult
   IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pOperation	请参考 ABS_OPERATION 的描述。
dwTemplateCount	pTemplateArray 中模板的数量。
pTemplateArray	指向模板数组的指针。
pResult	指针，指向存储对比结果的内存位置。如果与模板匹配，将结果编入 pTemplateArray 索引。如果不匹配，此函数的值为-1。
dwFlags	表示标识位（轻微修改函数操作）的位掩码。 函数支持 ABS_FLAG_NOTIFICATION 和 ABS_FLAG_AUTOREPEAT 标识位。 如果设置了 ABS_FLAG_NOTIFICATION，滑动指纹后验证操作才显示 GUI 反馈信息，以便应用各司其职，只有用户滑动指纹时候才需要做出特殊动作。用户滑动指纹前不会有任何对话框干扰。 显示/隐藏对话框是由 ABS_MSG_DLG_SHOW 和 ABS_MSG_DLG_HIDE 消息控制的。 如果使用了 ABS_FLAG_AUTOREPEAT 标识位，用户指纹不匹配 pTemplateArray 模板时自动开始重启验证操作。这种方法能避免在调用操作中隐藏和显示对话，所以它优于用户指纹匹配 pTemplateArray 模板前一直循环调用函数的方法。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

3.3.3. ABSVerifyMatch

此函数用于对比两个指定模板是否匹配。

```
ABS_STATUS ABSVerifyMatch(
    IN ABS_CONNECTION hConnection
    IN ABS_BIR *pEnrolledTemplate
    IN ABS_BIR *pVerificationTemplate
    OUT ABS_BOOL *pResult
    IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pEnrolledTemplate	待对比的第一个模板。 最常见地，匹配登记模板同另一种用途的模板时，必须用该参数传递登记模板。
pVerificationTemplate	待对比的第二个模板。 最常见地，匹配登记模板同另一种用途的模板时，必须用该参数传递另一模板。
pResult	用于设置对比结果的输出参数。如果两个 BIR 匹配，设为 ABS_TRUE；否则，设为 ABS_FALSE。
dwFlags	保留为将来所用。设置为 0。
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。

3.3.4. ABSCapture

此函数用于捕获特定用途的样本，并基于样本创建新的指纹模板。

```
ABS_STATUS ABSCapture(
    IN ABS_CONNECTION hConnection
    IN ABS_OPERATION *pOperation
    IN ABS_DWORD dwPurpose
    OUT ABS_BIR **ppCapturedTemplate
    IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pOperation	请参考 ABS_OPERATION 的描述。
dwPurpose	表示捕获生物识别数据的用途的值。 值为 ABS_PURPOSE_ENROLL 或 ABS_PURPOSE_VERIFY。禁止使用 ABS_PURPOSE_UNDEFINED。 注： 如果 dwPurpose 设置为 ABS_PURPOSE_ENROLL，调用 ABSCapture()等于调用 ABSEnroll()。
ppCapturedTemplate	指针,指向新分配模板。调用者须用 ABSFree 释放内存。
dwFlags	表示标识位（轻微修改函数操作）的位掩码。 此函数只支持 ABS_FLAG_NOTIFICATION。如果设置了 ABS_FLAG_NOTIFICATION，滑动指纹后验证操作才显示 GUI 反馈信息，以便应用各司其职，只有用户滑动指纹时候才需要做出特殊动作。用户滑动指纹前不会有任何对话框干扰。 显示/隐藏对话框是由 ABS_MSG_DLG_SHOW 和 ABS_MSG_DLG_HIDE 消息控制的。 只有 dwPurpose 设为 ABS_PURPOSE_VERIFY 时，才能使用标识位 ABS_FLAG_NOTIFICATION。
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。

3.3.5. ABSCheckLatent

此函数用于执行潜指纹检测。

注：*ABSEnroll 和 ABSVerify 默认执行隐式检测，除非已用全局参数 ABS_PARAM_LATENT_CHECK 关闭隐式检测。如果全局参数 ABS_PARAM_LATENT_CHECK 启用，调用 ABSVerify() 和 ABSEnroll() 这类函数后避免调用 ABSCheckLatent。*

更多有关潜指纹检测的信息，请参见 3.1.5 小节。

```
ABS_STATUS ABSCheckLatent(
   IN ABS_CONNECTION hConnection
   IN void *pReserved
   OUT ABS_BOOL *pboIsLatent
   IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pReserved	保留为将来所用。设置为 NULL。
pboIsLatent	指针，指向 ABS_BOOL。ABS_BOOL 用于存储检测结果。如果扫描的最新指纹被检测为潜指纹，它的值设置为 ABS_TRUE；否则，设为 ABS_FALSE。
dwFlags	保留为将来所用。设置为 0。
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。

3.3.6. ABSNavigate

此函数用于将 FM 切换到导航模式（a.k.a. 生物识别鼠标）。

仅部分设备支持导航模式。如果设备不支持，返回 ABS_STATUS_NOT_SUPPORTED。注：极少几台旧机器上的导航功能已经被破坏。这几台机器上，函数不会返回 ABS_MSG_NAVIGATE_CHANGE 消息给回调函数，导致导航模式不可用。

函数不返回 ABS_STATUS_OK 的原因是只有用 ABSCancelOperation（或者发生错误）取消函数的时候才会返回 ABS_STATUS_OK。

```
ABS_STATUS ABSNavigate(
   IN ABS_CONNECTION hConnection
   IN ABS_OPERATION *pOperation
   IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pOperation	请参考 ABS_OPERATION 的描述。
dwFlags	保留为将来所用。设置为 0。
返回值	ABS_STATUS 结果码。

3.4. 抓图函数

从传感器获取指纹图像有四种函数，各有利弊。本节将总结其不同之处，帮助用户正确抉择。

函数 ABSGrab 和 ABSGrabImage 与设备无关，即，使用它们时无需了解设备是否支持这两个函数。

函数 ABSRawGrab 和 ABSRawGrabImage 与设备相关，允许图像扫描过程中有较低级别的调整，包括检查图像质量和其他需要调整的属性。

对于函数 ABSGrab，用户可以通过使用或不使用标识位 ABS_FLAG_HIGH_RESOLUTION 来选择需要的图像格式（可能性很有限）。

函数 ABSRawGrab 允许通过抓取文件记录的形式明确指定图像格式。要使用这一功能，用户必须知道设备支持的格式。更多细节，请参考常量 ABS_PKEY_IMAGE_FORMAT 和 ABS_IFMT_xxxx 的文档。

函数 ABSGrabImage 和 ABSRawGrabImage 要求用户以 ABS_IMAGE_FORMAT 结构体描述需要的图像格式。可以通过函数 ABSListImageFormats 获取支持的格式列表。

函数 ABSGrab 和 ABSGrabImage 要求用户重复滑动指纹，直到其指纹图像通过内部的质量检查。这些检查为获取真实的指纹图像提供了一些保障。

相反，函数 ABSRawGrab 和 ABSRawGrabImage 在第一次滑动指纹以后就会返回图像。调用者自行检查图像质量并按需再次调用函数。

3.4.1. ABSGrab

此函数用于从 FM 抓取图像样本。

请注意获得符合最低质量要求的图像之前，抓图操作会自动重复，除非发生异常或已用 ABSCancelOperation 取消抓图操作。

```
ABS_STATUS ABSGrab(
    IN ABS_CONNECTION hConnection
    IN ABS_OPERATION *pOperation
    IN ABS_DWORD dwPurpose
    OUT ABS_IMAGE **ppImage
    IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pOperation	请参考 ABS_OPERATION 的描述。
dwPurpose	表示捕获生物识别数据的用途的值。 可能是任意 ABS_PURPOSE_xxxx 常量。
ppImage	函数，用于设置指针指向新分配的样本图像。用 ABSFree 释放分配的内存。
dwFlags	仅支持 ABS_FLAG_HIGH_RESOLUTION 标识位。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

3.4.2. ABSRawGrab

此函数用于从 FM 抓取图像样本。功能与 ABSGrab 类似，但是相对低级。

允许指定若干个特殊调整参数，方便用户为了特定用途而调整抓取操作。这些标识位能指定需要执行的质量检查项，图像格式和其他选项。

请注意所有的选项都是因设备而异的。各种设备模型从不同程度调整抓图操作的不同方面。请尽量选用 ABSGrab 或 ABSGrabImage。

不同于 ABSGrab，此函数只需刷一次指纹，即使图像质量低，也会结束操作。调用者可以用输出参数对得到的样本图像质量进行进一步检查。

```
ABS_STATUS ABSRawGrab(
    IN ABS_CONNECTION hConnection
    IN ABS_OPERATION *pOperation
    IN ABS_DWORD dwProfileSize
    IN ABS_PROFILE_DATA *pProfileData
    OUT ABS_IMAGE **ppImage
    OUT ABS_SWIPE_INFO **ppSwipeInfo
    IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pOperation	请参考 ABS_OPERATION 的描述。
dwProfileSize	确定 pProfileData 里有多少属性。
pProfileData	指针，指向配置文件数据数组的第一个成员。 更多信息，请参考 ABS_PROFILE_DATA 类型的描述。
ppImage	函数，用于设置指针指向新分配的样本图像。 用 ABSFree 释放分配的内存。
ppSwipeInfo	如果启用了该参数（即参数值不为 NULL），将为调用者提供滑动操作的补充信息。 用 ABSFree 释放返回的内存块。更多信息，请参考 ABS_SWIPE_INFO 的描述。
dwFlags	表示标识位（轻微修改函数操作）的位掩码。仅支持 ABS_FLAG_STRICT_PROFILE 标识位。 配置文件默认与设备相关，即自动忽略设备不支持的配置数据。相反，如果设置了标识位 ABS_FLAG_STRICT_PROFILE，配置数据的解释相对严格，如果设备不支持任何请求的调整参数或特定值，函数都会返回 ABS_STATUS_NOT_SUPPORTED。 注： 要求的图像格式（ABS_PKEY_IMAGE_FORMAT）总是严格的，即调用者必须知道使用的 FM 是否支持相关图像格式。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

3.4.3. ABSListImageFormats

此函数用于获取 FM 支持的图像格式列表。

ABSGrabImage 和 ABSRawGrabImage 的图像格式是由参数 ABS_FORMAT_IMAGE 定义的。

```
ABS_STATUS ABSListImageFormats(
   IN ABS_CONNECTION hConnection
   OUT ABS_DWORD *pdwCount
   OUT ABS_IMAGE_FORMAT **ppImageFormatList
   IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pdwCount	返回的图像格式个数。
ppImageFormatList	指针，用于存储新分配的图像格式结构体的数组。用 ABSFree 释放内存。
dwFlags	保留为将来所用。设置为 0。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

3.4.4. ABSGrabImage

此函数用于从 FM 抓取图像样本。

请注意获得符合最低质量要求的图像之前，抓图操作会自动重复，除非发生异常或已用 ABSCancelOperation 取消抓图操作。

```
ABS_STATUS ABSGrabImage(
    IN ABS_CONNECTION hConnection
    IN ABS_OPERATION *pOperation
    IN ABS_DWORD dwPurpose
    IN ABS_IMAGE_FORMAT *pImageFormat
    OUT ABS_IMAGE **ppImage
    OUT ABS_SWIPE_INFO **ppSwipeInfo
    IN void *pReserved
    IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pOperation	请参考 ABS_OPERATION 的描述。
dwPurpose	表示捕获生物识别数据的用途的值。 可能是任意 ABS_PURPOSE_xxxx 常量。
pImageFormat	指针，指向描述需要的图像格式的结构体。 TODO
ppImage	函数，用于设置指针指向新分配的样本图像。 用 ABSFree 释放分配的内存。
ppSwipeInfo	如果启用了该参数（即参数值不为 NULL），将为调用者提供滑动操作的补充信息。 用 ABSFree 释放返回的内存块。更多信息，请参考有关 ABS_SWIPE_INFO 的描述。
pReserved	保留为将来所用。设置为 NULL。
dwFlags	保留为将来所用。设置为 0。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

3.4.5. ABSRawGrabImage

此函数用于从 FM 抓取图像样本。功能与 ABSGrabImage 类似，但是相对低级。

指定若干个特殊调整参数，方便用户为了特定用途而调整抓取操作。这些标识位能指定需要执行的质量检查项和其他选项。

请注意所有的选项都是因设备而异的。各种设备模型从不同程度调整抓图操作的不同方面。请尽量选用 ABSGrab 或 ABSGrabImage。

不同于 ABSGrabImage，此函数只需刷一次指纹，即使图像质量低，也会结束操作。调用者可以用输出参数对得到的样本图像质量进行进一步检查。

注：抓图配置文件含有密钥 ABS_PKEY_IMAGE_FORMAT，但该密钥不起作用，因为函数总是使用 pImageFormat 参数指定的格式。

```
ABS_STATUS ABSRawGrabImage(
    IN ABS_CONNECTION hConnection
    IN ABS_OPERATION *pOperation
    IN ABS_DWORD dwProfileSize
    IN ABS_PROFILE_DATA *pProfileData
    IN ABS_IMAGE_FORMAT *pImageFormat
    OUT ABS_IMAGE **ppImage
    OUT ABS_SWIPE_INFO **ppSwipeInfo
    IN void *pReserved
    IN ABS_DWORD dwFlags
)
```

