API Reference

Moway 1

Version 1.0.6.0

Copyright & Trademarks

Disclaimer

All attempts have been made to make the information in this document complete and accurate. But we cannot guarantee everything is perfect, we will correct it in next version released in case some error has been found. Senseshield is not responsible for any direct or indirect damages or loss of business resulted from inaccuracies or omissions.

The specifications contained in this document are subject to change without notice.

Documentation Improvement

Any suggestion to this manual from you are welcome, we are glad to hear any feedback from you which will help us to continuously improve the documents quality and support and serve the developer to protect software products more efficiently.

Contact

Company: Beijing Senseshield Technology Co., Ltd

Address: Suite 510, Block C, Internet Innovation Center, Building 5, No.10, Xibeiwang East Road, Haidian District, Beijing China

Tel: +86-10-56730936

Fax: +86-10-56730936-8007

Sales: info@senselock.com;

Website: https://lm-global.virbox.com/

Virbox Developer Center Website: https://developer.lm-global.virbox.com/

About this Manual

Objective

This manual is designed to help software developer to learn about the principle, protection scheme of Moway 1 and how to use Moway 1 and associated tools to design the software protection scheme and protect their software.

Notes

Caution: The content with this mark in the manual indicates that you need to pay highly attention, otherwise may cause serious consequences.

Mark: The content with this mark in the manual indicates that you need to pay attention to read.

Table of Contents

1 Constant Definition

1.1 Open Moway device

Macro	Value	Description
MW_OPEN_SHARE_MODE	0x00	Open Moway device with share mode
MW_OPEN_EXCLUSIVE_MODE	0x01	Open Moway device with exclusive mode

1.2 PIN type

Macro	Value	description
MW_PIN_TYPE_NONE	0x00	PIN type: Null
MW_PIN_TYPE_USER	0x01	PIN type: User type
MW_PIN_TYPE_DEVELOPER	0x02	PIN type: Developer type

1.3 PIN length

Macro	Value	Description
MW_PIN_LENGTH_USER	8	The length of user PIN
MW_PIN_LENGTH_DEVELOPER	24	The length of Developer PIN (Master PIN)

1.4 File Type

Macro	Value	Description
MW_FILE_TYPE_BINARY	0x00	Binary data file
MW_FILE_TYPE_KEY	0x02	Key file

1.5 File Operation Privilege

Macro	Value	Description
MW_FILE_PRIV_TYPE_USE	0x00	Key file type, Developer & User use only
MW_FILE_PRIV_TYPE_READ	0x01	Read only file, read only with user permission
MW_FILE_PRIV_TYPE_READ_WRITE	0x02	Read & Write file, can be read & write with user permission.

1.6 Algorithm Type

Macro	Value	Description
MW_ALG_TYPE_AES	0x00	AES Algorithm
MW_ALG_TYPE_DES	0x01	DES Algorithm
MW_ALG_TYPE_TDES	0x02	TDES Algorithm
MW_ALG_TYPE_ECC	0x10	ECC Algorithm
MW_ALG_TYPE_RSA	0x11	RSA Algorithm

Note: for symmetric cryptography & Asymmetric cryptography use

1.7 Symmetric Algorithm

Macro	Value	Description
MW_SYM_ALGO_MODE_ECB	0x00	Symmetric algorithm ECB mode
MW_SYM_ALGO_MODE_CBC	0x01	Symmetric algorithm CBC mode

Note: for symmetric cryptography use.

1.8 RSA Encryption/Decryption Padding Mode

Macro	Value	Description
MW_RSA_MODE_NORMAL	0x00	No padding mode
MW_RSA_MODE_PKCS1_V1_5	0x01	PKCS1 V1_5 padding mode

1.9 RSA Algorithm Length

Macro	Value	Description
MW_RSA_1024_BYTE_SIZE	128	RSA1024 calculation, the byte length of Mod
MW_RSA_2048_BYTE_SIZE	256	RSA2048 calculation, the byte length of Mod

1.10 The Type of HASH signature

Macro	Value	Description
MW_HASH_ALGO_MD5	0x00	Signature MD5 hash algorithm
MW_HASH_ALGO_SHA1	0x01	Signature SHA1 hash algorithm
MW_HASH_ALGO_SHA256	0x02	Signature SHA256 hash algorithm
MW_HASH_ALGO_NONE	0xFF	No hash calculation when signed, Hash calculation by caller

1.11 HMAC Digest Length

Macro	Value	Description
MW_HMAC_MD5_DIGEST_LENGTH	16	The length of HMAC- MD5 algorithm digest
MW_HMAC_SHA1_DIGEST_LENGTH	20	The length of HMAC - SHA1 algorithm digest
MW_HMAC_SHA256_DIGEST_LENGTH	32	The length of HMAC - SHA256 algorithm digest

1.12 Multiple Language ID

Macro	Value	Description
MW_LANGUAGE_ID_DEFAULT	0x00	On default Chinese
MW_LANGUAGE_ID_CH	0x01	Chinese
MW_LANGUAGE_ID_EN	0x02	English

Note: for Error help function used

1.13 Macro related of length/size definition

Macro	Value	description
MW_PATH_LEN	1024	The maximum length of Moway device path.
MW_SYM_ALG_IV_LENGTH	16	The length of IV (Initialization vector) CBC mode of device symmetric cryptographic algorithm,
MW_SN_LENGTH	16	The Length of unique Serial Number of Moway device
MW_FILE_NAME_LENGTH_MAX	16	The maximum length of device file name.
MW_INPUT_DATA_LENGTH_MAX	1900	The maximum length of the input data to Moway device to execute.
MW_SEED_LENGTH_MAX	32	The maximum length of seed
MW_SEED_LENGTH_MIN	4	The minimum length of seed
MW_ENUM_DEVICE_MAX_COUNT	128	The maximum number of device in host machine

