VIRBOX

魔锐 API 参考

Version 1.0.0.0

许可协议

同意本许可协议的所有条款及此处包含的任何补充或特殊的许可条款是获得本产品许可的必要条件。如果您不同意此协议的所有条款，请在三天内将产品退还北京深思数盾科技股份有限公司。您对本软件的使用将表明您同意接受本协议中条款的约束。

1. 授予您使用许可权。您可以为了备份的目的而复制磁盘中的软件，可以为了将本产品集成到您的软件的目的，根据本产品的文档说明将我们提供的软件合并进您的程序中。
2. 除已按上述第一条被授权外，不可以复制、修改、逆向工程、分解或重组该产品的全部或部分，不可向他人销售、租借、许可、转让、分发全部或部分本产品或本协议授予的权利。
3. 保证在自产品交给您之日起的 12 个月内，在正常使用情况下，产品不会出现实质性的材料上和生产制造上的缺陷。北京深思数盾科技股份有限公司的全部责任和您能获得的全部补救措施为：可选择尝试更换或修理或其他补救措施。
4. 除了上述对本产品的原始购买者所提供的有限保证之外，不向任何人作任何其他的保证。对北京深思数盾科技股份有限公司的产品、性能或服务亦没有明示的或暗示的或其他任何形式的保证，包括但不仅仅限于商品的适销性和对特定用途的适用性。
5. 任何情况下，无论如何引起及依据何种责任理论，均不负担任何因使用或不能使用本产品造成的损失责任，包括：丢失数据、损失利润及其他特别的、偶然的、附随的、继发的或间接的损失。
6. 所有的产品，包括魔锐设备、软件、文档、与本产品一并附送的其他材料及您制作的备份的所有权与版权均属于北京深思数盾科技股份有限公司。
7. 违反上述条款时，本协议的授权将自动终止。

联系深思数盾

公司名称：北京深思数盾科技股份有限公司
办公地点：北京市海淀区西北旺东路 10 号院 5 号楼软件园二期互联网创新中心 C510
邮　　编：100872
电　　话：+86-10-556730936
传　　真：+86-10-556730936
电子邮件：sense@sense.com.cn
网　　址：http://www.sense.com.cn

阅读指南

〖手册目标〗

本手册主要对北京深思数盾科技股份有限公司开发的魔锐加密锁的使用进行说明，由于实际情况千变万化，本手册很难一次做到面面俱到，需要逐渐完善。

〖手册约定〗

手册中出现该标志的地方表示需要您引起高度重视，否则可能会引发严重的后果。

手册中出现该标志的地方表示需要您特别引起注意的内容。

目 录

第1章 常量定义

1.1 打开方式

宏名	数值	说明
MW_OPEN_SHARE_MODE	0x00	共享打开设备
MW_OPEN_EXCLUSIVE_MODE	0x01	独占打开设备

说明：打开函数使用

1.2 PIN 类型

宏名	数值	说明
MW_PIN_TYPE_NONE	0x00	空类型
MW_PIN_TYPE_USER	0x01	用户类型
MW_PIN_TYPE_DEVELOPER	0x02	开发商类型

1.3 PIN 长度

宏名	数值	说明
MW_PIN_LENGTH_USER	8	用户 PIN 长度
MW_PIN_LENGTH_DEVELOPER	24	开发商 PIN 长度

1.4 文件类型

宏名	数值	说明
MW_FILE_TYPE_BINARY	0x00	二进制数据文件
MW_FILE_TYPE_KEY	0x02	密钥文件

1.5 文件操作权限

宏名	数值	说明
MW_FILE_PRIV_TYPE_USE	0x00	密钥文件类型，开发商及用户只能使用
MW_FILE_PRIV_TYPE_READ	0x01	只读数据文件类型，用户权限只能读取
MW_FILE_PRIV_TYPE_READ_WRITE	0x02	读写数据文件类型，用户权限可以读写

1.6 算法类型

宏名	数值	说明
MW_ALG_TYPE_AES	0x00	AES 算法
MW_ALG_TYPE_DES	0x01	DES 算法
MW_ALG_TYPE_TDES	0x02	TDES 算法
MW_ALG_TYPE_ECC	0x10	ECC 算法
MW_ALG_TYPE_RSA	0x11	RSA 算法

说明：对称加解密函数与非对称加解密函数使用

1.7 对称算法模式

宏名	数值	说明
MW_SYM_ALGO_MODE_ECB	0x00	对称算法 ECB 模式
MW_SYM_ALGO_MODE_CBC	0x01	非对称算法 CBC 模式

说明：对称加解密函数使用

1.8 RSA 加解密填充模式

宏名	数值	说明
MW_RSA_MODE_NORMAL	0x00	不填充加密模式
MW_RSA_MODE_PKCS1_V1_5	0x01	PKCS1 V1_5 填充加密模式

1.9 RSA 算法长度

宏名	数值	说明
MW_RSA_1024_BYTE_SIZE	128	RSA1024 计算模字节长度
MW_RSA_2048_BYTE_SIZE	256	RSA2048 计算模字节长度

1.10 签名 HASH 类型

宏名	数值	说明
MW_HASH_ALGO_MD5	0x00	签名 MD5 hash 算法
MW_HASH_ALGO_SHA1	0x01	签名 SHA1 hash 算法
MW_HASH_ALGO_SHA256	0x02	签名 SHA256 hash 算法
MW_HASH_ALGO_NONE	0xFF	签名时不做 Hash 运算，由调用函数者自己做 Hash 运算

1.11 HMAC 摘要大小

宏名	数值	说明
MW_HMAC_MD5_DIGEST_LENGTH	16	hmac- MD5 算法摘要长度
MW_HMAC_SHA1_DIGEST_LENGTH	20	hmac - SHA1 算法摘要长度
MW_HMAC_SHA256_DIGEST_LENGTH	32	hmac - SHA256 算法摘要长度

1.12 多语言 ID

宏名	数值	说明
MW_LANGUAGE_ID_DEFAULT	0x00	默认中文
MW_LANGUAGE_ID_CH	0x01	中文
MW_LANGUAGE_ID_EN	0x02	英文