参数	说明
hConnection	FM 连接的句柄。
pOperation	请参考 ABS_OPERATION 的描述。
dwProfileSize	确定 pProfileData 里有多少属性。
pProfileData	指针，指向配置文件数据数组的第一个成员。 更多信息，请参考 ABS_PROFILE_DATA 类型的描述。
pImageFormat	指针，指向描述需要的图像格式的结构体。
ppImage	函数，用于设置指针指向新分配的样本图像。 用 ABSFree 释放分配的内存。
ppSwipeInfo	如果启用了该参数（即参数值不为 NULL），将为调用者提供滑动操作的补充信息。 用 ABSFree 释放返回的内存块。更多信息，请参考有关 ABS_SWIPE_INFO 的描述。
pReserved	保留为将来所用。设置为 NULL。

参数	说明
dwFlags	位掩码，表示轻微修改函数操作的标识位。仅支持标识位 ABS_FLAG_STRICT_PROFILE。 配置文件默认与设备相关，即自动忽略设备不支持的配置数据。相反，如果设置了标识位 ABS_FLAG_STRICT_PROFILE，配置数据的解释相对严格，如果设备不支持任何请求的调整参数或特定值，函数都会返回 ABS_STATUS_NOT_SUPPORTED。 注：要求的图像格式（ABS_PKEY_IMAGE_FORMAT）总是严格的，即调用者必须知道使用的 FM 是否支持相关图像格式。
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。

3.5. 各种函数

3.5.1. ABSCancelOperation

此函数用于取消正在进行的交互操作。被取消操作的函数返回 ABS_STATUS_CANCELED。

```
ABS_STATUS ABSCancelOperation(
   IN ABS_CONNECTION hConnection
   IN ABS_DWORD dwOperationID
)
```

参数	说明
hConnection	FM 连接的句柄。
dwOperationID	待取消操作的 ID，值为 0 表示取消当前线程中正处理的操作。 即，在回调函数中使用 0 可以取消调用该回调函数的操作。如果交互操作的 OperationID 设为 0，这是取消交互操作的唯一办法。 更多信息，请参考有关 ABS_OPERATION 的成员 OperationID 的描述。
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。

3.5.2. ABSSetAppData

此函数将任意数据存储到 FM 上。

其后，可以用函数 ABSGetAppData 获取数据。这些数据存在于整个 BSAPI 会话中，直到下一次调用此函数时覆盖这些数据。

注：*数据的最大长度是受限的，具体与设备型号相关。*

```
ABS_STATUS ABSSetAppData(
   IN ABS_CONNECTION hConnection
   IN ABS_DATA *pAppData
)
```

参数	说明
hConnection	FM 连接的句柄。
pAppData	待存储到设备的数据。
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。

3.5.3. ABSGetAppData

此函数用于获取 FM 上存储的数据。另见有关函数 SetAppData 的描述。

```
ABS_STATUS ABSGetAppData(
   IN ABS_CONNECTION hConnection
   OUT ABS_DATA **ppAppData
)
```

参数	说明
hConnection	FM 连接的句柄。
ppAppData	输出参数，待设置给新分配的 ABS_DATA 结构体。 用 ABSFree 释放分配的内存。
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。

3.5.4. ABSSetSessionParameter

此函数用于设置会话宽参数。

这些设置影响当前会话情景下调用的某些 BSAPI 函数的行为。

注：*当前版本里的所有参数都是全局参数，只能用 ABSSetGlobalParameter()进行设置。此函数当前不支持任何 dwParamID 值，因此返回 ABS_STATUS_INVALID_PARAMETER。*

```
ABS_STATUS ABSSetSessionParameter(
   IN ABS_CONNECTION hConnection
   IN ABS_DWORD dwParamID
   IN ABS_DATA *pParamValue
)
```

参数	说明
hConnection	FM 连接的句柄。
dwParamID	待设置参数的 ID。请参见有关常量 ABS_PARAM_xxxx 的描述。
pParamValue	参数值。数据格式和含义与参数相关。 更多信息，请参考特定常量 ABS_PARAM_xxxx（用作 dwParamID）的描述。
返回值	ABS_STATUS　结果码。ABS_STATUS_OK (0) 表示成功。

3.5.5. ABSGetSessionParameter

此函数用于获取会话宽参数的值。

注：*当前版本里的所有参数都是全局参数，只能用 ABSSetGlobalParameter()进行设置。此函数当前不支持任何 dwParamID 值，因此返回 ABS_STATUS_INVALID_PARAMETER。*

```
ABS_STATUS ABSGetSessionParameter(
   IN ABS_CONNECTION hConnection
   IN ABS_DWORD dwParamID
   OUT ABS_DATA **ppParamValue
)
```

参数	说明	
hConnection	FM 连接的句柄。	
dwParamID	待设置参数的 ID。请参见有关常量 ABS_PARAM_xxxx 的描述。	
pParamValue	输出参数，表示获取的值。函数将设置它指向新分配的 ABS_DATA 。 用 ABSFree 释放内存。 关于特定值的含义，请参考常量 ABS_PARAM_xxxx 的描述。	
返回值	ABS_STATUS	结果码。ABS_STATUS_OK (0) means success.

3.5.6. ABSSetGlobalParameter

此函数用于设置全局参数的值。

设置影响某些 BSAPI 函数的行为。不同于 ABSSetSession-参数，这些设置应用于进程上下文中调用的所有 BSAPI 函数，不管当前会话的情况如何。

```
ABS_STATUS ABSSetGlobalParameter(
   IN ABS_DWORD dwParamID
   IN ABS_DATA *pParamValue
)
```

参数	说明
dwParamID	待设置参数的 ID。 请参见有关常量 ABS_PARAM_xxxx 的描述。
pParamValue	参数值。数据格式和含义与参数相关。 更多信息，请参考特定常量 ABS_PARAM_xxxx（用作 dwParamID）的描述。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

3.5.7. ABSGetGlobalParameter

此函数用于获取全局参数的值。

```
ABS_STATUS ABSGetGlobalParameter(
   IN ABS_DWORD dwParamID
   OUT ABS_DATA **ppParamValue
)
```

参数	说明
dwParamID	待获取的参数的 ID。请参见有关常量 ABS_PARAM_xxxx 的描述。
pParamValue	输出参数，表示获取的值。函数将设置它指向新分配的 ABS_DATA。 用 ABSFree 释放内存。 关于特定值的含义，请参考常量 ABS_PARAM_xxxx 的描述。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

3.5.8. ABSSetLED

此函数用于控制两个用户接口 LED（可选择性连接到 FM）的状态和行为。

```
ABS_STATUS ABSSetLED(
    IN ABS_CONNECTION hConnection
    IN ABS_DWORD dwMode
    IN ABS_DWORD dwLED1
    IN ABS_DWORD dwLED2
)
```

参数	说明
hConnection	FM 连接的句柄。
dwMode	LED 型号。不同型号定义具体操作中的各种 LED 行为，特别是生物识别相关的行为。 值为 0 时，LED 处于手动工作模式。手动模式下，主机应用直接控制 LED 状态，包括设置闪烁类型。 其他值（不管它们是否被支持）都是因设备而异的。
dwLED1	定义第一个 LED 具体行为的参数。 该参数与具体型号相关。 对于手动模式（dwMode = 0），可以将值理解成位掩码，其含义如下： • Bits 0-15：闪烁类型。该设备遍历各个位（由位 16-19 确定的一个时钟周期里的一个位，如下所示），然后打开（位值是 1）或关闭（位值是 0）LED。首先显示 Bit 0，下一个时钟周期显示 Bit 1，以此类推。 • Bits 16-19：闪烁时钟周期指数。值 0 表示停止闪烁； 值 1 表示每个类型位占 1 毫秒； 值 2 表示每个类型位占 2 毫秒； 值 3 表示每个类型位占 4 毫秒； ... 值 15 表示每个类型位占 16384 秒；(值 N=每个类型位占 2^{N-1} 毫秒) 永久关闭 LED,使用类型"所有为 1"。永久打开 LED,使用类型"所有为 0"。
dwLED2	定义第二个 LED 具体行为的参数。 该参数与具体型号相关。 **对于手动模式（dwMode = 0），其含义与 dwLED1 一样：**
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

参数	说明
备注	FM 重启后，LED 默认处于手动模式（dwMode = 0），并且两个 LED 都关闭。如果关闭一个连接（会话），LED 将保持此前状态。如果连接关闭前 LED 处于闪烁，连接关闭后它们将继续闪烁。 因此，开始连接时，两个 LED 的状态可能不同。主机应用需要调用 ABSGetLED/ABSSetLED 来确认和/或定义 LED 状态。 当 FM 进入深度睡眠或备用状态，会关闭 LED，直到睡眠结束。被唤醒后，LED 将恢复睡眠前的状态（包括闪烁）。

3.5.9. ABSGetLED

此函数用于查询两个用户接口 LED（可选择性连接到 FM）的状态和行为。

关于本主题的更多信息，请参考 ABSSetLED 函数的有关文档。

```
ABS_STATUS ABSGetLED(
    IN ABS_CONNECTION hConnection
    OUT ABS_DWORD *dwMode
    OUT ABS_DWORD *dwLED1
    OUT ABS_DWORD *dwLED2
)
```

参数	说明
hConnection	FM 连接的句柄。
dwMode	返回 LED 的状态模式。更多信息，请参考有关 ABSSetLED()的描述。
dwLED1	返回定义第一个 LED 具体行为的参数。 更多信息，请参考有关 ABSSetLED()的描述。
dwLED2	返回定义第二个 LED 具体行为的参数。 更多信息，请参考有关 ABSSetLED()的描述。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。
备注	根据最新调用的 ABSGetLED 的设置，ABSGetLED 返回 LED 的状态。 手动模式（dwMode = 0）下，如果 LED 正在闪烁，返回值不包含表明 LED 位类型所处阶段的信息。

3.5.10. ABSBinarizeSampleImage

此函数用于将灰阶图像（用回调函数 ABSGrab 或 ABSRawGrab 获得的图像）转化成二元图像（只包含两种颜色）。

对于二元图像样本，ColorCount 设置为 2。两种颜色包括黑色（1）和白色（0）。二元图像显示效果好，更适合显示给终端用户。

***注：**如果 ppBinarizedImage 设为 NULL，表示就地转换；如果 ppBinarizedImage 设置不为 NULL，表示转换到新分配的图像结构体。*

```
ABS_STATUS ABSBinarizeSampleImage(
   INOUT ABS_IMAGE *pGrayScaleImage
   OUT ABS_IMAGE **ppBinarizedImage
)
```

参数	说明
pGrayScaleImage	指针，指向输入的灰阶图像结构体。 注： 此函数不支持所有图像格式，只支持每像素占 8 位（ABS_IMAGE::ColorCount == 256）和 381 x 381 DPI 或 508 x 508 DPI 的图像。 如果 ppBinarized5Image 设置为 NULL，则会当场修改该结构体的内容。
ppBinarizedImage	可选的输出参数，用于获取新的二元图像样本。 如果设置不为 NULL，转换的图像样本将被放置于该输出参数指定的新分配的缓存。然后，调用者负责用 ABSFre5e 释放内存。 如果设置为 NULL，就地转换原有的 pGraySc5aleImage。
返回值	ABS_STATUS　　结果码。ABS_STATUS_OK (0) 表示成功。

3.5.11. ABSGetLastErrorInfo

此函数用于获取当前线程发生的上一个 BSAPI 错误的附加信息。

注：此函数提供的信息不显示给终端用户。错误消息显示为英语（从未本地化），提示研发人员如何更简单地诊断问题。

```
void ABSGetLastErrorInfo(
   OUT ABS_DWORD *pErrorCode
   OUT const ABS_CHAR **ppErrorMessage
)
```

参数	说明
pErrorCode	输出参数，设为附加的系统相关错误码。 根据具体系统，可能是 Windows 平台上返回的错误或值或任意其他错误码，可提示研发人员如何诊断错误。
ppErrorMessage	输出时，设置为指向含有带文本消息的零终止字符串的缓存。 如果没有提供消息，设置为指向空字符串，调用者不必检查，因为它是空值。 由 BSAPI 管理缓存；不要使用 ABSFree 释放缓存。 注： 同一线程中调用其他 BSAPI 时，该缓存就失效了。下一次调用时，该缓存可能被 BSAPI 释放或重用。如需记住消息，用户必须将其复制到自己的缓存。

3.5.12. ABSEscape

此函数用于要求处理特殊功能。

```
ABS_STATUS ABSEscape(
    IN ABS_DWORD dwOpcode
    IN ABS_DATA *pInData
    OUT ABS_DATA **ppOutData
)
```

参数	说明
dwOpcode	待执行的操作码。 当前不支持任何操作码。即，当前版本里调用此函数总是失败。
pInData	输入数据，传输给 dwOpcode 指定的函数。 数据格式与 dwOpcode 相关。如果 dwOpcode 不要求任何输入数据，该参数值可以设为 NULL。
ppOutData	作为操作结果回传给调用者的数据。 如果没有数据回传，可以把参数值设为 NULL。
返回值	ABS_STATUS 结果码。ABS_STATUS_OK (0) 表示成功。

4.0. BSGUI.DLL 函数

4.1. 使用 BSGUI.DLL

BSGUI.DLL 为 BSAPI.DLL 提供默认的 ABS_CALLBACK 执行程序。

如需使用默认的回调函数执行程序，请将应用同时链接到 BSAPI.DLL 和 BSGUI.DLL。因此，无论何时开始交互操作，结构体 ABS_OPERATION 的成员回调函数都设置成从 BSGUI.DLL 指向函数 ABSDefaultCallback 的指针。