1.14 Device initialized status

Macro	Value	Description
MW_STATUS_FLAG_DEFAULT	0x00	Moway device in factory setting status (default status), both PID and Developer PIN (Master PIN) not modified yet.
MW_STATUS_FLAG_PID	0x01	Device default PID has been modified (a flag to PID has been set)
MW_STATUS_FLAG_PIN	0x02	Device default Developer PIN has been modified (a flag to developer PIN has been set)

1.15 Get Device information

Macro	Value	Description
MW_GET_INFO_ITEM_PID	0x00	Get Moway device PID， The PID length is 4

Macro	Value	Description
		bytes
MW_GET_INFO_ITEM_SN	0x01	Get the unique Serial number of device, the length is 16 bytes
MW_GET_INFO_ITEM_PRODUCE_DATE	0x02	Get the device production date, the length is 4 bytes
MW_GET_INFO_ITEM_ALL_CAPACITY	0x03	Get total capacity of device, the length is 4 bytes
MW_GET_INFO_ITEM_FREE_CAPACITY	0x04	Get the device available capacity, the length is 4 bytes
MW_GET_INFO_ITEM_VERSION	0x05	Get device version information, the length is 4 bytes
MW_GET_INFO_ITEM_SHELL_SN	0x06	Get the device shell ID, the first bytes present the length
MW_GET_INFO_ITEM_STATUS	0x07	Get the device initialized status.
MW_GET_INFO_ITEM_UPDATE_CODE	0x08	Get device the update code
MW_GET_INFO_ITEM_UPDATE_CTRL_CODE	0x09	Get the update device type of information
MW_GET_INFO_ITEM_ALL	0xFF	Get all of device information

1.16 Control code

Macro	Value	Description
MW_CTRL_CODE_ITEM_RESET	0x00	Reset device status, when device has been reset, the device restore and back to factory setting and all privilege has been cleared.
MW_CTRL_CODE_ITEM_LED	0x01	Control the device LED light flash status: turn on/off

2. Type of Definition

2.1 Device handle

```
typedef        void*        MWHANDLE;
```

2.2 Device information structure

```
typedef struct _MW_DEVICE_INFO_CTX{
      unsigned int       uiPID;                              //    Product ID
      unsigned char      ucSerialNum[MW_SN_LENGTH];    // Unique serial number
      unsigned char      ucDevPath[MW_PATH_LEN];    // The device path
      unsigned int       uiProtocol;                         // Communication protocol
       unsigned int       uiLocationID;                      // Mac OS system Location ID
}MW_DEVICE_INFO_CTX;
```

2.3 File property structure

```
typedef struct _MW_FILE_PROPERTY{
            unsigned char       ucValidate;              // Reserved
            unsigned char       ucType;                 // File type: binary file or key file
            unsigned short      usPrivilege;             // File privilege
            unsigned int        uiSize;                  // file size
            unsigned int        uiTime;                  // create time
            char acName[MW_FILE_NAME_LENGTH_MAX + 1];      // file name
}MW_FILE_PROPERTY;
```

2.4 System working status structure

```
typedef struct _MW_DEVICE_STATUS
{
            unsigned int       uiSysTime;               // System time
            unsigned int       uiSysStatus;              // System status
```

```
    unsigned short    usReserved;          // Reserved
    unsigned char     ucLoginStatus;       // Login status
    unsigned char     ucLedStatus;         // LED light status
}MW_DEVICE_STATUS;
```

2.5 Remote Update Structure

```
typedef struct _MW_UPDATE_FILE_CTX
{
    MW_FILE_PROPERTY stFileProperty;       // File property structure
    unsigned char *pData;                  // The update file data
    unsigned int uiDateSize;               //    The update file size
}MW_UPDATE_FILE_CTX;
```

2.6 Device information item structure

```
{
    unsigned int uiPID;                    // Product ID (PID)
    unsigned char acSN[MW_SN_LENGTH];      // Unique serial number
    unsigned int uiProduceDate;            // Production time
    unsigned int uiAllCapacity;            // Device total capacity
    unsigned int uiFreeCapacity;           // Available capacity
    unsigned int uiVersion;                // Version info
    unsigned int uiUpdateCode;             // Update code
    unsigned char ucStatus;                // Status
    unsigned char ucUpdateCtrlCode;        // Update control code
    unsigned short reserved;               // Reserved
}MW_DEVICE_ALL_ITEM_INFO;
```

3. API Description

3.1 mw_enum

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

Parameter definition：

pDevInfoList	Enumerate the structure data memory address which the Moway device output, it is required API caller to allocate the memory.
uiDevListCount	The number of structure data of device list which to allocate the (pDevInfoList) memory to store. Note: if you don't know the number of device currently, you may apply for the maximum number of device (Refer the Macro definition: MW_ENUM_DEVICE_MAX_COUNT)
puiDevCount	Returned with the number of valid Moway device enumerated, i.e., the number of valid device structure in the "pDevInfoList"

Return value description：

When success to execute, return: MW_SUCCESS
For other value returned, please refer the "error code" accordingly

API description：

Enumerate all of Moway device in current machine.
To call this API, it is require user to assign the memory to "pDevInfoList" according to the "uiDevListCount" multiply by "sizeof(MW_DEVICE_INFO_CTX)", the API internal does not assign memory to "pDevInfoList".
When the device memory assigned less than the actual device memory, returned with the number of device memory assigned.
When the device memory assigned more than the actual device memory, returned with the number of actual device memory.

3.2 mw_open

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

Parameter definition:

pDevInfo	The pointer to the Moway device information structure (One of the device structure in the "pDevInfoList" which enumerated by "mw_enum")
uiShareMode	Device open mode; include following 2 modes: Open with exclusive mode(MW_OPEN_EXCLUSIVE_MODE) and Open with share mode （MW_OPEN_SHARE_MODE)
phHandle	Device handle, all of operation to device will be proceed via this handle.