说明：调用错误帮助函数使用

1.13 大小相关宏定义

宏名	数值	说明
MW_PATH_LEN	1024	设备路径最大长度
MW_SYM_ALG_IV_LENGTH	16	设备对称算法 CBC 模式 IV 长度
MW_SN_LENGTH	16	设备唯一序列号长度
MW_FILE_NAME_LENGTH_MAX	16	设备文件名最大长度
MW_INPUT_DATA_LENGTH_MAX	1900	设备命令执行数据最大长度
MW_SEED_LENGTH_MAX	32	种子码最大长度
MW_SEED_LENGTH_MIN	4	种子码最小长度
MW_ENUM_DEVICE_MAX_COUNT	128	主机最大设备数量

1.14 设备初始化状态

宏名	数值	说明
MW_STATUS_FLAG_DEFAULT	0x00	设备处于出厂魔锐状态，PID 与开发商 PIN 均未修改
MW_STATUS_FLAG_PID	0x01	设备出厂魔锐 PID 已经修改
MW_STATUS_FLAG_PIN	0x02	设备出厂默认开发商 PIN 已经修改

说明：只有出厂 PID 与开发商 PIN 都修改了，才能使用魔锐相关应用功能

1.15 设备信息码

宏名	数值	说明
MW_GET_INFO_ITEM_PID	0x00	获取 PID，PID 为 4 字节
MW_GET_INFO_ITEM_SN	0x01	获取唯一序列号，长度 16 字节
MW_GET_INFO_ITEM_PRODUCE_DATE	0x02	获取生产日期，长度 4 字节
MW_GET_INFO_ITEM_ALL_CAPACITY	0x03	获取总的容量，长度 4 字节
MW_GET_INFO_ITEM_FREE_CAPACITY	0x04	获取剩余容量，长度 4 字节
MW_GET_INFO_ITEM_VERSION	0x05	获取设备版本，长度 4 字节
MW_GET_INFO_ITEM_SHELL_SN	0x06	获取外壳序号，首字节为长度
MW_GET_INFO_ITEM_STATUS	0x07	获取设备初始化状态，
MW_GET_INFO_ITEM_UPDATE_CODE	0x08	获取设备升级号
MW_GET_INFO_ITEM_UPDATE_CTRL_CODE	0x09	获取升级设备类型
MW_GET_INFO_ITEM_ALL	0xFF	获取以上所有设备信息

1.16 控制码

宏名	数值	说明
MW_CTRL_CODE_ITEM_RESET	0x00	复位设备状态，复位后设备回到上电状态权限都被清除
MW_CTRL_CODE_ITEM_LED	0x01	控制 LED 显示状态，亮或灭

第2章 类型定义

2.1 设备的句柄

```
typedef        void*        MWHANDLE;
```

2.2 设备信息结构体

```
typedef struct _MW_DEVICE_INFO_CTX{
    unsigned int      uiPID;                          // 产品 ID
    unsigned char     ucSerialNum[MW_SN_LENGTH];   // 唯一序列号
    unsigned char     ucDevPath[MW_PATH_LEN];   // 设备的路径
    unsigned int      uiProtocol;                     // 通讯协议
    unsigned int      uiLocationID;                   // Mac OS 系统 LocationID
}MW_DEVICE_INFO_CTX;
```

2.3 文件属性结构体

```
typedef struct _MW_FILE_PROPERTY{
    unsigned char     ucValidate;          // 该成员变量暂时不用
    unsigned char     ucType;              // 文件类型：二进制文件或密钥文件
    unsigned short    usPrivilege;         // 文件权限
    unsigned int      uiSize;              // 文件大小
    unsigned int      uiTime;              // 创建时间
    char acName[MW_FILE_NAME_LENGTH_MAX + 1];    // 文件名称
}MW_FILE_PROPERTY;
```

2.4 系统工作状态结构体

```
typedef struct _MW_DEVICE_STATUS
{
    unsigned int      uiSysTime;           // 系统时间
    unsigned int      uiSysStatus;         // 系统状态
    unsigned short    usReserved;          // 保留
    unsigned char     ucLoginStatus;       // 登录状态
    unsigned char     ucLedStatus;         // 灯的状态
}MW_DEVICE_STATUS;
```

2.5 远程升级结构体

```
typedef struct _MW_UPDADE_FILE_CTX
{
    MW_FILE_PROPERTY stFileProperty;      // 文件属性结构体
    unsigned char *pData;                 // 升级文件的数据内容
    unsigned int uiDateSize;              // 升级文件的数据大小
}MW_UPDADE_FILE_CTX;
```

2.6 设备信息子项结构体

```
typedef struct _MW_DEVICE_ALL_ITEM_INFO
{
    unsigned int uiPID;                   // 产品号
    unsigned char acSN[MW_SN_LENGTH];     // 唯一序列号
    unsigned int uiProduceDate;           // 生产时间
    unsigned int uiAllCapacity;           // 总容量
    unsigned int uiFreeCapacity;          // 可用容量
    unsigned int uiVersion;               // 版本号
    unsigned int uiUpdateCode;            // 升级码
    unsigned char ucStatus;               // 状态
    unsigned char ucUpdateCtrlCode;       // 升级控制码
    unsigned short reserved;              // 保留
}MW_DEVICE_ALL_ITEM_INFO;
```

第3章 函数说明

3.1 mw_enum

```
unsigned  int MWAPI mw_enum(OUT MW_DEVICE_INFO_CTX     *pDevInfoList,
                            IN  unsigned int            uiDevListCount,
                            OUT unsigned int           *puiDevCount);
```

参数说明：

pDevInfoList	枚举输出的魔锐设备列表结构数据内存地址，需要调用者分配此内存
uiDevListCount	分配(pDevInfoList)内存区能存放的设备列表结构数据的个数。注：不清楚当前设备个数时，可以按最大个数申请（参见宏定义 MW_ENUM_DEVICE_MAX_COUNT）
puiDevCount	返回实际枚举出的有效魔锐设备数量，即 pDevInfoList 中有效设备结构体个数

返回说明：

成功时返回 MW_SUCCESS
其它可能的返回值，请参阅“错误码”部分

函数说明：

枚举当前主机上所有的魔锐设备。
此函数需要用户必须按照 uiDevListCount 乘以 sizeof(MW_DEVICE_INFO_CTX)分配内存给 pDevInfoList，API 内部不分配内存给 pDevInfoList。
当分配的设备内存小于实际的设备内存时，返回分配的设备内存个数。
当分配的设备内存大于实际的设备内存时，返回实际的设备内存个数。

3.2 mw_open

```
unsigned  int MWAPI mw_open(IN   MW_DEVICE_INFO_CTX *pDevInfo,
                            IN   unsigned int    uiShareMode,
                            OUT  MWHANDLE        *phHandle);
```

参数说明：

pDevInfo	指向魔锐设备结构体的指针.（mw_enum 枚举到的 pDevInfoList 中某个设备结构体）
uiShareMode	打开模式，目前只支持设备独占模式打开；参见宏定义

	MW_OPEN_EXCLUSIVE_MODE.
phHandle	设备句柄，对设备所有操作均通过此句柄对设备操作.