安装应用时，文件 BSGUI.ZIP 必须与 BSGUI.DLL 位于同一目录。

注：BSGUI.DLL 只在 Windows 系统平台上可用。

4.2. GUI 定制

BSGUI.DLL 库从 BSGUI.ZIP 加载图像。

如需定制 BSGUI.DLL 会话的外观和感觉，可以替换 BSGUI.ZIP 里的部分或全部图像，如有需要，也可以更改 BSGUI.ZIP 里 BIO.XML 的布局。

4.3. 默认的回调函数执行程序

4.3.1. ABSDefaultCallback

默认的 BSAPI 回调函数执行程序。

它提供回调函数的默认执行程序，然后可以用 ABS_OPERATION 结构体传入 BSAPI 交互函数。如需保持一致的应用外观和感觉，请使用该回调函数提供的执行程序而非用户自己的执行程序。

绝不能直接调用此函数。此函数仅用作结构体 ABS_OPERATION 的成员回调函数将指向函数的指针传入 BSAPI 函数。

更多信息，请参考 ABS_CALLBACK 类型和 ABS_OPERATION 结构体的文档。

```
void ABSDefaultCallback(
   IN const ABS_OPERATION *pOperation
   IN ABS_DWORD dwMsgID
   IN void *pMsgData
)
```

参数	说明
pOperation	调用交互生物识别操作时指向 ABS_OPERATION 结构体的指针。 交互操作的调用者可以使用结构体的成员 Context 将数据传入默认的回调函数。ABSDefaultCallback 需要数据符合 ABS_DEFAULT_CALLBACK_CONTEXT 结构体的格式。 Context 的指针可以设置成 NULL。这时，将实施默认行为。
dwMsgID	消息 ID。请参见常量 ABS_MSG_xxxx 的描述。
pMsgData	指针，指向有关消息附加信息数据。 其含义是与具体消息相关的。请参考具体常量 ABS_MSG_xxxx 的文档。

4.4. ABS_DEFAULT_CALLBACK_CONTEXT

结构体，可选择性地作为上下文数据传入 ABSDefaultCallback。

可以调整默认回调函数执行程序的实际行为。要使用时，建立结构体成员并将 ABS_OPERATION 的 Context 设置成结构体地址。

生物识别操作的调用者必须保证指向结构体的指针（通过 ABS_OPERATION 结构体传入）持续有效，直到生物识别操作结束。

```
typedef struct abs_default_callback_context {
   ABS_DWORD Version;
   HWND ParentWindow;
   ABS_DWORD Flags;
} ABS_DEFAULT_CALLBACK_CONTEXT
```

参数	说明
版本号	结构体的版本号。设置为 1。
ParentWindow	设置为上级窗口的句柄或者 NULL。 设为 NULL 时，上级窗口是实际活跃的窗口。
标识位	标识位的位掩码。目前仅支持标识位 ABS_DEFAULT_CALLBACK_FLAG_ENABLE_SOUND。

4.5. ABS_DEFAULT_CALL BACK_CONTEXT 的标识位（ABS Default_CALLBACK_FLAG_xxxx）

以下标识位可以用于结构体 ABS_DEFAULT_CALLBACK_CONTEXT：

ABS_DEFAULT_CALLBACK_FLAG_ENABLE_SOUND	1h
回调函数执行成功时播放声音。	

5.0. 声明

5.1. 基本种类

typedef 有符号的整数类型（1 个字节）	char	ABS_CHAR
typedef 没有符号的整数类型（1 个字节）	unsigned char	ABS_BYTE
typedef 有符号的整数类型（2 个字节）	短	ABS_SHORT
typedef 没有符号的整数类型（2 个字节）	unsigned short	ABS_WORD
typedef 有符号的整数类型（4 个字节）	int	ABS_LONG
typedef 没有符号的整数类型（4 个字节）	unsigned int	ABS_DWORD
typedef 布尔值（0、非 0）	int	ABS_BOOL
typedef 返回状态。	ABS_LONG	ABS_STATUS
typedef 连接句柄。它代表同 FM 的会话。	ABS_DWORD	ABS_CONNECTION

5.2. 具体类型

5.2.1. ABS_DATA

ABS_DATA 结构体用于关联任意长数据块和长度信息。

```
typedef struct abs_data {
   ABS_DWORD Length;
   ABS_BYTE Data[ABS_VARLEN];
} ABS_DATA
```

参数	说明
Length	数据域的长度，单位为字节。
Data[ABS_VARLEN]	数据本身的长度可变。

5.2.2. ABS_BIR_HEADER

它是 BIR 的头部。这种类型等同于 BioAPI 的结构体 BioAPI_BIR_HEADER。

典型地，BIR 作为不透明数据处理，不必知道其头部的结构体。但是，出于完整性的要求，这里还是介绍一下。下面提供的值是 FM 产生的 BIR 中用到的标准值。

更多详情，请参阅 BioAPI 文档。

注：ABS_BIR_HEADER 所有的成员都采用小端字节序。这种做法有两个重要影响：

- 模板采用完全一致的二进制，存储到存储器或数据库时，可以通用于各种平台，无论平台采用什么样的字节序。
- 使用结构体的值时，必须将值转换成当前所用平台内在的字节序。

```
typedef struct abs_bir_header {
   ABS_DWORD Length;
   ABS_BYTE HeaderVersion;
   ABS_BYTE Type;
   ABS_WORD FormatOwner;
   ABS_WORD FormatID;
   ABS_CHAR Quality;
   ABS_BYTE Purpose;
   ABS_DWORD FactorsMask;
} ABS_BIR_HEADER
```

参数	说明
Length	头部长度 + 不透明的数据
HeaderVersion	HeaderVersion = 1
Type	Type = 4 (BioAPI_BIR_DATA_TYPE_PROCESSED)
FormatOwner	FormatOwner = 12h (STMicroelectronics)
FormatID	FormatID = 0
Quality	Quality = -2（不支持 BioAPI_QUALITY）
Purpose	Purpose (BioAPI_PURPOSE_xxxx, ABS_PURPOSE_xxxx). 相应的 BioAPI 和 BSAPI 常量的值一样。
FactorsMask	FactorsMask = 08h (BioAPI_FACTOR_FINGERPRINT)

5.2.3. ABS_BIR

生物识别数据的容器。

BIR 的全称是 Biometric Identification Record（生物特征识别的记录）。在 BSAPI 里，它表示指纹模板，但是可能也含有其他数据，例如审计数据。BIR 有一个头部，头部后面是不透明数据和签名（签名可选）。这种数据是与 BioAPI 的 BioAPI_BIR 兼容的二进制数据。唯一的区别是 BioAPI_BIR 数据分成 4 个单独的内存模块，而 ABS_BIR 所有数据在一起。

```
typedef struct abs_bir {
   ABS_BIR_HEADER Header;
   ABS_BYTE Data[ABS_VARLEN];
} ABS_BIR
```

参数	说明
命令头	BIR 头部。
Data[ABS_VARLEN]	组成指纹模板的数据。

5.2.4. ABS_OPERATION

容纳所有交互操作函数的通用数据。

```
typedef struct abs_operation {
   ABS_DWORD OperationID;
   void* Context;
   ABS_CALLBACK Callback;
   ABS_LONG Timeout;
   ABS_DWORD Flags;
} ABS_OPERATION
```

参数	说明
OperationID	唯一操作 ID 或 0。 非 0 值表示操作的唯一标识。可以用于函数 ABSCancel-Operation 来取消操作，包括取消其他线程的操作。注：调用者负责分配操作 ID，以防同一会话中出现同时进行的交互操作（例，其他线程）拥有重复的操作 ID。否则操作立即失败，返回 ABS_STATUS_INVALID_PARAMETER。 值为 0 时，用户只能从其回调函数取消操作（将 0 作为 ABSCancelOperation 的参数进行传输）。
Context	用户定义的指针，传入操作回调函数。 BSAPI 不会以任何方式解释或区别指针数据。
Callback	指针，指向应用定义的函数，用于执行操作回调函数。 允许应用开发者提供的用户界面通知用户操作处理进度并提示用户如何操作，例，把手指放在 FM 传感器上。 更多信息，请参考 ABS_CALLBACK 的文档。
Timeout	用户不活跃造成的超时，单位为毫秒。 如果交互操作需要用户活跃但是指定时间内检测不到用户活跃，操作中断，操作函数返回 ABS_STATUS_TIMEOUT。 值 0 表示没有设置超时（用户的不活跃不会引起操作中断）。值-1 表示使用默认的超时设置（视具体设备而定）。
标识位	标识位的位掩码，用于调整操作进程。 更多信息，请参考 ABS_OPERATION_FLAG_xxxx 常量的描述。

5.2.5. ABS_PROFILE_DATA

配置文件数据，用于调整原始抓取操作（ABSRawGrab）。

可为原始抓取操作设置各种特殊模式和属性（包括与 FM 有关的）。

函数 ABSRawGrab 用指针指向 ABS_PROFILE_DATA 结构体的数组。其它参数表示配置文件数组中的项目号。数组的每一条记录都包括一对密钥值。

密钥指定待调整的原始抓取操作的属性/参数，密钥值指定如何调整这些属性和参数。密钥值的含义和范围与特定密钥以及 FM 性能相关。

```
typedef struct abs_profile_data {
   ABS_DWORD Key;
   ABS_DWORD Value;
} ABS_PROFILE_DATA
```

参数	说明
Key	配置文件的密钥。可能是任意 ABS_PKEY_xxxx 常量。
Value	值，与密钥相关。

5.2.6. ABS_SWIPE_INFO

提供 ABSRawGrab、ABSGrabImage 或 ABSRawGrabImage 获取的滑动操作的各种信息。

注：BSAPI 的将来版本中，ABSRawGrab 能以结构体定义的其他格式返回滑动操作的信息。更多信息，请参考 ABSRawGrab 及其参数 ppSwipeInfo 的描述。

```
typedef struct abs_swipe_info {
   ABS_DWORD Version;
   ABS_WORD Height;
   ABS_BYTE ReconstructionScore;
   ABS_BYTE ImageScore;
   ABS_DWORD MsgID;
   ABS_DWORD Flags;
   ABS_DWORD BackgroundColor;
} ABS_SWIPE_INFO
```

参数	说明
版本号	结构体的版本号。当前是版本 1。 即，如果返回数据的前四个字节不是 1，用户不能将其他数据的剩余部分理解为结构体 SWIPE_INFO。
Height	指纹图像高度，单位为像素。
ReconstructionScore	重构质量分数，范围 0-100。 值越大，图像重构质量越高。
ImageScore	图像质量分数，范围 0-100。 值越大，图像质量越高。
MsgID	质量反馈消息的 ID。 根据原始抓取配置文件的设置，可以在原始抓图操作中进行各种测试及质量检测。如果成功通过所有测试（或全部测试被停止），MsgId 设置为 ABS_MSG_PROCESS_SUCCESS。 如果任意质量检测失败，值设置为最重要/相关的回调消息的 ID（请参考常量 ABS_MSG_QUALITY_xxxx）。
标识位	位掩码，表示滑动操作的各方面情况。 请参考常量 ABS_SWIPE_FLAG_xxxx。
BackgroundColor	滑动指纹图像样本的背景色。 实际选择的颜色取决于 ABS_SWIPE_INFO 相关的样本图像。值为 0，表示黑色； 值为 1（ABS_IMAGE::ColorCount - 1）表示白色。其它灰阶色介于黑白之间。 如果不能确定背景色，则 BackgroundColor 设为 FFFFFFFFh。

5.2.7. ABS_IMAGE_FORMAT

用于描述函数 ABSGrabImage 和 ABSRawrabImage 需要的图像格式。

用函数 ABSListImageFormats 获取可用的格式清单。

分辨率的单位总是 DPI（dots per inch）。扫描分辨率是扫描时传感器的分辨率。图像分辨率是所产生图像的分辨率。二者的值相同，除非传感器对扫描图像进行二次抽样或指定硬件没有可用的二次抽样信息。

```
typedef struct abs_image_format {
   ABS_WORD ScanResolutionH;
   ABS_WORD ScanResolutionV;
   ABS_WORD ImageResolutionH;
   ABS_WORD ImageResolutionV;
   ABS_BYTE ScanBitsPerPixel;
   ABS_BYTE ImageBitsPerPixel;
} ABS_IMAGE_FORMAT
```

参数	说明
ScanResolutionH	水平扫描分辨率，单位 DPI。
ScanResolutionV	垂直扫描分辨率，单位 DPI。
ImageResolutionH	所产生图像的水平扫描分辨率，单位 DPI。
ImageResolutionV	所产生图像的垂直扫描分辨率，单位 DPI。
ScanBitsPerPixel	比特每像素。
ImageBitsPerPixel	所产生图像的比特每像素。如果值为 N，产生的 ABS_IMAGE::ColorCount 是 2 的 n 次幂。

5.2.8. ABS_IMAGE

类 ABS_IMAGE 含有代表滑动指纹样本图像的数据。

函数 ABSCapture、ABSGrab、ABSRawGrab 使用该结构体。发送给 ABS_CALLBACK 的某些消息也能使用它作为附加数据来传输样本图像。

```
typedef struct abs_image {
   ABS_DWORD Width;
   ABS_DWORD Height;
   ABS_DWORD ColorCount;
   ABS_DWORD HorizontalDPI;
   ABS_DWORD VerticalDPI;
   ABS_BYTE ImageData[ABS_VARLEN];
} ABS_IMAGE
```