Return value description:

Success returned with: MW_SUCCESS.
For other value returned, please refer the "error code" accordingly

API description:

Open the Specified Moway device, currently support to open Moway device with exclusive mode only.

The Moway device structure obtained by enumerate API, which the device is one of the device structure in the device list.

3.3 mw_close

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

Parameter definition:

hHandle	Moway device handle， which obtained by API: mw_open

Return value description:

When success, return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description：

Close the specified Moway device

3.4 mw_verify_pin

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

Parameter definition：

hHandle	Moway device handle， which obtained by “mw_open” API
ucPinType	Pin Type, please refer macro definition: MW_PIN_TYPE_XXX. Note: "XXX" here represents several ASCII code wildcards in this document, which indicating that there are several types to choose from.
pucPin	Pin data， for User PIN is 8 bytes, Developer PIN (Master PIN) is 24 bytes

Return value description：

Success return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description:

Verify PIN； When PIN verified, user will obtained the correspondence permission to operate the device. Without plugin/out the Moway device, the permission to operate device is always valid. You can reset, clean the existed permission to device by API Reset API: mw_control

When you set the maximum count error limitation to PIN code input, if the number of input error is out of limitation set, it will lock the device.

For developer PIN (Master PIN, total 24 bytes), when set the limitation to error input, only the last 8 bytes PIN error will effective to error count limitation, otherwise the error count set will not be valid.

For User PIN error count limitation, all 8 bytes error input will be counted error.

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);
```

Parameter definition：

hHandle	Moway device handle， which obtained via API: mw_open
ucPinType	Pin Type, please refer macro definition: MW_PIN_TYPE_XXX
usLimitCount	The maximum error count to input the PIN, Note: if you do not want to set limitation to error input, then set the limit with 0, The range to set the maximum error count to input varies from: 1~15, and “parameter error” will be returned if you set to other value.
pucOldPin	Old Pin Data, User PIN 8 bytes, Developer PIN is 24 bytes
pucNewPin	New Pin data, User PIN 8 bytes, Developer PIN is 24 bytes

Return value description:

Success returned with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description:

Modify the PIN; This operation require the Developer or User Permission. it is mandatory operation to initialize and modify Developer PIN (Master PIN) when

Moway device deliver to developer, otherwise, cryptographic function to Moway device may not available. Use the Developer permission can modify the user PIN directly. in such circumstance, The setting to "pucOldPIN" will be invalid.

With Developer PIN (Master PIN), you will access the super administration permission, please keep the Developer PIN in safety; if the error count set with less, it is much easier to lock the Moway device if continuously input error. If the Moway device locked, nobody can restore the Developer PIN again. So, it is recommend to set the error count with reasonable number.

When developer received the Moway device, please modify the “Developer PIN (Master PIN) and set the PID first. Otherwise, you may not use Moway device to store the PID, key file or other license data file.

3.6 mw_set_pid

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

Parameter definition:

hHandle	Moway Device handle， Get the handle via API “mw_open”
pucPIDSeed	Seed to generate the PID
uiSeedLen	The length of seed code, the length range is in between 4 bytes~32 bytes, please refer the macro definition: MW_SEED_LENGTH_MIN, MW_SEED_LENGTH_MAX.

Return value description:

Success return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description：

Set Product ID: PID,

This operation require the Developer permission (With Developer PIN verification, i.e. Master PIN). It is mandatory to initialize the Moway device and set PID after Moway shipment from factory. Otherwise developer may not use Moway device accordingly.

When developer get the Moway device, please modify Developer PIN (Master PIN) and set PID first (initialization the Moway device), otherwise you can not use Moway device to store the PID, key file or other data file.

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);
```

Option description:

hHandle	Moway device handle，which obtained via API: mw_open
uiCtrlCodeItem	Control code, please refer the macro definition: MW_CTRL_CODE_ITEM_XXX
pvInBuffer	Input data
uiInBufferLen	The length of input data
pvOutBuffer	The output data
uiOutBufferLen	The length of output data
puiReturnedLen	The length of actual output data

Return value description:

Success then returned: MW_SUCCESS.

For other value returned, please refer the ”error code” accordingly

API description

Control command;

The operation to reset by control command can be used to clean current device permission status.

Control the LED light supports LED light Turn on/off only.

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);
```

Parameter definition:

hHandle	Moway device handle, which obtain via API: mw_open
ucInfoItem	To specify the information items to get from device, please refer the Macro definition: MW_GET_INFO_ITEM_XXX
pvBuffer	Output (return) the information items obtained from device, and the buffer size depends on the size of information items obtained, please refer the related macro definition of device size/length.
puiBufferLength	The data length of information items output

Return value description:

Success then return with: MW_SUCCESS.

For other value returned, please refer the ”error code” accordingly.

API description:

Get the Moway device information;

With this API, you can get the device information:

Serial number,

Production date,

Version information and

Capacity information etc.

3.9 mw_get_device_status

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

Parameter definition:

hHandle	Moway device handle, which obtained via API: mw_open
pstDeviceStatusCtx	Current device status structure, please refer the macro definition: MW_DEVICE_STATUS

Return value description:

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

Get the device current status; With this API, developer may get the device current status and permission status.

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);
```

Parameter definition

hHandle	Moway device handle which obtained via API: mw_open
pstFilePropertyList	Structure array: MW_FILE_PROPERTY which size(length) specified by: uiFileListCount
uiFileListCount	To specify the size of array: pstFilePropertyList
puiReturnedFileCount	The number of file actual enumerated.