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

打开指定魔锐设备；目前只支持独占打开； 魔锐设备结构体是由枚举函数获得，为设备列表中的某个设备结构体；

3.3 mw_close

```
unsigned  int MWAPI mw_close(IN   MWHANDLE   hHandle);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

关闭指定魔锐设备；

3.4 mw_verify_pin

```
unsigned  int MWAPI mw_verify_pin(IN   MWHANDLE      hHandle,
                                  IN   unsigned char ucPinType,
                                  IN   char          *pucPin);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
ucPinType	Pin 码类型　参见宏定义：MW_PIN_TYPE_XXX.（注意：文档中 XXX 代表通用替代指示字符，需要根据前缀查看实际的意义）
pucPin	Pin 码数据，用户 8 字节，开发商 24 字节.

返回说明：

成功时返回 MW_SUCCESS.

其它可能的返回值，请参阅“错误码”部分.

函数说明：

校验 PIN； 校验正确后，获取对应的设备操作权限，不插拔设备或关闭设备，设备权限一直有效，可以通过 mw_control 函数复位命令清除设备权限；

如果 PIN 码设置了校验错误次数限制，在校验错误次数超过限制时，会导致设备锁死。

3.5 mw_change_pin

```
unsigned  int MWAPI mw_change_pin(IN   MWHANDLE          hHandle,
                                  IN   unsigned char     ucPinType,
                                  IN   unsigned short    usLimitCount,
                                  IN   char              *pucOldPin,
                                  IN   char              *pucNewPin);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
ucPinType	Pin 码类型　参见宏定义：MW_PIN_TYPE_XXX
usLimitCount	最大尝试次数，不限次数设置为 0，限制次数范围是 1-15，　如为其它数则返回参数错误
pucOldPin	旧 Pin 码数据，用户 8 字节，开发商 24 字节
pucNewPin	新 Pin 码数据，用户 8 字节，开发商 24 字节.

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

修改 PIN； 此操作需要开发商或用户权限； 设备出厂后必须进行初始化修改开发商 PIN 操作，否则相关加密锁加密功能不能使用； 管理员权限可以直接修改用户 PIN，此时，pucOldPin 参数可以无效；

开发商 PIN 拥有最高权限，请你妥善保管；如果 PIN 码设置了校验错误次数太少，很容易造成设备锁死，用户锁死后可以通过开发商重置。开发商锁死后，深思也没有办法帮你重置，请根据实际情况需要进行次数的设置。

当你刚拿到设备，请你先进行开发商 PIN 码的修改与 PID 的设置，否则你不能使用魔锐相关应用功能。

3.6 mw_set_pid

```
unsigned int MWAPI mw_set_pid(IN     MWHANDLE          hHandle,
                              IN     unsigned char     *pucPIDSeed,
                              IN     unsigned int      uiSeedLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pucPIDSeed	种子码
uiSeedLen	种子码长度，长度范围是 4-32 字节，参见宏定义 MW_SEED_LENGTH_MIN、MW_SEED_LENGTH_MAX.

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

设置 PID;
此操作需要开发商权限；设备出厂后必须进行初始化设置 PID 操作，否则相关加密锁加密功能不能使用；

当你刚拿到设备，请你先进行开发商 PIN 码的修改与 PID 的设置，否则你不能使用魔锐相关应用功能。

3.7 mw_control

```
unsigned  int MWAPI mw_control(IN      MWHANDLE          hHandle,
                               IN      unsigned char     uiCtrlCodeItem,
                               IN      void              *pvInBuffer,
                               IN      unsigned int      uiInBufferLen,
                               OUT     void              *pvOutBuffer,
                               IN      unsigned int      uiOutBufferLen,
                               OUT     unsigned int     *puiReturnedLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
uiCtrlCodeItem	控制码，参见宏定义 MW_CTRL_CODE_ITEM_XXX
pvInBuffer	输入数据
uiInBufferLen	输入数据的长度
pvOutBuffer	输出数据
uiOutBufferLen	输出数据的长度
puiReturnedLen	实际输出数据的长度.

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

控制命令；
复位控制操作可以用于清除当前设备权限状态；
LED 只支持亮与灭控制；

3.8 mw_get_device_info

```
unsigned  int MWAPI mw_get_device_info(IN  MWHANDLE         hHandle,
                                       IN  unsigned char    ucInfoItem,
                                       OUT void             *pvBuffer,
                                    INOUT unsigned int      *puiBufferLength);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
ucInfoItem	指定获取的设备信息的具体项，参见宏定义：MW_GET_INFO_ITEM_XXX
pvBuffer	输出获取的设备信息项，缓冲区大小由获取项目的大小决

	定，参考设备大小宏定义相关
puiBufferLength	输出设备信息项的数据长度

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

获取设备信息； 可以获取设备序列号、生产日期、版本、容量等设备相关信息；

3.9 mw_get_device_status

```
Unsigned int MWAPI mw_get_device_status(IN WHANDLE      hHandle,
                                        OUT MW_DEVICE_STATUS  *pstDeviceStatusCtx);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pstDeviceStatusCtx	当前设备状态结构体，参看 MW_DEVICE_STATUS

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

获取设备当前状态； 可以获得设备当前的工作状态及当前权限等状态；

3.10 mw_enum_file

```
unsigned  int MWAPI mw_enum_file(IN  MWHANDLE          hHandle,
                                 OUT MW_FILE_PROPERTY   *pstFilePropertyList,
                                 IN  unsigned int       uiFileListCount,
                                 OUT unsigned int       *puiReturnedFileCount);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pstFilePropertyList	结构体 MW_FILE_PROPERTY 数组，它的大小由 uiFileListCount 指定

uiFileListCount	指定 pstFilePropertyList 数组的大小
puiReturnedFileCount	实际返回列举到文件的个数

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

列举文件；
此操作需要开发商或用户权限；
pstFilePropertyList 由函数外部调用者分配，计算大小为 uiFileListCount * sizeof(MW_FILE_PROPERTY)；
8K 容量的魔锐最大支持 16 个文件，32K 容量魔锐最大支持 64 个文件