参数	说明
Width	图像宽度，单位为像素
Height	图像高度，单位为像素
ColorCount	图像的最大颜色数量
HorizontalDPI	图像水平分辨率，单位 DPI
VerticalDPI	图像垂直分辨率，单位 DPI
ImageData[ABS_VARLEN]	所有像素的颜色值。 ImageData 是一个（宽 x 高）字节数组。每个像素用一个字节代表。第一个（宽度）字节代表第一行像素（从左至右），随后一行接一行，没有任何间隔。 每一字节的值表示一个灰阶色。颜色有编号，编号为 0 表示黑色而 1（colorCount - 1）表示白色。其它灰色线性分布于黑色和白色之间。

5.2.9. ABS_PROCESS_DATA

该结构体是一个容器，容纳发送给交互操作回调函数的 ABS_MSG_PROCESS_xxxx 消息关联的附加数据。

注：*部分 ABS_MSG_PROCESS_xxxx 消息使用更具体的结构体，但是所有 ABS_MSG_PROCESS_xxxx 消息都与 ABS_PROCESS_DATA 二进制兼容，即，可以将指向它们的指针安全投射到指向 ABS_PROCESS_DATA 的指针。*

```
typedef struct abs_process_data {
   ABS_DWORD ProcessID;
} ABS_PROCESS_DATA
```

参数	说明
Process ID	进程阶段的 ID。请参考 ABS_PROCESS_xxxx 常量的描述。

5.2.10. ABS_PROCESS_BEGIN_DATA

该结构体是一个容器，容纳发送给交互操作回调函数的 ABS_MSG_PROCESS_BEGIN 消息关联的附加数据。

```
typedef struct abs_process_begin_data {
   ABS_DWORD ProcessID;
   ABS_DWORD Step;
   ABS_DWORD StepCount;
} ABS_PROCESS_BEGIN_DATA
```

参数	说明
Process ID	进程阶段的 ID。请参考 ABS_PROCESS_xxxx 常量的描述。
Step	步骤编号。 部分操作包括多个步骤，例，合并登记时用户必须多次滑动指纹。第一步的标号都是 0。
StepCount	当前进程中子步骤的数量。如果数量未知（例，动态登记场景），参数值为 0。

5.2.11. ABS_PROCESS_PROGRESS_DATA

该结构体是一个容器，容纳发送给交互操作回调函数的 ABS_MSG_PROCESS_PROGRESS 消息关联的附加数据。

```
typedef struct abs_process_progress_data {
   ABS_DWORD ProcessID;
   ABS_DWORD Percentage;
} ABS_PROCESS_PROGRESS_DATA
```

参数	说明
Process ID	进程阶段的 ID。请参考 ABS_PROCESS_xxxx 常量的描述。
Percentage	决定进程完成的百分比。取值范围 0 - 100。如果操作进程不适宜用百分比衡量，参数值为 FFFFFFFFh。

5.2.12. ABS_PROCESS_SUCCESS_DATA

该结构体是一个容器，容纳发送给交互操作回调函数的 ABS_MSG_PROCESS_SUCCESS 消息关联的附加数据。

```
typedef struct abs_process_success_data {
   ABS_DWORD ProcessID;
   ABS_IMAGE* SampleImage;
   ABS_BIR* Template;
} ABS_PROCESS_SUCCESS_DATA
```

参数	说明
Process ID	进程阶段的 ID。请参考 ABS_PROCESS_xxxx 常量的描述。
SampleImage	指针，指向扫描的图像。 如果消息没有关联图像，值为 NULL。
模板	指向所处理模板的指针。 如果消息没有关联模板，值为 NULL。

5.2.13. ABS_NAVIGATION_DATA

该结构体是一个容器，容纳发送给交互操作回调函数的 ABS_MSG_NAVIGATE_CHANGE 消息相关联的附加数据。

```
typedef struct abs_navigation_data {
   ABS_LONG DeltaX;
   ABS_LONG DeltaY;
   ABS_BOOL FingerPresent;
} ABS_NAVIGATION_DATA
```

参数	说明
DeltaX	更改虚拟指针的 X 轴值。
DeltaY	更改虚拟指针的 Y 轴值。
FingerPresent	传感器上有手指按着，参数值为 ABS_TRUE；否则，参数值为 ABS_FALSE。

5.2.14. ABS_DEVICE_LIST_ITEM

设备信息列表的项目。

```
typedef struct abs_device_list_item {
   ABS_CHAR DsnSubString[260];
   ABS_BYTE reserved[256];
} ABS_DEVICE_LIST_ITEM
```

参数	说明
DsnSubString[260]	字符串，被 ABSOpen 用作 DSN 的一部分以连接设备。
Reserved[256]	保留为将来所用。

5.2.15. ABS_DEVICE_LIST

ABSEnumerateDevices 的返回数据，包含所有被列举设备的信息

注：ABSEnumerateDevices 真正的输出参数的长度可变–数组 List[] 列有 NumDevices 个项目。

```
typedef struct abs_device_list {
   ABS_DWORD NumDevices;
   ABS_DEVICE_LIST_ITEM List[ABS_VARLEN];
} ABS_DEVICE_LIST
```

参数	说明
NumDevices	列表里设备的数量
List[ABS_VARLEN]	设备列表

5.2.16. ABS_CALLBACK

一种回调函数，应用可将其提供给 BSAPI，为用户显示 GUI 状态信息。

它通过 ABS_OPERATION 传入交互操作的 BSAPI 函数。执行操作时，交互操作重复调用该回调函数。然后，应用可以根据操作的执行阶段做出回应并更新用户接口。

注：*可以用 ABS_OPERATION 的成员标识位进一步确定回调函数的实际调用方法。*

大多数应用按以下方法实施回调：生物识别操作的第一条消息出现时创建一个对话框，然后根据收到的消息更新对话框里的文本和/或图像。如果消息接收太快，前面的消息还没来得及显示就被后面的消息取代了，用户可能错过消息。为了防止这种情况，BSAPI.DLL 会显示正在发送的消息。例，如果用户没有按正确方式滑动指纹时，产生的图像质量差，可以用适当的反馈消息调用回调函数。再次调用该回调函数就会有时延，提示用户的新指纹质量差，并给用户一定时间查看劣质反馈。

但是，这种默认行为在某些情景中并不适用。例，如果回调函数执行程序把每一条消息写入新一行，用户就可以检查完整的消息历史；或者回调函数执行程序中不给用户提供任何反馈。这时，相继发送的消息之间仅延长生物识别操作所需时间。如需停止所有时延，请设置 ABS_OPERATION::Flags 的 ABS_OPERATION_FLAG_LL_CALLBACK。

ABS_OPERATION_FLAG_USE_IDLE 是 ABS_CALLBACK 相关的，ABS_OPERATION 支持的第二个标识位。设置后，BSAPI.DLL 一定会经常调用该回调函数（大约 100 毫秒）。如果没有内容需要上报，则发送 ABS_MSG_IDLE 消息。如此，用户可以合理地用回调函数取消操作（更多信息，请参见 ABSCancelOperation）。但是，这种做法也有不足之处：占用 CPU 周期。所以，如非必要（例，用户知道不需要取消操作或者可以从其他线程取消操作），应避免使用该标识位。

未设置标识位 ABS_OPERATION_FLAG_USE_IDLE 时，表示从不使用 ABS_MSG_IDLE 消息，回调函数的连续两次调用的间隔时间较长，例，生物识别操作已经开始，用户却长时间不触摸传感器。

```
void ABS_CALLBACK(
   IN const ABS_OPERATION *pOperation
   IN ABS_DWORD dwMsgID
   IN void *pMsgData
)
```

参数	说明
pOperation	调用交互操作时指向 ABS_OPERATION 结构体的指针。 交互操作的调用者可以使用结构体的成员 Context 将数据传入回调函数。
dwMsgData	消息 ID。请参见常量 ABS_MSG_xxxx 的描述。
pMsgData	指针，指向有关消息附加信息数据。 含义与具体消息相关。请参考具体 ABS_MSG_xxxx 常量的文档。

6.0. 具体常量

6.1. ABSInitializeEx 的标识位（ABS_INIT_FLAG_xxxx）

以下标识位可以用于函数 ABSInitializeEx：

ABS_INIT_FLAG_NT_SERVICE 1 以 Windows NT 服务兼容的方式初始化一个库。 只有 MS Windows 才支持这种方式。只有应用作为 NT 服务运行时，才能使用该标识位。 注：该标识位不与 ABS_INIT_FLAG_FORCE_REMOTE_SENSOR 共用。如果使用了 ABS_INIT_FLAG_NT_SERVICE，无论是否使用了 ABS_INIT_FLAG_FORCE_LOCAL_SENSOR 标识位，只能打开本地设备。
ABS_INIT_FLAG_FORCE_LOCAL_SENSOR 2h 强制 BSAPI 忽略远程会话并且总是以本地方式打开传感器。

6.2. ABS_OPERATION 的标识位（ABS_OPERATION_FLAG_xxxx）

以下标识位可以用于结构体 ABS_OPERATION：

ABS_OPERATION_FLAG_LL_CALLBACK 1 打开低级回调模式。 关于低级回调模式和默认高级回调模式的区别，请参考 ABS_CALLBACK 的文档。
ABS_OPERATION_FLAG_USE_IDLE 2 为回调操作启用发送 ABS_MSG_IDLE 消息的功能。 默认不发送无用消息。如果启用了这些功能，短时间内就可以从回调操作有效调用 ABSCancelOperation。 如非必要，请避免发送无用消息。

6.3. 生物识别和抓图函数的标识位（ABS_FLAG_xxxx）

以下标识位可以用于生物识别函数：

注：*所有生物识别函数接受下列所有标识位。更多细节，请参考各个函数的文档。*

ABS_FLAG_NOTIFICATION 1 启用生物识别操作的通知模式。 在通知模式下，用户滑动指纹前，生物识别函数不会有给用户任何反馈，避免干扰后台运行的应用。 GUI 会话是否可见或是否被 ABS_MSG_DLG_SHOW 和 ABS_MSG_DLG_HIDE 消息控制。 只有 ABSCapture 和 ABSVerify 函数支持该标识位。
ABS_FLAG_AUTOREPEAT 2 启用生物识别操作的自动重复模式。 如果启用了该模式，用户指纹不匹配模板时自动重启验证操作。 只有 ABSVerify 函数支持该标识位。更多细节，请参考此函数的文档。
ABS_FLAG_STRICT_PROFILE 4 需要对原始抓图配置文件进行严格解释。 如果设置了该标识位并且因为 FM 不支持而导致不符合配置文件数据要求，原始抓图操作失败并返回 ANS_STATUS_NOT_SUPPORTED。 如果没有设置该标识位，后面跟随的配置文件要视设备支持情况而定。即自动忽略设备不支持的配置数据，操作继续进行 *注：配置文件密钥 ABS_PKEY_IMAGE_FORMAT 总是被严格解释。* 只有 ABSRawGrab 函数支持该标识位。
ABS_FLAG_HIGH_RESOLUTION 8 需要较高的样本图像分辨率。 只有 ABSGrab 函数支持该标识位。使用带有该标识位的 ABSGrab 函数可以取代 ABSRawGrab、 ABSGrabImage 或 ABSRawGrabImage 函数，为设备准确指定要求的图像格式。 对于 UPEK 生产的大部分设备，如果设置了标识位，设备支持的情况下可以使用 508 x 508 DPI；如果没有设置标识位，使用 381 x 381 DPI。

6.4. 模板用途常量（ABS_PURPOSE_xxxx）

本节描述了指纹模板（BIR）中可能用到的值。

将用途作为参数之一的生物识别函数可以利用该信息优化操作处理。例，登记通常要求更高级的模板质量，因此指定 ABS_PURPOSE_ENROLL 后用于确定模板质量的内置生物识别测试会更严格。

请注意此处定义的常量对应 BioAPI 定义的常量（BioAPI_PURPOSE_xxxx）。还请注意 BSAPI 支持的用途是 BioAPI 里定义的所支持用途的子集。

ABS_PURPOSE_UNDEFINED 0 没有指定用途。 没有为任何特定 BIR 用途优化生物识别操作。
ABS_PURPOSE_VERIFY 1 BIR 的用途是验证。
ABS_PURPOSE_ENROLL 3 BIR 的用途是登记。