Return value description

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description:

Enumerate the file in the device;

To operate this API require the developer or user permission;

pstFilePropertyListassigned by functions caller from the external, and the size is: uiFileListCount * sizeof(MW_FILE_PROPERTY);

For Moway device with 8K capacity, the maximum number of file support is 16 files,

For Moway device with 32K capacity, the maximum number of file supported is 64 files.

3.11 mw_create_file

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

Parameter definition:

hHandle	Moway device handle which obtained by API: mw_open
pstFileProperty	To specify the file property which need to be created, in which file type, file permission, file size and file name is required. Other is option.

Return value description:

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description:

Create File

This operation require developer permission;

File type support:

Key file type (MW_FILE_TYPE_KEY),

Binary file type (MW_FILE_TYPE_BINARY).

File permission can be defined:

Key file permission,

Read only permission,

Read & write permission

in which key file corresponds to key file permission,

The binary file corresponds to read-only and read-write permissions.

The length of Filename will be limited within 16 bytes.

For Moway device with 8K capacity, the maximum number of file support is 16 files, for Moway device with 32K capacity, the maximum number of file supported is 64 files.

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);
```

Option description:

hHandle	Moway device handle, which obtained by API: mw_open
pcFileName	File name, the string which ends with '\0'
uiReadOffset	The offset of the read file
uiReadSize	the size of the read file
pucReadBuffer	Read the file to buffer Note: it is required to allocate sufficient size to "uiReadSize", otherwise it will out of size when read the file.

Return value description

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description:

Read file data;

This operation require developer's or user's permission;

With developer and user permission, it can read all of data file; No any of permission can read the key file;

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);

Parameter definition:

hHandle	Moway device handle which obtained via API: mw_open
pcFileName	File name, the string which ended with '\0'
uiWriteOffset	Address offset to write the file
uiWriteSize	The size to write file
pucWriteBuffer	The buffer that need to write to the file.

Return value description:

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description:

Write file to device

This operation require developer's permission;

With the user's permission, it can only to write the data file which for user's permission to write;

For other data file, with user's permission, it can only can read the data file;

With Developer's permission, it can write all kind of data file;

For key file, it can be written in one times, and it can not be written with offset.

3.14 mw_delete_file

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

参数说明:

hHandle	Moway device handle which obtained by API: mw_open
pcFileName	File name, the string which ended with '\0'

Return value description:

Success then returned with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

Delete the file This operation require developer permission.

3.15 mw_get_file_property

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

Parameter definition

hHandle	Moway device handle which obtained via API: mw_open
pcFileName	File name, the string which ended with '\0'
pstFileProperty	The file property’s structure data input

Return value description:

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

To get the file property of the specified file. This operation require developer or user permission

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);
```

Option description:

hHandle	Moway device handle, which obtained via API: mw_open
pcKeyFileName	Key file name, the string which ended with '\0'
ucAlgoMode	Symmetric encryption algorithm mode, you can set with 2 symmetric algorithms: please refer: MW_SYM_ALGO_MODE_XXX
pucIV	Vector required to be input in Symmetric encryption algorithm, in CBC mode, For DES and TDES algorithm, the length is 8 bytes, for AES algorithm, the length is fixed with 16 bytes, in ECB mode, the vector can be option.
pucInData	The plaintext input to be encrypted, it must be a multiple of 16.
uiInDataLen	The data length of the plaintext to be encrypted.
pucOutData	The cyphertext output after encrypted.
puiOutDataLen	The data length of ciphertext after encrypted.

Returned value description:

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

Symmetric encryption

This operation require developer or user permission; Key file need to be consistent with the symmetric algorithm type: DES, TDES, AES type, Otherwise, error will be output. The input data must be multiple of 16.

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);
```

Parameter definition

hHandle	Moway device handle which obtained via API: mw_open
pcKeyFileName	Key file name, the string which ended with '\0'
ucAlgoMode	Symmetric encryption algorithm, please refer the macro definiation: MW_SYM_ALGO_MODE_XXX
pucIV	Vector required to be input in Symmetric encryption algorithm, in CBC mode, forDES and TDES algorithm, the length is 8 bytes, for AES algorithm, the length is fixed with 16 bytes, in ECB mode, the vector can be optional.
pucInData	The cyphertext input must be a multiple of 16.
uiInDataLen	The data length of the cyphertext input
pucOutData	The plaintext output when decrypted
puiOutDataLen	The length of output plaintext when decrypted.

Return value description

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

Symmetric decryption This operation require developer's or user's permission; The API used need to be consistent with the key type: DES,TDES， AES. If the type inconsistent, error will be output. The input data must be multiple of 16.

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);
```

Parameter definition

hHandle	Moway device handle which obtained via API: mw_open
pcKeyFileName	Key File name, the string which ended with '\0'
ucPadMode	The encryption algorithm, please refer macro definition: MW_RSA_MODE_XXX
pucInData	The plaintext data input
uiInDataLen	The data length of plaintext input
pucOutData	The output cyphertext data
puiOutDataLen	The data length output cyphertext

Return value description

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

Rsa Asymmetric encryption

This operation require developer's or user's permission;

The key file must be the RSA key, otherwise error will be returned.