3.11 mw_create_file

```
MWAPI mw_create_file(IN      MWHANDLE            hHandle,
                     IN      MW_FILE_PROPERTY    *pstFileProperty);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pstFileProperty	指定需要创建文件的属性，其中文件类型、文件权限、文件大小、文件名称为必填项

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

创建文件；
此操作需要开发商权限；

8K 设备最多创建 16 个文件，32K 设备最多创建 64 个文件；。

3.12 mw_read_file

```
unsigned  int MWAPI mw_read_file(IN      MWHANDLE          hHandle,
                                 IN      char              *pcFileName,
                                 IN      unsigned int      uiReadOffset,
                                 IN      unsigned int      uiReadSize,
                                 OUT     unsigned char   *pucReadBuffer);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcFileName	文件名称，以’\0’结尾的字符串
uiReadOffset	读文件数据的偏移量
uiReadSize	读取文件数据的大小
pucReadBuffer	读取文件数据到缓冲区。注：必须分配足够 uiReadSize 大小，否则会访问越界

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

读文件数据；
此操作需要开发商或用户权限；
开发商与用户权限可以读取所有数据文件内容，密钥文件任何权限都不能读取；

3.13 mw_write_file

```
unsigned  int MWAPI mw_write_file(IN      MWHANDLE          hHandle,
                                  IN      char              *pcFileName,
                                  IN      unsigned int      uiWriteOffset,
                                  IN      unsigned int      uiWriteSize,
                                  IN      unsigned char *pucWriteBuffer);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcFileName	文件名称，以’\0’结尾的字符串
uiWriteOffset	写文件的地址的偏移量
uiWriteSize	写文件数据大小
pucWriteBuffer	需要写入文件的数据缓冲区

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

写文件；
此操作需要开发商权限；
用户权限只能写用户写权限的数据文件，其他数据文件只有读取权限；
开发商可以写所有文件，密钥文件只能一次性写入，不能按偏移量写；

3.14 mw_delete_file

```
unsigned  int MWAPI mw_delete_file(IN      MWHANDLE      hHandle,
                                   IN      char          *pcFileName);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcFileName	文件名称，以’\0’结尾的字符串

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

删除文件；
此操作需要开发商权限；

3.15 mw_get_file_property

```
unsigned int MWAPI mw_get_file_property(IN   MWHANDLE           hHandle,
                                        IN   char               *pcFileName,
                                        OUT MW_FILE_PROPERTY *pstFileProperty);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcFileName	文件名称，以’\0’结尾的字符串
pstFileProperty	输入的文件属性结构体数据

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

获取指定文件的文件属性； 此操作需要开发商或用户权限；

3.16 mw_sym_encrypt

```
unsigned int MWAPI mw_sym_encrypt(IN       MWHANDLE          hHandle,
                                  IN       char              *pcKeyFileName,
                                  IN       unsigned char     ucAlgoMode,
                                  IN       unsigned char     *pucIV,
                                  IN       unsigned char     *pucInData,
                                  IN       unsigned int      uiInDataLen,
                                  OUT      unsigned char     *pucOutData,
                                  INOUT    unsigned int      *puiOutDataLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcKeyFileName	密钥文件名称 ， 以’\0’结尾的字符串
ucAlgoMode	对称加密的算法，参见宏定义：MW_ALG_TYPE_XXX
pucIV	对称加密算法中需要使用到的向量，固定 16 字节，CBC 模式需要填入，ECB 可不填
pucInData	加密输入的明文，必须是 16 的倍数
uiInDataLen	加密明文的数据长度
pucOutData	密后输出的密文
puiOutDataLen	实际输出密文的数据长度

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

对称加密； 操作需要开发商或用户权限 密钥文件必须要对此算法 DES,TDES，AES 类型，否则会返回错误； 输入数据必须是 16 字节整数倍；

DES 与 TDES 加密 CBC 的 IV 只需要填写前 8 字节，模式最大支持 1888 字节长

度解密；。

3.17 mw_sym_decrypt

```
unsigned int MWAPI mw_sym_decrypt(IN       MWHANDLE          hHandle,
                                  IN       char              *pcKeyFileName,
                                  IN       unsigned char     ucAlgoMode,
                                  IN       unsigned char     *pucIV,
                                  IN       unsigned char     *pucInData,
                                  IN       unsigned int      uiInDataLen,
                                  OUT      unsigned char     *pucOutData,
                                  INOUT    unsigned int      *puiOutDataLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcKeyFileName	密钥文件名称 ，以'\0'结尾的字符串
ucAlgoMode	对称加密的算法，参见宏定义：MW_ALG_TYPE_XXX
pucIV	对称加密算法中需要使用到的向量，固定 16 字节，CBC 模式需要填入，ECB 可不填
pucInData	输入的密文，必须是 16 的倍数
uiInDataLen	输入密文长度
pucOutData	解密后输出明文
puiOutDataLen	解密后输出明文长度

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

对称解密；
此操作需要开发商或用户权限；
密钥文件必须要对此算法 DES,TDES，AES 类型，否则会返回错误；
输入数据必须是 16 字节整数倍；

DES 与 TDES 加密 CBC 的 IV 只需要填写前 8 字节，模式最大支持 1888 字节长度解密；。

3.18 mw_rsa_encrypt

```
unsigned int MWAPI mw_rsa_encrypt(IN       MWHANDLE        hHandle,
                                  IN       const char      *pcKeyFileName,
                                  IN       unsigned char   ucPadMode,
                                  IN       unsigned char   *pucInData,
                                  IN       unsigned int    uiInDataLen,
                                  OUT      unsigned char   *pucOutData,
                                  INOUT    unsigned int    *puiOutDataLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcKeyFileName	密钥文件名称 ，以’\0’结尾的字符串
ucPadMode	加密的算法，参见宏定义：MW_RSA_MODE_XXX
pucInData	输入明文数据
uiInDataLen	输入明文数据的长度
pucOutData	输出密文数据
puiOutDataLen	输出密文数据的长度.