6.5. ABS_PROFILE_DATA 的密钥常量（ABS_PKEY_xxxx）

常量 ABS_PKEY_xxxx 是成员 Key 可能的值，针对：

ABS_PKEY_WAIT_FOR_ACCEPTABLE 1 关闭 ABSRawGrab 默认的“一次尝试”方法。 值为 0（默认）时，ABSRawGrab 让用户仅仅刷一次指纹。如果因为质量检查结果不满意而不能获取图像，此函数仍返回 ABS_STATUS_OK，而 NULL 是通过输出参数 ppImage 进行传递。 设置为非 0 值时，抓图操作需要用户刷多次直到获取的图像能通过质量检查。 注：当质量检查被关闭时，（即，ABS_PKEY_SCAN_QUALITY_QUERY 和 ABS_PKEY_IMAGE_QUALITY_QUERY 都没有设置为 0），该标识位没有任何影响，因为任意刷一次指纹都会被认为是可接受的。
ABS_PKEY_SCAN_QUALITY_QUERY 2 用于设置扫描质量检查的模式。 设置为非 0（默认）时，将忽略滑动指纹过程中的扫描质量问题，因而它们不会被发送给回调函数。仍可以通过 ABS_SWIPE_INFO 结构体获取有关扫描质量的信息。 设置为 0 时，用 ABS_MSG_QUALITY_xxxx 消息将扫描质量问题发送给回调函数。
ABS_PKEY_IMAGE_QUALITY_QUERY 3 用于设图像质量检查模式。 设置为非 0（默认）时，将忽略滑动指纹过程中的图像质量问题，因而它们不会被发送给回调函数。仍可以通过 ABS_SWIPE_INFO 结构体获取有关图像质量的信息。 设置为 0 时，用 ABS_MSG_QUALITY_xxxx 消息将图像质量问题发送给回调函数。
ABS_PKEY_ALLOW_HW_SLEEP 4 启用 HW 睡眠模式。 设置为非 0（默认）时，将在原始抓图操作中启用设备的 HW 睡眠模式。 设置为 0 时，将停用睡眠模式。 注： 即使在严格模式下，SONLY 也会忽略该配置文件密钥，因为 SONLY 有自己更复杂的策略来确定什么时候启用睡眠模式。
ABS_PKEY_IMAGE_FORMAT 5 指定需要的图像格式。 值可能是任意 ABS_PVAL_IFMT_xxxx 常量。 注：不同的设备支持不同的图像格式。如果使用了设备不支持的图像格式，将导致 ABSRawGrab 执行失败，进而返回 ABS_STAUS_NOT_SUPPORTED，不管严格模式是否在用。

ABS_PKEY_REC_TERMINATION_POLICY 指定图像的重构终止策略。 可能是任意 ABS_PVAL_RTP_xxxx 常量。它决定 FM 停止扫描指纹的时间。 默认值与 FM 型号相关： • TFM 2.0:ABS_PVAL_RTP_CORE • ESS 2.1:ABS_PVAL_RTP_CORE • ESS 2.2:ABS_PVAL_RTP_CORE_PLUS • SONLY:ABS_PVAL_RTP_CORE_PLUS • TCD 50:ABS_PVAL_RTP_FINGERTIP	6
ABS_PKEY_REC_RETUNING 启用传感器自动重调。 设置为非 0（默认）时，将启用传感器自动校准调整的功能，同时等待指纹以获取最好的图像。 设置为 0 时，将停用校准调整。	7
ABS_PKEY_REC_DIGITAL_GAIN 该值用于数字图像增强。 该值决定数字图像增强的一个因素。强烈建议保留参数原有设置。 只有 TFM 2.0 和 ESS 2.1 支持。	8
ABS_PKEY_REC_FLAG_DGAIN 9 该值用于数字图像增强。 设置为非 0（默认）时，将启用数字增益增强；如果设置为 0，则停用数字增益增强。 只有 TCD 50 支持。	
ABS_PKEY_REC_FLAG_SRA_DOWN 10 启用从上到下的条纹消除算法。 设置为非 0（默认）时，启用该算法。设置为 0 时，停用该算法。	
ABS_PKEY_REC_FLAG_SRA_UP 11 启用从下到上的条纹消除算法。 设置为非 0（默认）时，启用该算法。设置为 0 时，停用该算法。	
ABS_PKEY_REC_FLAG_SKEW 12 启用色偏补偿算法。 设置为非 0（默认）时，启用该算法。设置为 0 时，停用该算法。 只有 TCD 50 支持。	

ABS_PKEY_REC_FLAG_GRADIENT 13 启用坡度折减算法。 设置为非 0（默认）时，启用该算法。设置为 0 时，停用该算法。 只有 TCD 50 支持。
ABS_PKEY_REC_SWIPE_DIRECTION 14 指定滑动方向模式。 可能被设置成任意 ABS_PVAL_SWIPEDIR_xxxx 常量。默认值为 ABS_PVAL_SWIPEDIR_STANDARD。如果设置成默认值以外的任意值，也应该把 ABS_PKEY_SCAN_QUALITY_QUERY 设置为 0。 TFM 2.0 和 ESS 2.1 不支持。
ABS_PKEY_REC_NOISE_ROBUSTNESS 15 指定噪音强度模式。 可能被设置成任意 ABS_PVAL_NOIR_xxxx 常量。对于 SONLY，默认值是 ABS_PVAL_NOIR_DISABLED；对于 TCD 50，默认值是 ABS_PVAL_NOIR_ON_DETECTION。 只有 SONLY 和 TCD 50 支持。
ABS_PKEY_REC_NOISE_ROBUSTNESS_TRIGGER 16 指定噪声对抗触发器。 它决定多少次连续的不合格滑动指纹动作将触发噪声对抗。0 意味着不合格滑动指纹动作不会触发噪声对抗。默认值为 3。 只有 TCD 50 支持。
ABS_PKEY_REC_SWIPE_TIMEOUT 17 滑动终止的超时时间，单位是毫秒。 如果超时作废，将终止图像重构。然而，重构的图像仍将进入下一处理阶段，决定它的质量。 默认的超时时间为 6000 毫秒。 TFM 2.0 和 ESS 2.1 不支持。
ABS_PKEY_REC_NO_MOVEMENT_TIMEOUT 18 因为没有动作而超时。 如果指定时间段（单位：毫秒）内没有检测到动作，即使发现了手指，也将终止滑动指纹。设置为 0 时，停用该功能。 默认为 500 毫秒。 TFM 2.0 和 ESS 2.1 不支持。

ABS_PKEY_REC_NO_MOVEMENT_RESET_TIMEOUT 19 因为没有动作复位而超时。 如果指定时间段（单位：毫秒）内没有检测到动作并且图像很短，将不再重启重构。设置为 0 时，停用该功能。 默认为 2000 毫秒。 TFM 2.0 和 ESS 2.1 不支持。
ABS_PKEY_SENSOR_SECURITY_MODE 20 传感器安全模式。 指定是否（及如何）在 FM 传感器和芯片组或处理通信数据的计算机之间发送加密的通信数据。 可能被设置成任意 ABS_PVAL_SSM_xxxx 常量。 对于 SONLY，默认模式是 ABS_PVAL_SSM_ENCRYPT（传感器发送数据给计算机）；对于其他 FM 型号（数据存储在 FM 设备里），默认模式是 ABS_PVAL_SSM_DISABLED。 只有 SONLY 和 TCD 50 支持。

6.6. ABS_PKEY_IMAGE_FORMAT Values (ABS_PVAL_IFMT_xxxx)

可能的图像格式。

请注意，总是按照原始抓图配置文件的严格模式评估需要的图像格式。

注：*符号常量的名字包含符合格式要求的边际参数：水平和垂直分辨率，单位为每英寸点数 DPI、位数每像素 BPP（它们决定所产生样图的颜色素）。*

无论图像格式如何，结构体 ABS_IMAGE 总是使用 1 字节每像素的 BPP。（FM 设备和计算机通信期间，图像压缩较多。）

一些符号常量的名字中，图像格式包含“BINARIZED”的字样。这时，设备上的图像是二进制图像（SONLY 例外），通信速度更快。

注：*特定图像格式是否可用与具体的固件版本和设备校准情况有关。小功率模式尤其需要校准。下表列出了各型号 FM 支持的图像格式，仅用作基本导引。*

ABS_PVAL_IFMT_381_381_8 2 用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取指纹图像，381 x 381 DPI，8 位/像素。 TFM、ESS、SONLY 和 TCD 50 支持。
ABS_PVAL_IFMT_254_254_8 3 用 1:2 子采样（每 4 个像素缩减成 3 个像素）方法抓取指纹图像，254 x 254 DPI，8 位/像素。 TFM 和 ESS 支持。
ABS_PVAL_IFMT_381_381_8_BINARIZED 4 用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取指纹图像，381 x 381 DPI，8 位/像素，转化为二进制后是 1 位/像素。 TFM、ESS、SONLY 和 TCD 50 支持。
ABS_PVAL_IFMT_508_254_8 5 Y 轴上用 1:2 子采样方法抓取整个指纹图像，508 x 254 DPI，8 位每像素。 ESS 和 TCD 50 支持。
ABS_PVAL_IFMT_508_508_4 6 以 508 DPI，8 位每像素的全分辨率抓取整个指纹图像，然后缩减成 4 位每像素。 ESS 和 TCD 50 支持。
ABS_PVAL_IFMT_381_381_4 7 以 3:4 子采样的方法抓取整个指纹图像，381 x 381 DPI，8 位每像素，然后缩减成 4 位每像素。 ESS 和 TCD 50 支持。

ABS_PVAL_IFMT_508_254_4 8 Y 轴上以 1:2 子采样的方法抓取整个指纹图像，508 x 254 DPI，8 位每像素，然后缩减成 4 位每像素。 ESS 和 TCD 50 支持。
ABS_PVAL_IFMT_254_254_4 9 以 1:2 子采样的方法抓取整个指纹图像，254 x 254 DPI，8 位每像素，然后缩减成 4 位每像素。 只有 ESS 支持。
ABS_PVAL_IFMT_508_508_8_WIDTH208 10 抓取位于中央、大小为 208 x 288 像素的窗口，采用全分辨率 508 x 508 DPI，8 位每像素。 只有 TFM 和 ESS 支持。
ABS_PVAL_IFMT_508_508_8_COMPRESS1 11 以 508 DPI，8 位每像素的全分辨率抓取整个指纹图像。如果设备支持，建议尽量选用 ABS_PVAL_IFMT_508_508_8_COMPRESS2。 只有 ESS 支持。
ABS_PVAL_IFMT_508_508_4_SCAN4 12 以 508 DPI 的全分辨率、4 位扫描模式抓取整个指纹图像。该模式的能耗低，但图像质量也低。 只有早于 2.1 rev.K 的 ESS 才支持。
ABS_PVAL_IFMT_381_381_8_FAST 13 用 3:4 子采样方法抓取整个指纹图像，381 x 381 DPI，8 位每像素。这种模式内部采用 508 x 254 扫描方法，扫描指纹的速度快，但图像质量低。 ESS 和 TCD 50 支持。
ABS_PVAL_IFMT_508_254_4_SCAN4 14 Y 轴上用 1:2 子采样方法、4 位扫描模式抓取整个指纹图像，508 x 254 DPI。该模式的能耗低，但图像质量也低。 只有早于 2.1 rev.K 的 ESS 才支持。
ABS_PVAL_IFMT_254_254_4_SCAN4 15 用 1:2 子采样方法、4 位扫描模式抓取整个指纹图像，254 x 254 DPI。该模式的能耗低，但图像质量也低。 只有 ESS 支持。

ABS_PVAL_IFMT_381_381_4_FAST 16 以 3:4 子采样的方法抓取整个指纹图像，381 x 381 DPI，8 位每像素，然后缩减成 4 位每像素。这种模式内部采用 508 x 254 扫描方法，扫描指纹的速度快，但图像质量低。 ESS 和 TCD 50 支持。
ABS_PVAL_IFMT_381_381_8_BINARIZED_FAST 17 以 3:4 子采样的方法抓取整个指纹图像，381 x 381 DPI，8 位每像素，然后缩减成 1 位每像素。这种模式内部采用 508 x 254 扫描方法，扫描指纹的速度快，但图像质量低。 ESS 和 TCD 50 支持。
ABS_PVAL_IFMT_508_508_8_COMPRESS2 18 以 508 DPI，8 位每像素的全分辨率抓取整个指纹图像。 ESS 2.2 支持。
ABS_PVAL_IFMT_381_381_8_SCAN381 19 用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取整个指纹图像，381 x 381 DPI，8 位/像素。传感器用本身的 381 扫描格式直接进行子采样。 带 TCS3C 的 ESS 2.2，较新的传感器，以及 TCD 50 支持。
ABS_PVAL_IFMT_381_381_4_SCAN381 20 以 3:4 子采样的方法抓取整个指纹图像，381 x 381 DPI，8 位每像素，然后缩减成 4 位每像素。传感器用本身的 381 扫描格式直接进行子采样。 带 TCS3C 的 ESS 2.2，较新的传感器，以及 TCD 50 支持。
ABS_PVAL_IFMT_381_381_8_BINARIZED_SCAN381 21 用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取指纹图像，381 x 381 DPI，8 位/像素，转化为二进制后是 1 位/像素。传感器用本身的 381 扫描格式 直接进行子采样。 带 TCS3C 的 ESS 2.2，较新的传感器，以及 TCD 50 支持。
ABS_PVAL_IFMT_381_381_8_LP 22 低能耗模式下，用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取指纹图像，381 x 381 DPI，8 位/像素。 ESS 2.2 和 TCD 50 支持。