When encryption mode: "ucPadMode" set to be "MW_RSA_MODE_NORMAL", The data length of plaintext to be encrypted with RSA_1024 algorithm must be with the "MW_RSA_1024_BYTE_SIZE", and the data length of plaintext to be encrypted with RSA_2048 encryption algorithm must be with the "MW_RSA_2048_BYTE_SIZE";

When "ucPadMode" encryption mode set to be "MW_RSA_MODE_PKCS1_V1_5", the maximum of plaintext to be encrypted with RSA_1024 is: MW_RSA_1024_BYTE_SIZE-11, and the maximum of plaintext to be encrypted with RSA_2048 is: 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);
```

Parameter definition

hHandle	Moway device handle which obtained via API: mw_open
pcKeyFileName	Key File name, the string which ended with '\0'
ucPadMode	The encryption algorithm, please refer macro definition: MW_RSA_MODE_XXX
pucInData	The cypher text input
uiInDataLen	The length of cyphertext input, note: the length must be: 128 "MW_RSA_1024_BYTE_SIZE" or 256 "MW_RSA_2048_BYTE_SIZE"
pucOutData	The plaintext output
puiOutDataLen	The length of plaintext output

Return value description:

Success then return with: MW_SUCCESS.

For other value returned, please refer the ”error code” accordingly

API description:

Rsa, Asymmetric decryption;

This operation require developer's or user's permission;

The key file must be the RSA key, otherwise error will be returned.

When decryption mode: "ucPadMode" set to be "MW_RSA_MODE_NORMAL", The data length of plaintext to be decrypted with RSA_1024 algorithm must be with the "MW_RSA_1024_BYTE_SIZE", and the data length of plaintext to be decrypted with RSA_2048 encryption algorithm must be with the "MW_RSA_2048_BYTE_SIZE";

When "ucPadMode" encryption mode set to be "MW_RSA_MODE_PKCS1_V1_5", the maximum of plaintext to be decrypted with RSA_1024 is: MW_RSA_1024_BYTE_SIZE-11, and the maximum of plaintext to be decrypted with RSA_2048 is: 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);
```

Parameter definition:

hHandle	Moway device handle, which obtained via API: mw_open
pcKeyFileName	Key File name, the string which ended with '\0'
ucHashAlg	Hash algorithm type, please refer the macro definition: MW_HASH_ALGO_SHA1 MW_HASH_ALGO_SHA256 MW_HASH_ALGO_NONE
pucMessageData	The input message data
uiMessageDataLen	The length of message data,

	Note: the maximum data length, please refer the macro definition: MW_INPUT_DATA_LENGTH_MAX
pucSignData	The output signing data
puiSignDataLen	The length of signing data output. The maximal data length is 'MW_RSA_2048_BYTE_SIZE'

Return value description

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description:

Data signed

This operation require developer's or user's permission;

The key file must be RSA or ECC key file, otherwise it will return with error;

When `ucHashAlg` set to `MW_HASH_ALGO_NONE`,

The length of data signed by `RSA_1024` algorithm will not exceed 125 bytes(MW_RSA_1024_BYTE_SIZE - 3);

The length of data signed by `RSA_2048` will not exceed 253 bytes (MW_RSA_1024_BYTE_SIZE - 3);

Except for the above cases, the data length of the signing data in the remaining cases shall not exceed 1644 bytes(MW_INPUT_DATA_LENGTH_MAX-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);
```

Parameter definition:

hHandle	Moway device handle which obtained via API: mw_open

pcKeyFileName	Key File name, the string which ended with '\0'
ucHashAlg	Hash Algorithm type, please refer the macro definition: MW_HASH_ALGO_SHA1, MW_HASH_ALGO_SHA256, MW_HASH_ALGO_NONE
pucSignData	Input signed data
uiSignDataLen	The length of input signed data, Note: the maximum data length is: `MW_RSA_2048_BYTE_SIZE`
pucMessageData	Input message data
uiMessageDataLen	The length of input message data

Return value description:

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

Verify the data signed This operation require developer's or user's permission; The key file must be the RSA or ECC key file, otherwise error will be returned;

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);
```

Parameter definition:

hHandle	Moway device handle which obtained via API: mw_open
pcKeyFileName	Key File name, the string which ended with '\0'
pucInData	The input data
uiInDataLen	The length of input data

pucOutData	The output digest data
puiOutDataLen	The length of output digest data, please refer the definition: MW_HMAC_MD5_DIGEST_LENGTH 、MW_HMAC_SHA1_DIGEST_LENGTH 、MW_HMAC_SHA256_DIGEST_LENGTH

Return value description:

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

HMAC calculation; This operation require developer's or user's permission; The key file must be the HMAC key file, otherwise error will be returned; The length of data return will be varies depending on the hash calculation algorithm of HMAC key type selected, the length are different for different algorithm selected.

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_UPDATE_FILE_CTX  *pstUpdateFileList,
                                      IN     unsigned int        uiFileCount,
                                      OUT    unsigned char       *pucOutPkg,
                                      INOUT  unsigned int        *puiOutPkgLen);
```

Option description:

hHandle	Moway device handle which obtained via API: mw_open
uiDevPID	The product ID (PID) of device
pucSN	The unique Moway hardware serial number (S/N) with fix length: `MW_SN_LENGTH`; When binding device but not specify the S/N, it can be with `NULL`

pstUpdateFileList	The Update file list, the maximal number of Update file need to refer the definition: MW_UPDATE_FILE_COUNT_MAX, and the number of Update file is: uiFileCount.
uiFileCount	The number of Update file, which defined the size of “pstUpdateFileList”
pucOutPkg	The update file package data
puiOutPkgLen	The length of update file package data

Return value description:

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

Create update file package by developer; This operation require Developer's permission; Prerequisite: create the update file package with the “Master lock”.

3.24 mw_update

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

Parameter definition:

hHandle	Moway device handle, which obtained via API: mw_open
pucInPkg	Update file package data
uiInPkgLen	The length of update file package data

Return value description

Success then return with: MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

Remote update the update file package in user premise; The operation only require user permission.

3.25 mw_restore_factory

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

Parameter definition:

hHandle	Moway device handle which obtained via API: mw_open

Return value description

Success then return with:MW_SUCCESS.
For other value returned, please refer the ”error code” accordingly

API description

To restore the Moway factory setting

This operation require developer permission

To restore the Moway device to factory setting, that means, Restore the default PID, default PIN, and the file system is cleared also. After restoring to default setting, re-initialization is required to use device related functions.

When restore the Moway device to factory setting, all of device information will back to factory setting on default, all of file in the Moway device will be deleted, so, please double confirm before execute this operation.

3.26 mw_get_error_desc

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

Parameter definition:

uiErrorCode	Moway error code
uiLanguageID	String language type ID of error code parsing returned.

Return value description

Returns the error code parsing string in the specified language

API description

Parse the error code;

4. Error code