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

rsa 非对称加密；
此操作需要开发商或用户权限；
密钥文件必须为 RSA 密钥，否则会返回错误；
当模式为 MW_RSA_MODE_NORMAL 时， RSA1024 算法加密输入明文长度为 MW_RSA_1024_BYTE_SIZE ， RSA2048 算法加密输入明文长度为 MW_RSA_2048_BYTE_SIZE；
当模式为 MW_RSA_MODE_PKCS1_V1_5 时，RSA1024 算法加密输入明文最大长度为 MW_RSA_1024_BYTE_SIZE-11 ， RSA2048 算法加密输入明文最大长度为 MW_RSA_2048_BYTE_SIZE-11

3.19 mw_rsa_decrypt

```
unsigned int MWAPI mw_rsa_decrypt(IN       MWHANDLE        hHandle,
                                  IN       char            *pcKeyFileName,
                                  IN       unsigned char   ucPadMode,
                                  IN       unsigned char   *pucInData,
                                  IN       unsigned int    uiInDataLen,
                                  OUT      unsigned char   *pucOutData,
                                  INOUT    unsigned int    *puiOutDataLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcKeyFileName	密钥文件名称 ，以'\0'结尾的字符串
ucPadMode	加密的算法，参见宏定义：MW_RSA_MODE_XXX
pucInData	输入密文
uiInDataLen	输入密文长度，长度必须为 128(1024)或者 256(2048)
pucOutData	输出明文
puiOutDataLen	输出明文长度.

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

Rsa，非对称解密；
此操作需要开发商或用户权限；
密钥文件必须为 RSA 密钥，否则会返回错误；
当模式为 MW_RSA_MODE_NORMAL 时， RSA1024 算法加密输出明文长度为 MW_RSA_1024_BYTE_SIZE ， RSA2048 算法加密输出明文长度为 MW_RSA_2048_BYTE_SIZE；
当模式为 MW_RSA_MODE_PKCS1_V1_5 时，RSA1024 算法加密输出明文最大长度为 MW_RSA_1024_BYTE_SIZE-11 ， RSA2048 算法加密输出明文最大长度为 MW_RSA_2048_BYTE_SIZE-11

3.20 mw_signature

```
unsigned int MWAPI mw_signature(IN       MWHANDLE        hHandle,
                                IN       char            *pcKeyFileName,
                                IN       unsigned char   ucHashAlg,
                                IN       unsigned char   *pucMessageData,
                                IN       unsigned int    uiMessageDataLen,
                                OUT      unsigned char   *pucSignData,
                                INOUT    unsigned int    *puiSignDataLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcKeyFileName	密钥文件名称 ，以’\0’结尾的字符串
ucHashAlg	哈希算法类型，参见宏定义：MW_HASH_ALGO_SHA1, MW_HASH_ALGO_SHA256, MW_HASH_ALGO_NONE
pucMessageData	输入消息数据
uiMessageDataLen	输入消息数据长度 注：最大数据长度参见宏定义 MW_INPUT_DATA_LENGTH_MAX
pucSignData	输出签名数据
puiSignDataLen	输出签名数据长度

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

数据签名；
此操作需要开发商或用户权限；
密钥文件必须为 RSA 或 ECC 密钥，否则会返回错误；
返回签名数据长度根据算法各不相同，最大为 RSA2048 签名长度为 MW_RSA_2048_BYTE_SIZE；

3.21 mw_verify_sign

```
unsigned int MWAPI mw_verify_sign(IN       MWHANDLE          hHandle,
                                  IN       char              *pcKeyFileName,
                                  IN       unsigned char     ucHashAlg,
                                  IN       unsigned char     *pucSignData,
                                  IN       unsigned int      uiSignDataLen,
                                  IN       unsigned char     *pucMessageData,
                                  IN       unsigned int      uiMessageDataLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcKeyFileName	密钥文件名称 ，以’\0’结尾的字符串
ucHashAlg	哈希算法类型，参见宏定义：MW_HASH_ALGO_SHA1, MW_HASH_ALGO_SHA256, MW_HASH_ALGO_NONE
pucSignData	输入签名数据
uiSignDataLen	输入签名数据的长度
pucMessageData	输入消息数据

uiMessageDataLen	输入消息数据长度

返回说明：

成功时返回 MW_SUCCESS.

其它可能的返回值，请参阅“错误码”部分.

函数说明：

验证签名；
此操作需要开发商或用户权限；
密钥文件必须为 RSA 或 ECC 密钥，否则会返回错误；

3.22 mw_hmac_calc

```
unsigned int MWAPI mw_hmac_calc(IN      MWHANDLE        hHandle,
                                IN      char            *pcKeyFileName,
                                IN      unsigned char   *pucInData,
                                IN      unsigned int    uiInDataLen,
                                OUT     unsigned char   *pucOutData,
                                INOUT   unsigned int    *puiOutDataLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pcKeyFileName	密钥文件名称 ，以’\0’结尾的字符串
pucInData	输入数据
uiInDataLen	输入数据的长度
pucOutData	输出摘要数据
puiOutDataLen	输出摘要数据长度，参见宏定义：MW_HMAC_MD5_DIGEST_LENGTH 、MW_HMAC_SHA1_DIGEST_LENGTH 、MW_HMAC_SHA256_DIGEST_LENGTH

返回说明：

成功时返回 MW_SUCCESS.

其它可能的返回值，请参阅“错误码”部分.

函数说明：

HMAC 计算；
此操作需要开发商或用户权限；
密钥类型必须是 hmac 类型密钥文件，否则会返回错误；
返回数据长度根据选择 hmac 密钥文件的哈希算法类型不同,返回长度各不相同；

3.23 mw_make_update_pkg

```
unsigned int MWAPI mw_make_update_pkg(IN      MWHANDLE            hHandle,
                                      IN      unsigned int        uiDevPID,
                                      IN      unsigned char       *pucSN,
                                      IN      MW_UPDADE_FILE_CTX  *pstUpdateFileList,
                                      IN      unsigned int        uiFileCount,
                                      OUT     unsigned char       *pucOutPkg,
                                      INOUT   unsigned int        *puiOutPkgLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
uiDevPID	设备产品号
pucSN	硬件唯一序列号 固定长度为：MW_SN_LENGTH，不指定 SN 绑定时，可以为 NULL
pstUpdateFileList	文件路径列表，最大文件个数参见宏定义 MW_UPDATE_FILE_COUNT_MAX，升级文件的个数为 uiFileCount
uiFileCount	升级文件的个数,它指定了 pstUpdateFileList 中的大小
pucOutPkg	升级包数据
puiOutPkgLen	升级包数据长度.

返回说明：

成功时返回 MW_SUCCESS.
其它可能的返回值，请参阅“错误码”部分.

函数说明：

制作升级包；
此操作需要开发商权限；
此功能需要控制锁才能制作升级包；

3.24 mw_update

```
unsigned int MWAPI mw_update(IN      MWHANDLE        hHandle,
                             IN      unsigned char   *pucInPkg,
                             IN      unsigned int    uiInPkgLen);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.
pucInPkg	升级包数据

uiInPkgLen	升级包数据长度

返回说明：

成功时返回 MW_SUCCESS.