ABS_PVAL_IFMT_381_381_4_LP 23 低能耗模式下，以 3:4 子采样的方法抓取整个指纹图像，381 x 381 DPI，8 位每像素，然后缩减成 4 位每像素。 ESS 2.2 和 TCD 50 支持。
ABS_PVAL_IFMT_381_381_8_BINARIZED_LP 24 低能耗模式下，用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取指纹图像，381 x 381 DPI，8 位/像素，转化为二进制后是 1 位/像素。 ESS 2.2 和 TCD 50 支持。
ABS_PVAL_IFMT_381_381_8_VLP 25 极低能耗模式下，用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取指纹图像，381 x 381 DPI，8 位/像素。 串行通信速度不高于 75600 kbps 的 ESS 2.2 支持。
ABS_PVAL_IFMT_381_381_4_VLP 26 极低能耗模式下，以 3:4 子采样的方法抓取整个指纹图像，381 x 381 DPI，8 位每像素，然后缩减成 4 位每像素。 串行通信速度不高于 75600 kbps 的 ESS 2.2 支持。
ABS_PVAL_IFMT_381_381_8_BINARIZED_VLP 27 极低能耗模式下，用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取指纹图像，381 x 381 DPI，8 位/像素，转化为二进制后是 1 位/像素。 串行连接速度不高于 57600 kbps 的 ESS 2.2 支持。
ABS_PVAL_IFMT_381_381_8_SCAN381_381_4 28 用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取整个指纹图像，381 x 381 DPI，8 位/像素。内部采用 381/381/4 格式扫描图像。 只有 SONLY 的部分变体支持。
ABS_PVAL_IFMT_381_381_8_BINARIZED_SCAN381_381_4 30 用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取整个指纹图像，381 x 381 DPI，8 位/像素。内部采用 381/381/4 格式扫描图像。 只有 SONLY 的部分变体支持。
ABS_PVAL_IFMT_381_381_8_SCAN381_254_4 31 用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取整个指纹图像，381 x 381 DPI，8 位/像素。内部采用 381/254/4 格式扫描图像。 只有 SONLY 的部分变体支持。

ABS_PVAL_IFMT_381_381_8_BINARIZED_SCAN381_254_4 33 用 3:4 子采样（每 4 个像素缩减成 3 个像素）方法抓取指纹图像，381 x 381 DPI，8 位/像素，转化为二进制后是 1 位/像素。内部采用 381/254/4 格式扫描图像。 只有 SONLY 的部分变体支持。
ABS_PVAL_IFMT_508_508_8_SCAN508_508_8 34 抓取整个指纹图像，508 x 508 DPI，8 位每像素。内部采用 508/508/8 格式扫描图像。 只有 SONLY 的部分变体支持。
ABS_PVAL_IFMT_508_508_4_SCAN508_508_8 35 以全分辨率（508 x 508 DPI）、8 位每像素抓取整个指纹图像，然后缩减成 4 位每像素。内部采用 508/508/8 格式扫描图像。 只有 SONLY 的部分变体和 TCD 50 支持。
ABS_PVAL_IFMT_508_508_8_BINARIZED_SCAN508_508_8 36 以全分辨率（508 x 508 DPI）、8 位每像素抓取整个指纹图像，转化成二进制是 1 位每像素。内部采用 508/508/8 格式扫描图像。 SONLY 的部分变体和 TCD 50 支持。
ABS_PVAL_IFMT_508_508_8_COMPRESS3 39 用全分辨率（508 x 508 DPI）、8 位每像素抓取整个指纹图像，然后缩减成 3:4 子采样格式图（381 x 381 DPI），8 位每像素。 TCD 50 支持。
ABS_PVAL_IFMT_508_254_8_LP 40 TCD 50 支持。
ABS_PVAL_IFMT_508_254_4_LP 41 TCD 50 支持。
ABS_PVAL_IFMT_381_381_8_FAST_LP 42 TCD 50 支持。
ABS_PVAL_IFMT_381_381_4_FAST_LP 43 TCD 50 支持。
ABS_PVAL_IFMT_381_381_8_BINARIZED_FAST_LP 44 TCD 50 支持。

6.7. ABS_PKEY_REC_TERMINATION_POLICY 的□（ABS_PVAL_RTP_xxxx）

更多信息，请参考 ABS_PKEY_REC_TERMINATION_POLICY 的描述。

ABS_PVAL_RTP_BASIC 0 基础图像的重构终止策略。 如果扫描图像长于最大长度，仅返回最开始的图像的开头部分。
ABS_PVAL_RTP_FINGERTIP 1 指尖图像的重构终止策略。 如果扫描图像长于最大长度，返回图像末尾、指尖前的部分。
ABS_PVAL_RTP_CORE 2 核心图像的重构终止策略。 如果扫描图像长于最大长度，返回对生物识别最重要、最有价值的部分。
ABS_PVAL_RTP_CORE_PLUS 3 增强核心图像的重构终止策略。 如果扫描图像长于最大长度，返回对生物识别最重要、最有价值的部分，忽略手指关节部分。 TFM 2.0 和 ESS 2.1 不支持。

6.8. ABS_PKEY_REC_SWIPE_DIRECTION 的值（ABS_PVAL_SWIPEDIR_xxxx）

更多信息，请参考有关 ABS_PKEY_REC_SWIPE_DIRECTION 的描述。

ABS_PVAL_SWIPEDIR_STANDARD 0 标准滑动方向。
ABS_PVAL_SWIPEDIR_INVERTED 1 逆滑动方向。
ABS_PVAL_SWIPEDIR_AUTODETECT 2 滑动开始时的自动检测。
ABS_PVAL_SWIPEDIR_STANDARD_WARN 3 带告警的标准滑动方向。 如果检测到向后滑动，将发送 ABS_MSG_QUALITY_BACKWARD 消息给回调函数。
ABS_PVAL_SWIPEDIR_INVERTED_WARN 4 带告警的逆滑动方向。 如果检测到向后滑动，将发送 ABS_MSG_QUALITY_BACKWARD 消息给回调函数。

6.9. ABS_PKEY_REC_NOISE_ROBUSTNESS 的值（ABS_PVAL_NOIR_xxxx）

更多信息，请参考有关 ABS_PKEY_REC_NOISE_ROBUSTNESS 的描述。

ABS_PKEY_NOIR_DISABLED 0 关闭噪音对抗。
ABS_PKEY_NOIR_FORCED 1 开启噪音对抗。
ABS_PKEY_NOIR_ON_DETECTION 2 开启噪音对抗的自动检测模式。

6.10. ABS_PKEY_SENSOR_SECURITY_MODE 的值（ABS_PVAL_SSM_xxxx）

更多信息，请参考有关 ABS_PKEY_SENSOR_SECURITY_MODE 的描述。

ABS_PVAL_SSM_DISABLED 0 停用传感器安全模式。
ABS_PVAL_SSM_ENCRYPT 1 传感器安全被设置成“加密”模式。
ABS_PVAL_SSM_SIGN_ALL 2 传感器安全被设置成“标记所有”模式。
ABS_PVAL_SSM_SIGN_PARTIAL_V1 3 传感器安全被设置成“标记部分版本.1”。 与版本 2 相比，速度快，安全性低。
ABS_PVAL_SSM_SIGN_PARTIAL_V2 4 传感器安全被设置成“标记部分版本.2”。 与版本 1 相比，速度慢，安全性高。

6.11. 滑动信息标识位（ABS_SWIPE_FLAG_xxxx）

结构体 ABS_SWIPE_INFO 的成员标识位是一个位掩码，描述了 ABSRawGrab 里用户滑动指纹的各种属性。

ABS_SWIPE_FLAG_TOO_FAST 01h 滑动速度过快。 如果用户滑动速度过快，设备不能处理所有数据。
ABS_SWIPE_FLAG_TOO_SKEWED 02h 滑动过于歪斜。
ABS_SWIPE_FLAG_BACKWARDS_MOVEMENT 04h 滑动方向错误。 注： 只有自动检测滑动方向的模式（即，配置文件值 ABS_PKEY_REC_SWIPE_DIRECTION 设置成 ABS_PVAL_SWIPEDIR_AUTODETECT）下，才会设置该标识位。
ABS_SWIPE_FLAG_JOINT_DETECTED 08h 滑动中检测到手指关节。
ABS_SWIPE_FLAG_TOO_SHORT 10h 滑动过于短暂。 用户手指的滑动区域仅有很少的生物识别数据，会导致所生成的生物识别模板质量差。

6.12. 进程常量（ABS_PROCESS_xxxx）

这些常量用于标识交互操作进程。

结构体 ABS_PROCESS_DATA,、ABS_PROCESS_BEGIN_DATA 和 ABS_PROCESS_SUCCESS_DATA 拥有一个成员，它被称为进程（Process），用于标识当前操作所处的阶段。

部分高级生物识别操作包含多个进程。交互操作一般是由进程树组成的。进入和离开进程树的一个节点时，分别发送 ABS_MSG_PROCESS_BEGIN 和 ABS_MSG_PROCESS_END 消息以调用回调函数。

根据相应进程和进度，可选择发送其他 ABS_MSG_PROCESS_xxxx 消息。

例，一个典型的合并登记操作可能包括以下顺序的消息：

1. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_ENROLL)
2. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_CONSOLIDATED_CAPTURE)
3. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_CAPTURE)
4. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_GRAB)
5. ...引导用户正确滑动指纹的消息
6. ABS_MSG_PROCESS_END（ABS_PROCESS_GRAB 的结尾）
7. ABS_MSG_PROCESS_PROGRESS (百分比=23%)
8. ABS_MSG_PROCESS_END （ABS_PROCESS_CAPTURE 的结尾）
9. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_CAPTURE)
10. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_GRAB)
11. ...引导用户正确滑动指纹的消息
12. ABS_MSG_PROCESS_END（ABS_PROCESS_GRAB 的结尾）
13. ABS_MSG_PROCESS_PROGRESS (百分比=32%)
14. ABS_MSG_PROCESS_END （ABS_PROCESS_CAPTURE 的结尾）
15. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_CAPTURE)
16. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_GRAB)
17. ...引导用户正确滑动指纹的消息
18. ABS_MSG_PROCESS_END（ABS_PROCESS_GRAB 的结尾）
19. ABS_MSG_PROCESS_PROGRESS (百分比=39%)
20. ABS_MSG_PROCESS_END （ABS_PROCESS_CAPTURE 的结尾）
21. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_CAPTURE)
22. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_GRAB)
23. ...引导用户正确滑动指纹的消息
24. ABS_MSG_PROCESS_END（ABS_PROCESS_GRAB 的结尾）
25. ABS_MSG_PROCESS_PROGRESS (百分比=64%)
26. ABS_MSG_PROCESS_END（ABS_PROCESS_CAPTURE 的结尾）
27. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_CAPTURE)
28. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_GRAB)
29. ...引导用户正确滑动指纹的消息

30. ABS_MSG_PROCESS_END （ABS_PROCESS_GRAB 的结尾）
31. ABS_MSG_PROCESS_PROGRESS (百分比=88%)
32. ABS_MSG_PROCESS_END （ABS_PROCESS_CAPTURE 的结尾）
33. ABS_MSG_PROCESS_BEGIN (ProcessID = ABS_PROCESS_CAPTURE)
34. ABS_MSG_PROCESS_BEGIN（ProcessID = ABS_PROCESS_GRAB）
35. ...引导用户正确滑动指纹的消息
36. ABS_MSG_PROCESS_END（ABS_PROCESS_GRAB 的结尾）
37. ABS_MSG_PROCESS_PROGRESS (百分比=100%)
38. ABS_MSG_PROCESS_END （ABS_PROCESS_CAPTURE 的结尾）
39. ABS_MSG_PROCESS_END（ABS_PROCESS_CONSOLIDATED_CAPTURE 的结尾）
40. ABS_MSG_PROCESS_END（ABS_PROCESS_ENROLL 的结尾）

ABS_PROCESS_NAVIGATE 1 导航的根进程（ABSNavigate）。 典型地，它包括类型 ABS_PROCESS_CONSOLIDATED_CAPTURE 的一个子进程。
ABS_PROCESS_ENROLL 2 登记的根进程（ABSEnroll）。 典型地，包括一个子进程 ABS_PROCESS_CONSOLIDATED_CAPTURE。
ABS_PROCESS_VERIFY 3 验证的根进程（ABSVerify）。 典型地，包括一个子进程 ABS_PROCESS_CAPTURE。
ABS_PROCESS_IDENTIFY 4 认证的根进程。
ABS_PROCESS_CONSOLIDATED_CAPTURE 5 扫描仪的合并模板的进程。 复杂进程，包括 ABS_PROCESS_CAPTURE 的若干个子进程和一个最终 ABS_PROCESS_CONSOLIDATE。

ABS_PROCESS_CONSOLIDATE 6 合并进程。 将一个手指的几个模板合并成一个高质量的登记模板。 注 : 自 BSAPI 3.5 以后，没有使用过该进程，并且定义的常量仅仅是为了向后兼容，因为定做的 ABS_CALLBACK 执行程序的源码可以参考该常量。
ABS_PROCESS_CAPTURE 7 扫描仪的模板捕获进程。 典型地，包括一个子进程 ABS_PROCESS_GRAB。
ABS_PROCESS_MATCH 8 将一个模板与一套模板进行匹配的进程。
ABS_PROCESS_GRAB 9 扫描仪上抓取样本图像的进程。 它是一个从传感器上抓取指纹样本图像的相对低级的进程。
ABS_PROCESS_NOTIFY 10 通知进程。 用于允许使用通知模式的函数（参见 ABS_FLAGF_NOTIFICATION）。