```
#define MW_SUCCESS                          0x00000000 // Success
#define MW_ERROR_INVALID_HANDLE             0x00000001 // handle may be null or empty.
#define MW_ERROR_INVALID_PARAMETER          0x00000002 // Invalid parameter
#define MW_ERROR_NOT_ENOUGH_MEMORY          0x00000003 // No sufficient capacity to proceed this operation
#define MW_ERROR_NO_DEVICE                  0x00000004 // no device or no specified device found
#define MW_ERROR_TIMEOUT                    0x00000005 // timeout for communication with Device
#define MW_ERROR_UNSUPPORTED_FLAG           0x00000006 // Unsupported flag passed to API
#define MW_ERROR_INSUFFICIENT_BUFFER        0x00000007 // Insufficient buffer to store data
#define MW_ERROR_EXCHG_MEMORY_NOT_FOUND     0x00000008 // Not found the specified share memory
#define MW_ERROR_SYSTEM_FILE_NOT_FOUND      0x00000009 // No file has been found in the system
#define MW_ERROR_SYSTEM_FILE_INVALID_ACCESS 0x0000000a // Failed to access the system file
#define MW_ERROR_FILE_EXISTS                0x0000000b // The specified file existed.
#define MW_ERROR_FILE_NOT_FOUND             0x0000000c // The specified file not found
#define MW_ERROR_NO_PRIVILEGE               0x0000000d // No privilege to operate
#define MW_ERROR_WRONG_PASSWORD             0x0000000e // Error password
#define MW_ERROR_PASSWORD_LOCKED            0x0000000f // The password has been locked
#define MW_ERROR_FUNCTION_NOT_SUPPORTED     0x00000011 // Not support this function
#define MW_ERROR_COMMUNICATION              0x00000015 // Failed communication with USB port, may restart computer
#define MW_ERROR_UNSUPPORTED_PASSWORD_TYPE  0x00000016 // Password type error.
#define MW_ERROR_WRONG_PASSWORD_LENGTH      0x00000017 // The length of PIN error, developer PIN is 24 bytes, User PIN
is 8 bytes
#define MW_ERROR_ALREADY_EXCLUSIVELY_LOGIN   0x00000018 // Login with exclusive mode and failed to login with share
mode.
#define MW_ERROR_ALREADY_SHARED_LOGIN        0x00000019 // Login with share mode, and failed to login with exclusive
mode.
#define MW_ERROR_ALREADY_TEMP_EXCLUSIVELY_USING 0x0000001a // Handle has been used with exclusive in temporarily.
#define MW_ERROR_NOT_TEMP_EXCLUSIVELY_USING 0x0000001b // Handle has been used with exclusive not in temporarily.
#define MW_ERROR_TOO_MUCH_DATA              0x0000001c // out of the length of the message response functions
#define MW_ERROR_INSUFFICIENT_DEVICE_SPACE  0x0000001e // Insufficient device capacity.
#define MW_ERROR_DEVICE_FILESYSTEM_ERROR    0x0000001f // Device file system error
#define MW_ERROR_FILE_OUT_RANGE             0x00000020 // Device file out of range
#define MW_ERROR_UNSUPPORTED_FILE_TYPE      0x00000021 // The file type doesn't support which passed to EV API
#define MW_ERROR_FILE_OFFSET_MUST_BE_ZERO   0x00000022 // To write or read the key file, offset must be 0
#define MW_ERROR_BAD_EXECUTIVE_FILE_FORMAT  0x00000023 // Executive file format error
```

```
#define MW_ERROR_OPEN_TOO_MANY_DEVICE_FILES          0x00000024 // Too many device file opened
#define MW_ERROR_INVALID_DEVICE_FILE_HANDLE          0x00000025 // Invalid Device file handle
#define MW_ERROR_FILE_NOT_FINISHED                   0x00000026 // File note complete
#define MW_ERROR_BAD_FILENAME                        0x00000027 // File name error
#define MW_ERROR_BAD_NAME                            0x00000028 // Invalid file name, directory name or volume syntax
#define MW_ERROR_DEVICE_TIMER                        0x00000029 // Invalid device time
#define MW_ERROR_NO_EXECUTIVE_FILE_RUNNING           0x0000002a // No executive file running in the device
#define MW_ERROR_DEVICE_USER_BUFFER_FULL             0x0000002b // Insufficient Device user buffer, failed to send data.
#define MW_ERROR_DEVICE_USER_BUFFER_EMPTY            0x0000002c // When device user buffer is empty, it can receive data.
#define MW_ERROR_DEVICE_MSG_NOT_REPLIED              0x0000002d // Device require a message response (no replied yet)
#define MW_ERROR_DEVICE_DUMMY_MSG                    0x0000002e // Device has dummy message replied (no reply required)
#define MW_ERROR_DEVICE_MEMORY_ADDR                  0x0000002f // Invalid device memory address
#define MW_ERROR_DEVICE_MEMORY_LENGTH                0x00000030 // Invalid the length of device memory
#define MW_ERROR_CONTROL_CODE                        0x00000031 // Invalid control code
#define MW_ERROR_UNKNOW_COMMAND                      0x00000032 // Invalid/unknown command to device
#define MW_ERROR_INVALID_COMMAND_PARAMETER           0x00000033 // Invalid command parameter
#define MW_ERROR_COMMAND_DATA_LENGTH                 0x00000034 // Invalid command data length
#define MW_ERROR_DEVICE_BUFFER_FULL                  0x00000035 // device buffer full
#define MW_ERROR_EXECUTIVE_FILE_RUNNING              0x00000036 // Part of operation doesn't support when executive file running
in the device
#define MW_ERROR_NO_DEVICE_MESSAGE                   0x00000037 // No device message available
#define MW_ERROR_LOGIN_COUNT                         0x00000038 // Error in Device login count
#define MW_ERROR_KEYEXCHANGE_FAILED                  0x00000039 // failed exchange communication key
#define MW_ERROR_BAD_COMMUNICATION_KEY               0x0000003a // Communication key error
#define MW_ERROR_BAD_DEVICE_TIME                     0x0000003b // Invalid device time
#define MW_ERROR_BAD_DEVICE_INFOMATION               0x0000003c // Device information error
#define MW_ERROR_BAD_COMMAND_SEQUENCE                0x0000003d // Command sequence error
#define MW_ERROR_COMMUNICATION_DATA_LENGTH           0x0000003e // The length of communication data error
#define MW_ERROR_ECC                                 0x0000003f // Device ECC encryption error
#define MW_ERROR_BAD_UPDATE_DATA_LENGTH              0x00000040 // The length of update Data error
#define MW_ERROR_UPDATE_STATE                        0x00000042 // Update status error
#define MW_ERROR_UPDATE_KEY_NOT_ENABLE               0x00000043 // It is required to set remote update key before remote update
#define MW_ERROR_LOCKED_FOREVER                      0x00000044 // Device locked permanently.
#define MW_ERROR_LOCKED_TEMPORARY                    0x00000045 // Device locked temporary
#define MW_ERROR_DEVICE_UNLOCKED                     0x00000046 // Device unlocked
#define MW_ERROR_DEVICE_FLASH                        0x00000047 // Device flash error, may replace with new device.
```