其它可能的返回值，请参阅“错误码”部分.

函数说明：

远程升级；

用户权限可进行升级操作；

3.25 mw_restore_factory

```
unsigned int MWAPI mw_restore_factory(
                                    IN      MWHANDLE          hHandle);
```

参数说明：

hHandle	设备句柄，mw_open 函数获得.

返回说明：

成功时返回 MW_SUCCESS.

其它可能的返回值，请参阅“错误码”部分.

函数说明：

恢复出厂设置；

此操作需要开发商权限；

将设备恢复出厂状态，即默认 PID、默认 PIN，文件系统清空；

恢复默认后，需要重新初始化才能使用设备应用相关功能；

回复出厂后，设备与你刚购买到时的设备状态相同，所有设备信息将回到默认状态，设备内文件将会被全部删除，执行此命令前请确认执行。

3.26 mw_get_error_desc

```
const char * mw_get_error_desc(IN  unsigned int uiErrorCode,
          ;                IN  unsigned int uiLanguageID);
```

参数说明：

uiErrorCode	魔锐错误码.
uiLanguageID	错误码解析返回字符串语言种类 ID

返回说明：

返回指定语言的错误码解析字符串；

函数说明：

错误码解析；

第4章 错误码

```
#define MW_SUCCESS                                0x00000000 // 成功.
#define MW_ERROR_INVALID_HANDLE                   0x00000001 // 句柄可能为空
#define MW_ERROR_INVALID_PARAMETER                0x00000002 // 参数不合法
#define MW_ERROR_NOT_ENOUGH_MEMORY                0x00000003 // 没有足够的存储空间来处理这条命令
#define MW_ERROR_NO_DEVICE                        0x00000004 // 没有发现设备或特定设备
#define MW_ERROR_TIMEOUT                          0x00000005 // 加密锁通信超时
#define MW_ERROR_UNSUPPORTED_FLAG                 0x00000006 // 传递给了 APIs 不支持的标识
#define MW_ERROR_INSUFFICIENT_BUFFER              0x00000007 // 缓冲区太小不足以容纳数据
#define MW_ERROR_EXCHG_MEMORY_NOT_FOUND           0x00000008 // 未发现指定的共享内存
#define MW_ERROR_SYSTEM_FILE_NOT_FOUND            0x00000009 // 未发现文件在系统中
#define MW_ERROR_SYSTEM_FILE_INVALID_ACCESS       0x0000000a // 不能访问系统文件
#define MW_ERROR_FILE_EXISTS                      0x0000000b // 指定文件已存在
#define MW_ERROR_FILE_NOT_FOUND                   0x0000000c // 未发现指定文件
#define MW_ERROR_NO_PRIVILEGE                     0x0000000d // 操作需要更高权限
#define MW_ERROR_WRONG_PASSWORD                   0x0000000e // 密码错误
#define MW_ERROR_PASSWORD_LOCKED                  0x0000000f // 密码被锁定
#define MW_ERROR_FUNCTION_NOT_SUPPORTED           0x00000011 // 不支持该功能
#define MW_ERROR_COMMUNICATION                    0x00000015 // USB 端口通信失败，可能需要重启电脑
#define MW_ERROR_UNSUPPORTED_PASSWORD_TYPE        0x00000016 // 传递给 APIs 不支持的密码类型
#define MW_ERROR_WRONG_PASSWORD_LENGTH            0x00000017 // 密码长度错误，开发商是 24 字节，用户是 8 字节
#define MW_ERROR_ALREADY_EXCLUSIVELY_LOGIN        0x00000018 // 设备已经唯一登陆，不能通过共享模式登陆
#define MW_ERROR_ALREADY_SHARED_LOGIN             0x00000019 // 设备已经共享登陆，不能通过唯一模式登陆
#define MW_ERROR_ALREADY_TEMP_EXCLUSIVELY_USING   0x0000001a // 句柄已经临时使用
#define MW_ERROR_NOT_TEMP_EXCLUSIVELY_USING       0x0000001b // 句柄不是临时使用
#define MW_ERROR_TOO_MUCH_DATA                    0x0000001c // 消息响应函数具有数据长度限制
#define MW_ERROR_INSUFFICIENT_DEVICE_SPACE        0x0000001e // 设备空间不足
#define MW_ERROR_DEVICE_FILESYSTEM_ERROR          0x0000001f // 设备文件系统错误
#define MW_ERROR_FILE_OUT_RANGE                   0x00000020 // 设备文件超出范围
#define MW_ERROR_UNSUPPORTED_FILE_TYPE            0x00000021 // 传递给 EV APIs 不支持的文件类型
#define MW_ERROR_FILE_OFFSET_MUST_BE_ZERO         0x00000022 // 写或读关键文件，偏移量必须是 0
#define MW_ERROR_BAD_EXECUTIVE_FILE_FORMAT        0x00000023 // 可执行文件格式错误
#define MW_ERROR_OPEN_TOO_MANY_DEVICE_FILES       0x00000024 // 打开太多设备文件
#define MW_ERROR_INVALID_DEVICE_FILE_HANDLE       0x00000025 // 设备文件句柄错误
#define MW_ERROR_FILE_NOT_FINISHED                0x00000026 // 文件未完成
#define MW_ERROR_BAD_FILENAME                     0x00000027 // 文件名错误
#define MW_ERROR_BAD_NAME                         0x00000028 // 文件名，目录名，或者卷标语法错误
#define MW_ERROR_DEVICE_TIMER                     0x00000029 // 设备时间错误
#define MW_ERROR_NO_EXECUTIVE_FILE_RUNNING        0x0000002a // 设备中没有进程运行
#define MW_ERROR_DEVICE_USER_BUFFER_FULL          0x0000002b // 当设备用户缓存已满，不能发送数据
#define MW_ERROR_DEVICE_USER_BUFFER_EMPTY         0x0000002c // 当设备用户缓存为空，可以接收数据
#define MW_ERROR_DEVICE_MSG_NOT_REPLIED           0x0000002d // 设备先需要个消息回应
```

```
#define MW_ERROR_DEVICE_DUMMY_MSG                0x0000002e // 设备不需要消息回应
#define MW_ERROR_DEVICE_MEMORY_ADDR              0x0000002f // 设备内存地址错误
#define MW_ERROR_DEVICE_MEMORY_LENGTH            0x00000030 // 设备内存长度错误
#define MW_ERROR_CONTROL_CODE                    0x00000031 // 提供不支持的控制代码
#define MW_ERROR_UNKNOW_COMMAND                  0x00000032 // 提供给设备不支持的命令