6.13. 设备属性常量（ABS_DEVPROP_xxxx）

下列常量适用于 dwPropertyId 参数的值。

ABS_DEVPROP_DEVICE_VERSION 0 标识 FM 设备 ROM 的版本。 输出的 ABS_DATA 有 4 个字节，理解成 ABS_DWORD。最高字节表示主版本号，次高字节表示副版本号，而低字节表示子版本号/修订版本号。 • 020 xxx (2.0.x) - 用于 TFM 2.0 • 0401xxxx (4.1.x) - 用于 ESS 2.1 • 0401xxxx (4.2.x) - 用于 ESS 2.2 • 050 xxx (5.0.x) - 用于 TCD 50
ABS_DEVPROP_DEVICE_ID 1 如果 FM 支持，它表示设备的唯一标识。 ABS_DATA 可能含有任意顺序的字节，组成标识。如果 FM 不支持，不会分配 ABS_DATA，但会发送 NULL。
ABS_DEVPROP_FIRMWARE_VARIANT 2 标识固件变体。 输出的 ABS_DATA 有 4 个字节，理解成 ABS_DWORD。
ABS_DEVPROP_SENSOR_TYPE 4 标识传感器的类型。 输出的 ABS_DATA 有 4 个字节，理解成 ABS_DWORD。
ABS_DEVPROP_FUNCTIONALITY 8 提供 FM 的功能信息。 输出的 ABS_DATA 有 4 个字节，理解成位掩码。 Bit 15 决定 FM 为 SONLY（位已设置）或不是 SONLY（位没有被设置）。
ABS_DEVPROP_DSN_STRING 11 提供设备的 DSN。 输出的 ABS_DATA 包括任意个字节，最后一个字节总是 0，理解成零终止的 C 字符串。它能当作 DSN 串用于函数 ABSOpen 和 ABSEnumerateDevices。

ABS_DEVPROP_GUID
12

获取设备的 GUID。

输出的 ABS_DATA 含有设备上存储的一个二进制 GUID。GUID 是在设备第一次重启或将固件加载到 NVM 时产生的，具体与设备类型有关。

注 :
只有 ESS 2.2 和较新的设备支持该功能。

6.14. 会话和全局参数常量（ABS_PARAM_xxxx）

这些常量用于标识每一个参数，可以用作函数 ABSSetSessionParameter 和 ABSGetSessionParameter（针对会话参数），或者 ABSSet-GlobalParameter 和 ABSGetGlobalParameter（针对全局参数）里参数 dwParamID 的值。

ABS_PARAM_CONSOLIDATION_COUNT 1 表示合并的指纹滑动次数。 合并是指登记进程中将多个指纹模板的信息合并成一个高质量模板的进程。它决定了将多少个模板合并成一个到呢估计模板。 设置为 0 时，将使用动态登记。这时，登记操作会一直重复，直到产生的模板符合质量方面的若干个最低要求。 它的值是 1 个字节长的 ABS_DATA。这个字节理解成没有符号的指纹滑动次数。 默认值是 0（即，默认采用动态登记）。 注：当前只支持 0（动态登记）和 5 两个值。
ABS_PARAM_CONSOLIDATION_TYPE 2 决定合并的类型。 即，如何将多个模板合并成一个高质量模板。也请参考有关参数 ABS_PARAM_CONSOLIDATION_COUNT 的描述。 它的值是 1 个字节长的 ABS_DATA，可以是任意常量 ABS_CONSOLIDATION_xxxx。请参见这些常量的描述。
ABS_PARAM_MATCH_LEVEL 3 决定用 ABSVerifyMatch 对比两个模板时要求的安全等级。 它的值是 1 个字节长的 ABS_DATA，关于支持的值列表，请参考 ABS_MATCH_xxxx。
ABS_PARAM_DISABLE_SENSOR_SLEEP 4 停止传感器睡眠。 它的值是 1 个字节长的 ABS_DATA，值为 0 表示 BSAPI 可以将设备切换成省电的睡眠模式。值不为 0 表示停止睡眠模式。
ABS_PARAM_DISABLE_SELECTIVE_SUSPEND 5 可以停止部分挂起的功能。 它的值是 1 个字节长的 ABS_DATA，值为 0 表示可以使用部分挂起的功能。值不为 0 表示停用部分挂起的功能。

ABS_PARAM_POWER_SAVE_MODE 6 如果该常量没有用 ABS_PARAM_DISABLE_SENSOR_SLEEP 完全停止，该常量用于设置默认的电源安全模式。 它的值是 1 个字节长的 ABS_DATA。可能的值是：0—省电模式总是关闭的（耗电多）； 1—省电模式总是打开的（耗电最少，可能会有较长的时延）； 2—根据用户活跃程度自动切换省电模式。
ABS_PARAM_POWER_SAVE_TIMEOUT 7 决定用户接触传感器/使用键盘/使用鼠标后多长时间是用户的活跃时间。 只有 ABS_PARAM_POWER_SAVE_MODE 设置为 2 时，它才能影响电源管理。 它的值是 4 个字节长的 ABS_DATA。该值可以理解成 ABS_DWORD，用于指定秒数。 另见 ABS_PARAM_POWER_SAVE_CHECK_KEYBOARD。
ABS_PARAM_ANTISPOOFING_POLICY 8 它的值是 1 个字节长的 ABS_DATA， 关于支持的值列表，请参考 ABS_ANTISPOOFING_xxxx。
ABS_PARAM_ANTISPOOFING_LEVEL 9 如果使用了反欺算法，该设置决定如何在安全性和用户便利之间取舍平衡。 它的值是 1 个字节长的 ABS_DATA，可能的值是：0—优选便利性（默认）； 1—优选安全性。
ABS_PARAM_OPEN_TOTAL_TIMEOUT 10 打开会话的总计时间超时 如果设备出现某些错误（例，ESD 引起的通信错误），BSAPI 自动尝试为设备恢复通信会话。该参数表示 BSAPI 尝试重开会话的最大时间长度。 它的值是 4 个字节长的 ABS_DATA。该值可以理解成 ABS_DWORD，用于指定毫秒数。默认值为 5000。
ABS_PARAM_OPEN_RETRY_UI_NOTIFY_TIMEOUT 11 为尝试重开会话超时发送的用户界面通知。 如果 BSAPI 尝试重开会话的时间超出了参数值，交互操作的回调函数将收到 ABS_MSG_PROCESS_PROGRESS 消息，通知终端用户设备处于繁忙状态。 它的值是 4 个字节长的 ABS_DATA。该值可以理解成 ABS_DWORD，用于指定毫秒数。默认值为 2000。

ABS_PARAM_OPEN_RETRY_DELAY 12 两个连续的会话重开尝试之间的时延 BSAPI 重复尝试重开一个会话直到成功为止或者直到 ABS_PARAM_OPEN_TOTAL_TIMEOUT 超时。该参数则用于指定两个连续的重开尝试之间的时延。 它的值是 4 个字节长的 ABS_DATA。该值可以理解成 ABS_DWORD，用于指定毫秒数。默认值为 500。
ABS_PARAM_IFACE_VERSION 13 它是一个只读的全局参数，表示 BSAPI 接口的版本号。 它的值是 1 个字节长的 ABS_DATA，当前版本的 BSAPI 使用 V2.0 的接口。 注： 相应版本的 BSAPI.DLL 获取的参数值总是一样的。
ABS_PARAM_POWER_SAVE_CHECK_KEYBOARD 14 该全局参数表示键盘和鼠标活动是否算作用户活动。 它的值是 4 个字节长的 ABS_DATA。值为 0 表示键盘和鼠标活动不算作用户活动，只有用户接触传感器时才影响电源管理。值非 0 表示键盘和鼠标活动算作用户活动。 Window 系统下，默认值是 1，除非 BSAPI 运行在 NT 服务兼容模式下（即，使用设置了 ABS_INIT_FLAG_NT_SERVICE 标识位的 ABSInitializeEx 函数对 BSAPI 进行了初始化）。如果 BSAPI 运行在 NT 服务兼容模式下，默认值是 0。 注： 如果 BSAPI 在 Windows 系统上运行，并且该参数设置为 1，则只在有效的用户会话场景下检测用户活动。如果进程不在有效的用户会话场景下运行，不检测用户的键盘和鼠标操作。 BSAPI 运行在其他系统上时，默认值是 0 而且不支持设置参数值。 另见 ABS_PARAM_POWER_SAVE_MODE 和 ABS_PARAM_POWER_SAVE_TIMEOUT。
ABS_PARAM_LATENT_CHECK 16 该全局参数表示是否隐式执行潜指纹检测。 它的值是 1 个字节长的 ABS_DATA。值为 0 表示关闭潜指纹检测；值为 1（默认）表示打开潜指纹检测，调用 ABSEnroll 或 ABSVerify 时都会在区域传感器上进行隐式的潜指纹检测。 再调用其它函数（例，ABSGrab 或 ABSCapture）后，根据应用要求手动调用函数 ABSCheckLatent 进行检测。 关于潜指纹检测的更多信息，请参考 3.1.5 小节。

6.15. 参数 ABS_PARAM_CONSOLIDATION_TYPE 的值（ABS_CONSOLIDATION_xxxx）

如下常量是可能的参数值：

ABS_CONSOLIDATION_NORMAL 0 使用一般的合并算法。 基于收集到的模板的子集或提供的所有模板中的一个（最好的一个）构建登记模板。内置启发式算法会决定具体采用哪种方法。
ABS_CONSOLIDATION_CONVENIENT 1 使用便捷的合并算法。 类似于 ABS_CONSOLIDATION_NORMAL 策略，放宽了图像/模板进入登记模板过程的标准。
ABS_CONSOLIDATION_STRICT 2 使用严格的合并算法。 类似于 ABS_CONSOLIDATION_NORMAL 策略，但要求所有收集的模板必须相互匹配（即，所有采集过程使用同一根手指）。

6.16. 参数 ABS_PARAM_MATCH_LEVEL 的值（ABS_MATCH_xxxx）

如下常量是可能的参数值：

ABS_MATCH_MIN_SECURITY 1 最低安全设置。
ABS_MATCH_LOWER_SECURITY 2 较低安全设置。
ABS_MATCH_MEDIUM_SECURITY 3 中等安全设置。
ABS_MATCH_HIGHER_SECURITY 4 较高安全设置。
ABS_MATCH_MAX_SECURITY 5 最高安全设置。

6.17. 参数 ABS_PARAM_ANTISPOOFING_POLICY 的值（ABS_ANTISPOOFING_xxxx）

如下常量是可能的参数值：

ABS_ANTISPOOFING_DISABLED 0 指纹传感器上的反欺检查是明确关闭的。这是默认值。
ABS_ANTISPOOFING_AUTODETECT 1 如果设备支持，反欺检查是明确打开的。
ABS_ANTISPOOFING_DEVICE_DEFAULT 2 无论如何也接触不到反欺设置，因此保留默认设置（与设备相关）。

6.18. 回调消息代码（ABS_MSG_xxxx）

这些代码可以设置成 ABS_CALLBACK 的参数 dwMsgID 的值。

回调函数能对各种消息作出相应反应，通常在一个对话框里显示/更新若干条消息。具体的 ABS_MSG_xxxx 值也决定了回调函数的 pMsgData 参数的含义。

消息分类如下：

- 进程消息（ABS_MSG_PROCESS_xxxx）。决定完整的交互操作的生命周期。每一个进程的开始和结束是用 ABS_MSG_PROCESS_BEGIN 和 ABS_MSG_PROCESS_END 划分的。在这两者之间，也可能有其他消息（包括嵌套子进程）。如需了解更多关于操作生命周期的信息，请参考有关 ABS_PROCESS_xxxx 常量的描述。
- 提示消息（ABS_MSG_PROMPT_xxxx）。回调函数应提示用户如何操作 FM 传感器，例，扫描手指，或者把手指从传感器上拿开。
- 质量反馈消息（ABS_MSG_QUALITY_xxxx）。通知用户与传感器的互动不符合要求。根据交互操作的性质，这种情况可能会导致进程重复，提示用户重新操作。
- 导航消息（ABS_NAVIGATE_xxxx）。只有导航时才会调用导航消息（参考 ABSNavigate 的描述）。

下表列出了所有被支持的消息代码。

ABS_MSG_PROCESS_BEGIN 11000000h 表示交互操作开始了新的进程阶段。 该消息与 ABS_MSG_PROCESS_END 一起定义一个交互操作生命周期的框架。 pMsgData 指向结构体 ABS_PROCESS_BEGIN_DATA 存储的附加数据。
ABS_MSG_PROCESS_END 12000000h 表示交互操作结束了进程阶段。 该消息与 ABS_MSG_PROCESS_BEGIN 一起定义一个交互操作生命周期的框架。如需了解更多关于操作生命周期的信息，请参考有关 ABS_PROCESS_xxxx 常量的描述。 pMsgData 指向结构体 ABS_PROCESS_DATA 存储的附加数据。
ABS_MSG_PROCESS_SUSPEND 13000000h 表示交互操作已经被挂起。 其他同等或更高优先级的操作占用传感器时会发送该消息，占用结束后会恢复原交互操作。 pMsgData 指向结构体 ABS_PROCESS_DATA 存储的附加数据。
ABS_MSG_PROCESS_RESUME 14000000h 表示已恢复交互操作。 pMsgData 指向结构体 ABS_PROCESS_PROGESS_DATA 存储的附加数据。