```
#define MW_ERROR_DEVICE_ERROR                          0x00000048 // Device error
#define MW_ERROR_TOO_MANY_DEVICE                       0x00000049 // Too many device, it can not exceed 128
#define MW_ERROR_DEVICE_NOT_ENOUGH_USER_MEMORY         0x0000004a // no sufficient device memory reserve to user
#define MW_ERROR_FAKE_DEVICE                           0x0000004b // Fake device
#define MW_ERROR_DEVICE_BLK_OUT_RANGE                  0x0000004c // Device out of range for batch read/write
#define MW_ERROR_DEVICE_FS_ERR_BAK_ERROR               0x0000004d // Device backup error
#define MW_ERROR_DEVICE_TMR_ERR_ADJUST_TIME_TIMEOUT    0x0000004e // Timeout when device adjust time
#define MW_ERROR_DEVICE_EXCH_ERROR                     0x0000004f // Device exchange memory error
#define MW_ERROR_DEVICE_AES_ERR                        0x00000050 // Device AES algorithm error
#define MW_ERROR_DEVICE_HASH_ERR_UNSUPPORTED_ALGO      0x00000051 // Hash algorithm doesn't support
#define MW_ERROR_DEVICE_BAD_PRIVATE_KEY                0x00000052 // Private key error
#define MW_ERROR_DEVICE_BAD_PUBLIC_KEY                 0x00000053 // Public key error
#define MW_ERROR_DEVICE_BAD_SYMMETRIC_KEY              0x00000054 // Symmetric key error
#define MW_ERROR_DEVICE_BAD_SIGNATURE                  0x00000055 // signature error
#define MW_ERROR_DEVICE_KEY_ERR_BAD_ALGO               0x00000056 // algorithm error
#define MW_ERROR_DEVICE_LOGO_ERR_LOGO                  0x00000057 // Logic arror
#define MW_ERROR_EXEC_PARAM_TOO_LONG                   0x00000058 // The length of executive parameter data error
#define MW_ERROR_OPEN_ERROR                            0x00000059 // Failed to open device
#define MW_ERROR_LOAD_SYS_MODULE_ERROR                 0x0000005A // Failed to load system module
#define MW_ERROR_SYS_MODULE_FUNCTION_ERROR             0x0000005B // The function address of system module error
#define MW_ERROR_RSA                                   0x0000005C // Device RSA encryption error
#define MW_ERROR_KEY                                   0x0000005D // The public key for encryption error
#define MW_ERROR_DEVICE_EXEC_ERR_UNALIGNED_MEM_ADDR    0x0000005E // The memory address doesn't aligned
#define MW_ERROR_DEVICE_EXEC_ERR_STACK_OVERFLOW        0x0000005F // User stack overflow
#define MW_ERROR_DEVELOPER_ID_MISMATCH                 0x00000060 // Developer ID doesn't consistent.
#define MW_ERROR_LM_GENERAL_ERROR                      0x00000061 // Invalid data format returned from LM
#define MW_ERROR_LM_UNSUPPORTED_CERT_TYPE              0x00000062 // Invalid certificate type
#define MW_ERROR_LM_UNSUPPORTED_UPDATE_OBJECT_TYPE     0x00000063 // Invalid update object type
#define MW_ERROR_LM_UPDATE_PKG_FORMAT_WRONG            0x00000064 // Invalid update package format
#define MW_ERROR_CERT                                  0x00000065 // Invalid certificate
#define MW_ERROR_DEX_NOT_LOADED                        0x00000066 // Dynamic execution not load to the device
#define MW_ERROR_SYSAPP_NOT_FOUND                      0x00000067 // System application not found
#define MW_ERROR_UNSUPPORTED_OBJECT_TYPE               0x00000068 // Invalid object type
#define MW_ERROR_UPDATE_BLOCK_LENGTH                   0x00000069 // The length of update block error
#define MW_ERROR_UPDATE_SEED_NOT_FOUND                 0x0000006A // update seed not found
#define MW_ERROR_HEAP                                  0x0000006B // Device system heap collapse
```