#define MW_ERROR_INVALID_COMMAND_PARAMETER       0x00000033 // 命令参数错误
#define MW_ERROR_COMMAND_DATA_LENGTH             0x00000034 // 命令长度错误
#define MW_ERROR_DEVICE_BUFFER_FULL              0x00000035 // 设备缓存已满
#define MW_ERROR_EXECUTIVE_FILE_RUNNING          0x00000036 // 当设备中有进程在运行，一些操作不支持
#define MW_ERROR_NO_DEVICE_MESSAGE               0x00000037 // 没有设备消息
#define MW_ERROR_LOGIN_COUNT                     0x00000038 // 设备登陆计数错误
#define MW_ERROR_KEYEXCHANGE_FAILED              0x00000039 // 通信秘钥交换错误
#define MW_ERROR_BAD_COMMUNICATION_KEY           0x0000003a // 通信秘钥错误
#define MW_ERROR_BAD_DEVICE_TIME                 0x0000003b // 设备时间错误
#define MW_ERROR_BAD_DEVICE_INFOMATION           0x0000003c // 设备信息错误
#define MW_ERROR_BAD_COMMAND_SEQUENCE            0x0000003d // 命令顺序错误
#define MW_ERROR_COMMUNICATION_DATA_LENGTH       0x0000003e // 通信数据长度错误
#define MW_ERROR_ECC                             0x0000003f // 设备 ECC 加密错误
#define MW_ERROR_BAD_UPDATE_DATA_LENGTH          0x00000040 // 升级数据长度错误
#define MW_ERROR_UPDATE_STATE                    0x00000042 // 升级状态错误
#define MW_ERROR_UPDATE_KEY_NOT_ENABLE           0x00000043 // 当使用远程升级，必须先设置远程升级秘钥
#define MW_ERROR_LOCKED_FOREVER                  0x00000044 // 设备永久锁定
#define MW_ERROR_LOCKED_TEMPORARY                0x00000045 // 设备临时锁定
#define MW_ERROR_DEVICE_UNLOCKED                 0x00000046 // 设备未被锁定
#define MW_ERROR_DEVICE_FLASH                    0x00000047 // 设备闪存异常，可能需要更换加密锁
#define MW_ERROR_DEVICE_ERROR                    0x00000048 // 设备错误
#define MW_ERROR_TOO_MANY_DEVICE                 0x00000049 // 设备编号错误，应不大于 128
#define MW_ERROR_DEVICE_NOT_ENOUGH_USER_MEMORY   0x0000004a // 没有足够内存留给用户编码
#define MW_ERROR_FAKE_DEVICE                     0x0000004b // 设备是假的
#define MW_ERROR_DEVICE_BLK_OUT_RANGE            0x0000004c // 设备大量读写超出范围
#define MW_ERROR_DEVICE_FS_ERR_BAK_ERROR         0x0000004d // 设备备份错误
#define MW_ERROR_DEVICE_TMR_ERR_ADJUST_TIME_TIMEOUT 0x0000004e // 调节时间超时
#define MW_ERROR_DEVICE_EXCH_ERROR               0x0000004f // 交换内存错误
#define MW_ERROR_DEVICE_AES_ERR                  0x00000050 // 设备 AES 错误
#define MW_ERROR_DEVICE_HASH_ERR_UNSUPPORTED_ALGO 0x00000051 // 不支持的哈希算法
#define MW_ERROR_DEVICE_BAD_PRIVATE_KEY          0x00000052 // 私钥损坏
#define MW_ERROR_DEVICE_BAD_PUBLIC_KEY           0x00000053 // 公钥损坏
#define MW_ERROR_DEVICE_BAD_SYMMETRIC_KEY        0x00000054 // 对称密钥损坏
#define MW_ERROR_DEVICE_BAD_SIGNATURE            0x00000055 // 签名损坏
#define MW_ERROR_DEVICE_KEY_ERR_BAD_ALGO         0x00000056 // 低劣算法
#define MW_ERROR_DEVICE_LOGO_ERR_LOGO            0x00000057 // 低劣逻辑
#define MW_ERROR_EXEC_PARAM_TOO_LONG             0x00000058 // 执行参数数据过长
#define MW_ERROR_OPEN_ERROR                      0x00000059 // 打开设备错误
#define MW_ERROR_LOAD_SYS_MODULE_ERROR           0x0000005A // 加载系统模块错误
```

```
#define MW_ERROR_SYS_MODULE_FUNCTION_ERROR          0x0000005B // 系统模块函数地址错误
#define MW_ERROR_RSA                                0x0000005C // 设备 RSA 加密错误
#define MW_ERROR_KEY                                0x0000005D // 加密公钥错误
#define MW_ERROR_DEVICE_EXEC_ERR_UNALIGNED_MEM_ADDR 0x0000005E // 未对齐的内存地址
#define MW_ERROR_DEVICE_EXEC_ERR_STACK_OVERFLOW     0x0000005F // 用户栈溢出
#define MW_ERROR_DEVELOPER_ID_MISMATCH              0x00000060 // 开发商 ID 不匹配
#define MW_ERROR_LM_GENERAL_ERROR                   0x00000061 // LM 返回数据格式错误
#define MW_ERROR_LM_UNSUPPORTED_CERT_TYPE           0x00000062 // 不支持的证书类型
#define MW_ERROR_LM_UNSUPPORTED_UPDATE_OBJECT_TYPE  0x00000063 // 不支持的升级对象类型
#define MW_ERROR_LM_UPDATE_PKG_FORMAT_WRONG         0x00000064 // 升级包格式错误
#define MW_ERROR_CERT                               0x00000065 // 证书错误
#define MW_ERROR_DEX_NOT_LOADED                     0x00000066 // 动态执行不加载到锁
#define MW_ERROR_SYSAPP_NOT_FOUND                   0x00000067 // 未发现系统程序
#define MW_ERROR_UNSUPPORTED_OBJECT_TYPE            0x00000068 // 不支持的对象类型
#define MW_ERROR_UPDATE_BLOCK_LENGTH                0x00000069 // 升级区长度错误
#define MW_ERROR_UPDATE_SEED_NOT_FOUND              0x0000006A // 未发现升级种子
#define MW_ERROR_HEAP                               0x0000006B // 锁内系统堆崩溃
#define MW_ERROR_DEX_OUT_OF_RANGE                   0x0000006C // 动态执行超出范围