ABS_MSG_PROCESS_PROGRESS 15000000h 通知用户操作正在处理中。 pMsgData 指向结构体 ABS_PROCESS_DATA 存储的附加数据。
ABS_MSG_PROCESS_SUCCESS 16000000h 通知用户进程成功。 如果发送了该消息，它是 ABS_MSG_PROCESS_END 前的最后一条消息。根据特定进程的性质，它可以在 ABS_MSG_PROCESS_END 前使用 ABS_MSG_PROCESS_SUCCESS 或 ABS_MSG_PROCESS_FAILURE，但有些进程可能两者都不调用。 pMsgData 指向结构体 ABS_PROCESS_SUCCESS_DATA 存储的附加数据。
ABS_MSG_PROCESS_FAILURE 17000000h 通知用户进程失败。 如果发送了该消息，它是 ABS_MSG_PROCESS_END 前的最后一条消息。根据特定进程的性质，它可以在 ABS_MSG_PROCESS_END 前使用 ABS_MSG_PROCESS_SUCCESS 或 ABS_MSG_PROCESS_FAILURE，但有些进程可能两者都不调用。 pMsgData 指向结构体 ABS_PROCESS_DATA 存储的附加数据。
ABS_MSG_PROMPT_SCAN 21000000h 回调函数须提示用户滑动指纹。 pMsgData 总是 NULL。
ABS_MSG_PROMPT_TOUCH 22000000h 回调函数须提示用户接触传感器。 pMsgData 总是 NULL。
ABS_MSG_PROMPT_KEEP 23000000h 回调函数须提示用户保持将手指放置在传感器上的状态。 pMsgData 总是 NULL。
ABS_MSG_PROMPT_LIFT 24000000h 回调函数须提示用户将手指从传感器上抬起。 pMsgData 总是 NULL。
ABS_MSG_PROMPT_CLEAN 25000000h 回调函数须提示用户清理传感器。 pMsgData 总是 NULL。

ABS_MSG_QUALITY 30000000h 滑动质量低。 注： 如果可行，BSAPI 会发送更详细的 ABS_MSG_QUALITY_xxxx 消息。没有其它合适的更详细的消息时，才会发送该消息。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_CENTER_HARDER 31000000h 滑动质量低。用户应将手指放在传感器的中央并用力按。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_CENTER 31100000h 滑动质量低。用户应将手指放在传感器的中央。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_LEFT 31110000h 滑动质量低。滑动过左。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_RIGHT 31120000h 滑动质量低。滑动过右。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_HIGH 31130000h 滑动质量低。滑动过高。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_LOW 31140000h 滑动质量低。滑动过低。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_HARDER 31200000h 用户按指纹时应更用力。 pMsgData 总是 NULL。

ABS_MSG_QUALITY_TOO_LIGHT 31210000h 滑动质量低。滑动用力太小。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_DRY 31220000h 滑动质量低。手指过于干燥。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_SMALL 31230000h 滑动质量低。滑动面积过小。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_SHORT 32000000h 滑动质量低。滑动时间过短。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_FAST 33000000h 滑动质量低。滑动过快。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_SKEWED 34000000h 滑动质量低。滑动过斜。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_TOO_DARK 35000000h 滑动质量低。指纹过暗。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_BACKWARD 36000000h 滑动质量低。后向滑动。 pMsgData 总是 NULL。
ABS_MSG_QUALITY_BACKWARD 36000000h 滑动质量低。后向滑动。 pMsgData 总是 NULL。

ABS_MSG_QUALITY_JOINT 37000000h 滑动质量低。检测到关节。 pMsgData 总是 NULL。
ABS_MSG_NAVIGATE_CHANGE 41000000h 通知用户导航变化（用户已经移动了手指，接触了传感器或者拿开了手指）。导航操作中才会使用。 pMsgData 指向结构体 ABS_NAVIGATION_DATA。
ABS_MSG_NAVIGATE_CLICK 42000000h 告知用户用手指单击了传感器。导航操作中才会使用。 pMsgData 总是 NULL。
ABS_MSG_DLG_SHOW 51000000h 告知需要显示反馈对话框。 pMsgData 总是 NULL。
ABS_MSG_DLG_HIDE 52000000h 告知需要隐藏反馈对话框。 pMsgData 总是 NULL。
ABS_MSG_IDLE 0h 特殊消息，给回调函数提供一次取消交互操作的机会。pMsgData 总是 NULL。 注： 只有结构体 ABS_OPERATION 指定了 ABS_OPERATION_FLAG_USE_IDLE 标识位时，才会用到该消息。 这样，即使是单线程应用也能取消交互操作。

7.0. 定义的结果代码列表

下列结果代码是 PT_STATUS 可能返还的值。

成功返回状态。
ABS_STATUS_OK (0)

一般、未知、或不明确的错误。
ABS_STATUS_GENERAL_ERROR (-5001)

内部错误。
ABS_STATUS_INTERNAL_ERROR (-5002)

BSAPI 已初始化。
ABS_STATUS_ALREADY_INITIALIZED (-5003)

BSAPI 未初始化。
ABS_STATUS_NOT_INITIALIZED (-5004)

已打开连接。
ABS_STATUS_ALREADY_OPENED (-5005)

无效参数。
ABS_STATUS_INVALID_PARAMETER (-5006)

无效（连接）句柄。
ABS_STATUS_INVALID_HANDLE (-5007)

没有发现该类设备。
ABS_STATUS_NO_SUCH_DEVICE (-5008)

因为超时，操作已被中断。
ABS_STATUS_TIMEOUT (-5009)

请求的特性/函数还没有实施。
ABS_STATUS_NOT_IMPLEMENTED (-5010)

请求的特性/函数不被支持。
ABS_STATUS_NOT_SUPPORTED (-5011)

操作已被取消。
ABS_STATUS_CANCELED (-5012)

未找到操作（无效的操作 ID
或操作已完成）。
ABS_STATUS_NO_SUCH_OPERATION (-5013)

8.0. 版本 3.5 的新特性

8.1. 全局参数 ABS_PARAM_IFACE_VERSION

BSAPI 3.5 增加了一个与以前版本不兼容的特性：动态登记（见下文），新增全局参数 BSAPI，使调用者可以检测 BSAPI 接口。

旧接口（不支持该参数）是版本 1，当前是版本 2。如果 BSAPI 将来的版本引入了新的不兼容特性，它将使用更高的版本号。

如需循序渐进的区分接口版本 1 和版本 2，调用 ABSGetGlobalParameter()并将 dwParam 设置成 ABS_PARAM_IFACE_VERSION。返回值 ABS_STATUS_NOT_SUPPORTED 表示接口版本 1。返回值 ABS_STATUS_OK 表示要进一步解释返回的输出参数才能决定接口版本号。

***注：** 该参数是只读参数。即，只能用 ABSGetGlobalParameter() 获取参数值。在 ABSSetGlboalParameter()中尝试使用失败会返回 ABS_STATUS_NOT_SUPPORTED。*

8.2. 动态登记

BSAPI.DLL 3.5 及以后版本使用动态登记。这意味着不能提前知道登记一个指纹需要滑动手指多少次。登记进程中，每扫描一次指纹，将实现并分析生成的模板。BSAPI 评估指纹模板质量合格后进程结束。

这个变化要求 API 做出几个相应变化。

8.2.1. 全局参数 ABS_PARAM_CONSOLIDATION_COUNT

老版本 BSAPI 里已有出现，但是新版本里支持的值有所变化。老版本里，几乎所有设备支持 3 到 5 次滑动。当前版本仅支持 0（默认值，意味着动态登记）和 5。

3.5 及以后的版本不再支持值 3。

8.2.2. 结构体 ABS_PROCESS_BEGIN_DATA

消息 ABS_MSG_PROCESS_BEGIN 附加了最后一个发送给 ABS_CALLBACK 函数的参数指定的信息。数据是由 ABS_PROCESS_BEGIN_DATA 描述的。

现在，如果步骤数未知，例动态登记的情况下，可以将成员 StepCount 设置为 0。

8.2.3. 结构体 ABS_PROCESS_PROGRESS_DATA

新版本的 ABS_MSG_PROCESS_PROGRESS 携带更多数据。老版本里该结构体发送的是 ABS_PROCESS_DATA，现在换成了 ABS_PROCESS_PROGRESS_DATA。

***注：** ABS_PROCESS_PROGRESS_DATA 是二进制数据，与 ABS_PROCESS_DATA 兼容。ABS_PROCESS_PROGRESS_DATA 是 ABS_PROCESS_DATA 的超集。除了老的 ProcessID 成员，ABS_PROCESS_PROGRESS_DATA 的成员 Percentage 会将操作进度告知调用者。*

Percentage 的值达到 100 时，进程终止（动态登记）。对于不适用 Percentage 的进程，Percentage 的值设置为 FFFFFFFF。

8.2.4. 常量 ABS_PROCESS_CONSOLIDATE

合并不是登记进程的一个单独步骤，应为合并会改变登记进程而且是将几个分开的指纹模板逐渐的合并成一个登记模板。

对于结构体 ABS_PROCESS_DATA、ABS_PROCESS_BEGIN_DATA、ABS_PROCESS_PROGRESS_DATA 、和 ABS_PROCESS_SUCCESS_DATA，成员 ProcessID 的值绝不会设置成该常量。该常量保留在 bstypes.h 头部，已备应用源码向后兼容时使用。

8.3. 抓图函数

8.3.1. 常量 ABS_FLAG_HIGH_RESOLUTION

函数 ABSGrab 接受新标识位 ABS_FLAG_HIGH_RESOLUTION，该标识位要求函数使用现有的最高分辨率的图像。

8.3.2. 结构体 ABS_IMAGE

结构体 ABS_SAMPLE_IMAGE 改名成 ABS_IMAGE。ABS_SAMPLE_IMAGE 保留成 ABS_IMAGE 的 typedefed 别名，用于后向兼容。

8.3.3. 新抓图函数

BSAPI.DLL 3.5 新增了 3 个全新的抓图函数：ABSListImageFormats、ABSGrabImage 和 ABSRawGrabImage。它们都使用新结构体 ABS_IMAGE_FORMAT 鉴别被支持和需要的图像格式。

8.4. 全局参数 ABS_PARAM_POWER_SAVE_CHECK_KEYBOARD

新增全局参数 ABS_PARAM_POWER_SAVE_CHECK_KEYBOARD，调整电源管理的更多细节。

8.5. 内部模板格式类型

请注意本文仅指模板本身的数据，例 ABS_BIR 结构体的成员数据。BSAPI 返回的内部模板类型的前面总是 ABS_BIR_HEADER，同时 ABS_BOR_HEADER 也总是先于带输入数据。

UPEK 一般使用以下几种模板格式类型：

- 传统模板、
- 抢先模板、
- 稳定模板、
- 抢先多模板。

BSAPI.DLL 3.5 返回的指纹模板的内部格式有所改变。BSAPI.DLL 3.0 及以前的版本返回的指纹模板都是传统模板。对于 BSAPI.DLL 3.5 及以后的版本，抢先多模板和验证模板中的登记进程产生的指纹模板都是稳定模板。未来的 BSAPI 版本可能会采用其他新的深圳市现在未定义的模板格式。

BSAPI 3.5 能输入上述任意所列模板类型。用 ABSVerifyMatch 函数对比两个模板时，每一个输入的模板可以有不同的格式。所以，典型情况下不需要关注模板的起源版本。

如果确实需要特定格式的模板，例，需要将模板传入另一个库（不是 BSAPI SDK 的一部分）时不支持太新的模板类型，则可以使用 BCLIB 库。BCLIB 是作为 BSAPI SDK 的一部分提供的。提供的接口可以转换不同的模板格式类型。但是，某些模板的转换可能导致数据丢失，而且有些转换是不被支持的（例，不能将传统模板转换成稳定模板）。关于该主题的更多信息，请参考 BCLIB 文档。

8.6. 兼容 Windows NT 服务

BSAPI.DLL 3.5 及以后版本提供了新函数 ABSInitializeEx，此函数只有一个位掩码标识位作为其参数。在 Windows 平台上，此函数可以在 Windows NT 服务兼容的模式下进行初始化。

8.7. ABS_CALLBACK 和线程

交互操作的回调函数必须与调用交互操作的线程上下文一致。这一点对于在不支持多线程的编程语言，例 MS VisualBasic 中使用 BSAPI.DLL 的研发者尤其重要。

8.8. 终端和 Citrix 的支持情况

未来，BSAPI 能够通过 Windows 终端服务和 Citrix 的远程会话上下文中打开设备。当前版本的 BSAPI 还不支持该功能，但是 BSAPI 已经在几个方面有更新，以更好的适应该功能。

该功能的实施意味着使用 BSAPI 的进程能在远程会话中运行，并在客户端上打开指纹传感器设备。

目前，用户尝试在远程会话中打开设备时，调用 BSAPI 会失败并返回内部错误，错误消息的大概意思是“已经检测到当前情况，但 BSAPI 还不支持这一特性，所以调用失败”。

但是，有一个新的初始化标识位 ABS_INIT_FLAG_FORCE_LOCAL_SENSOR 能被传入 ABSInitializeEx 函数。使用后，检测远程会话的功能将被关闭，而 BSAPI 只能在本地设备上打开。该标志位已经被完全实施，即目前可以用该标识位防止 BSAPI 返回上诉内部错误。将来，它将强制 BSAPI 使用本地传感器，而从不使用远程传感器。

如果用户会话不是远程的，该标识位没有影响，则 BSAPI 总是使用本地传感器。