```
#define MW_ERROR_DEX_OUT_OF_RANGE                0x0000006C // Out of range to dynamic execution
#define MW_ERROR_DEX_TOO_LARGE                   0x0000006D // Too big size to dynamic execution
#define MW_ERROR_UPDATE_INFO                     0x0000006E // Update information error
#define MW_ERROR_DEVICE_STACK_OVERFLOW           0x0000006F // Device stack overflow
#define MW_ERROR_COMMUNICATION_CRYPT             0x00000070 // Communication cryptography error
#define MW_ERROR_BAD_UPDATE_PKG                  0x00000071 // Update package error
#define MW_ERROR_UPDATE_PKG_VERSION_LOW          0x00000072 // The version of update package is too low
#define MW_ERROR_UPDATE_OBJECT_TYPE              0x00000073 // The update object type error
#define MW_ERROR_UPDATE_DEVELOPER_ID             0x00000074 // the update developer ID incorrect
#define MW_ERROR_UPDATE_FILE_TYPE                0x00000075 // Invalid update file type
#define MW_ERROR_NO_ADJUST_TIME_REQUEST          0x00000076 // No time adjustment request available
#define MW_ERROR_CERT_REVOKED                    0x00000077 // Certificate has been revoked
#define MW_ERROR_WRONG_CERT_SLOT                 0x00000078 // Certificate slot number error
#define MW_ERROR_DEVICE_HASH_ERR_NOT_MATCH       0x00000079 // Inconsistent of Hash calculation
#define MW_ERROR_BAD_RND                         0x0000007A // Random value error
#define MW_ERROR_RTC                             0x0000007B // Real time clock error, please replace with new device
#define MW_ERROR_RTC_POWER                       0x0000007C // Power off of real time clock, please replace with new device
#define MW_ERROR_NO_PID_DEVICE                   0x00002001 // The Device with specified PID not found
#define MW_ERROR_RET_DATA_FORMAT                 0x00002002 // Invalid data format returned
#define MW_ERROR_FILE_NAME_LEN                   0x00002003 // Incorrect length of file name
#define MW_ERROR_BLOCK_DATA                      0x00002004 // Data block error
#define MW_ERROR_FILE_PROPERTY_PARAM             0x00002005 // File property parameter error
#define MW_ERROR_FILE_COUNT_MAX                    0x00002006 // The actual file number out of the maximum file number to
device
#define MW_ERROR_MALLOC_MEMORY                   0x00002007 // Failed to memory allocation
#define MW_ERROR_PIN_LENGTH                      0x00002008 // Invalid PIN length
#define MW_ERROR_UPDATA_PKG_PID                  0x00002009 // The Update Package PID doesn't consistent with the device PID

#define MW_ERROR_ENTRY_TYPE                      0x0000A001 // entry type error
#define MW_ERROR_ENTRY_FLAG                      0x0000A002 // entry flag error
#define MW_ERROR_OPCODE                          0x0000A003 // Error operate code
#define MW_ERROR_PARAM                           0x0000A004 // The parameter error
#define MW_ERROR_DATA_LENGTH                     0x0000A005 // The data length error
#define MW_ERROR_PIN                             0x0000A006 // PIN error
#define MW_ERROR_DEVICE_BLOCK                    0x0000A007 // Device PIN blocked
#define MW_ERROR_PRIVILEGE                       0x0000A008 // No privilege
```

```
#define MW_ERROR_DATA_INFO                  0x0000A009 // Data information error
#define MW_ERROR_HASH_TYPE                  0x0000A00A // Error in hash type
#define MW_ERROR_ALG_MODE                   0x0000A00B // Error in algorithm mode
#define MW_ERROR_INIT_EVT                   0x0000A00C // Failed in initialization
#define MW_ERROR_ALG_TYPE                   0x0000A00D // Error algorithm type
#define MW_ERROR_BIT_SIZE                   0x0000A00E // Error of key bit size
#define MW_ERROR_FILE_TYPE                  0x0000A00F // error file type
#define MW_ERROR_NO_INIT_PID                0x0000A010 // PID not initialized
#define MW_ERROR_NO_INIT_PIN                0x0000A011 // PIN not modified
#define MW_ERROR_NO_VALIDATE_PKG            0x0000A012 // invalid update file package
#define MW_ERROR_AUTH_STAGE                 0x0000A013 // Failed in authentication
#define MW_ERROR_FILE_PRIVILEGE             0x0000A014 // File permission error
#define MW_ERROR_FILE_RANGE                 0x0000A015 // Error file range
#define MW_ERROR_KEY_FILE                   0x0000A016 // Key file error
#define MW_ERROR_REQ_IDENTIFY               0x0000A017 // Error to return identification
#define MW_ERROR_INVALID_SEED               0x0000A018 // Invalid seed
#define MW_ERROR_SEED_CHECK                 0x0000A019 // Seed verification error
#define MW_ERROR_UPDATE_HASH                0x0000A01A // Hash update error
#define MW_ERROR_UPDATE_PID                 0x0000A01B // Update PID error
#define MW_ERROR_UPDATE_INDEX               0x0000A01C // Update index error
#define MW_ERROR_UPDATE_TIME                0x0000A01D // Update time error
#define MW_ERROR_UPDATE_SERIAL_NUM          0x0000A01E // The update serial number error
#define MW_ERROR_UPDATE_SN                  0x0000A01F // Update file version error
#define MW_ERROR_UPDATE_ORDER               0x0000A020 // The sequence to update file error
#define MW_ERROR_UPDATE_OPERATE             0x0000A021 // Update Operation error
#define MW_ERROR_KEY_TYPE                   0x0000A022 // Key type error
#define MW_ERROR_OTHER                      0x0000AFFF // Other error
```