#define MW_ERROR_DEX_TOO_LARGE                      0x0000006D // 动态执行过大
#define MW_ERROR_UPDATE_INFO                        0x0000006E // 升级信息错误
#define MW_ERROR_DEVICE_STACK_OVERFLOW              0x0000006F // 锁内系统栈崩溃
#define MW_ERROR_COMMUNICATION_CRYPT                0x00000070 // 通信加密错误
#define MW_ERROR_BAD_UPDATE_PKG                     0x00000071 // 升级包损坏
#define MW_ERROR_UPDATE_PKG_VERSION_LOW             0x00000072 // 升级包版本过低
#define MW_ERROR_UPDATE_OBJECT_TYPE                 0x00000073 // 升级对象类型错误
#define MW_ERROR_UPDATE_DEVELOPER_ID                0x00000074 // 错误的开发商 ID
#define MW_ERROR_UPDATE_FILE_TYPE                   0x00000075 // 错误的文件类型
#define MW_ERROR_NO_ADJUST_TIME_REQUEST             0x00000076 // 未获得调节时间请求
#define MW_ERROR_CERT_REVOKED                       0x00000077 // 证书被吊销
#define MW_ERROR_WRONG_CERT_SLOT                    0x00000078 // 证书槽号错误
#define MW_ERROR_DEVICE_HASH_ERR_NOT_MATCH          0x00000079 // 哈希值结果不匹配
#define MW_ERROR_BAD_RND                            0x0000007A // 随机值已损坏
#define MW_ERROR_RTC                                0x0000007B // 硬件时钟错误，请更换时钟锁
#define MW_ERROR_RTC_POWER                          0x0000007C // 硬件时钟电池掉电，请更换时钟锁

#define MW_ERROR_NO_PID_DEVICE                      0x00002001 // 主机上找不到指定的 PID 魔锐锁
#define MW_ERROR_RET_DATA_FORMAT                    0x00002002 // 返加数据格式不正确
#define MW_ERROR_FILE_NAME_LEN                      0x00002003 // 文件名长度不正确
#define MW_ERROR_BLOCK_DATA                         0x00002004 // 数据块错误
#define MW_ERROR_FILE_PROPERTY_PARAM                0x00002005 // 文件属性参数错误
#define MW_ERROR_FILE_COUNT_MAX                     0x00002006 // 实际文件个数超过预分配的个数
#define MW_ERROR_MALLOC_MEMORY                      0x00002007 // 分配内存失败
#define MW_ERROR_PIN_LENGTH                         0x00002008 // PIN 长度有错误
#define MW_ERROR_UPDATA_PKG_PID                     0x00002009 // 升级包 PID 与锁内 PID 不一致
```

```
#define MW_ERROR_ENTRY_TYPE                0x0000A001 // entry 类型错误
#define MW_ERROR_ENTRY_FLAG                0x0000A002 // entry 标识错误
#define MW_ERROR_OPCODE                    0x0000A003 // 错误命令
#define MW_ERROR_PARAM                     0x0000A004 // 错误参数
#define MW_ERROR_DATA_LENGTH               0x0000A005 // 错误数据长度
#define MW_ERROR_PIN                       0x0000A006 // 错误 PIN 码
#define MW_ERROR_DEVICE_BLOCK              0x0000A007 // 设备 PIN 码锁死
#define MW_ERROR_PRIVILEGE                 0x0000A008 // 没有权限
#define MW_ERROR_DATA_INFO                 0x0000A009 // 数据域数据错误
#define MW_ERROR_HASH_TYPE                 0x0000A00A // 错误 HASH 类型
#define MW_ERROR_ALG_MODE                  0x0000A00B // 错误的模式
#define MW_ERROR_INIT_EVT                  0x0000A00C // 初始化过程问题错误
#define MW_ERROR_ALG_TYPE                  0x0000A00D // 错误算法类型
#define MW_ERROR_BIT_SIZE                  0x0000A00E // 错误的密钥位大小
#define MW_ERROR_FILE_TYPE                 0x0000A00F // 错误的文件类型
#define MW_ERROR_NO_INIT_PID               0x0000A010 // PID 未初始化
#define MW_ERROR_NO_INIT_PIN               0x0000A011 // PIN 码未修改
#define MW_ERROR_NO_VALIDATE_PKG           0x0000A012 // 无效升级包
#define MW_ERROR_AUTH_STAGE                0x0000A013 // 权限价段错误
#define MW_ERROR_FILE_PRIVILEGE            0x0000A014 // 文件权限错误
#define MW_ERROR_FILE_RANGE                0x0000A015 // 文件范围错误
#define MW_ERROR_KEY_FILE                  0x0000A016 // 文件密钥错误
#define MW_ERROR_REQ_IDENTIFY              0x0000A017 // 返回认证错误
#define MW_ERROR_INVALID_SEED              0x0000A018 // 无效种子错误
#define MW_ERROR_SEED_CHECK                0x0000A019 // 种子校验错误
#define MW_ERROR_UPDATE_HASH               0x0000A01A // 升级哈希错误
#define MW_ERROR_UPDATE_PID                0x0000A01B // 升级 PID 错误
#define MW_ERROR_UPDATE_INDEX              0x0000A01C // 升级索引错误
#define MW_ERROR_UPDATE_TIME               0x0000A01D // 升级时间错误
#define MW_ERROR_UPDATE_SERIAL_NUM         0x0000A01E // 升级序列号错误
#define MW_ERROR_UPDATE_SN                 0x0000A01F // 升级 SN 错误
#define MW_ERROR_UPDATE_ORDER              0x0000A020 // 升级顺序错误
#define MW_ERROR_UPDATE_OPERATE            0x0000A021 // 长级操作错误
#define MW_ERROR_KEY_TYPE                  0x0000A022 // 关键字类型错误
#define MW_ERROR_OTHER                     0x0000AFFF // 其他错误
```