META Development Kit User Guide

Programming Guide

Customer Support

6001

Doc No:	CS6001-H4C-PGD-V1.0EN
Version:	V1.0
Release date:	2017-07-29
Classification:	Confidential B

META Development Kit User Guide

Programming Guide

MediaTek Inc.

Postal address

No. 1, Dusing 1st Rd. , Hsinchu Science Park, Hsinchu City, Taiwan 30078

MTK support office address

No. 1, Dusing 1st Rd. , Hsinchu Science Park, Hsinchu City, Taiwan 30078

Internet

http://www.mediatek.com/

Document Revision History

Revision	Date	Description
V1.0	201-07-10	Initial release

Table of Contents

Lists of Tables

Lists of Figures

1 Introduction

1.1 Purpose

META (Mobile Engineering Testing Architecture) is designed to provide the functionality of RF testing, NVRAM access testing, speech related testing of advanced feature – melody and iMelody. Regarding the architecture of META, it is composed of META-TARGET and META-LAB or META-Factory. META-TARGET is MediaTek Inc. hardware platform with conventional full image but operated in test mode, which only TST task, FT task, NVRAM task and L1SP task are spawn. In MediaTek Inc. software package, META windows application tool provides 2 main applications, META-LAB and META-Factory. META-LAB offers versatile testing features in RF TX/RX/AFC control, NVRAM access testing and editing, melody and iMelody play testing, but all testing procedure should be operated manually due to no specific equipment control. Therefore, META-LAB is designed dedicatedly for R&D application. Contrarily, META-Factory only provide the RF calibration function required in factory mass production line, of course, it supports test equipments (e.g. R&S CMU/CMW500, Arnitsu MT8820/MT8870, KeySight EXT/EXM…etc) with automation calibration. In fact, phone developer may prefer to develop their factory tool optimized for their production line and phone design. So, META-FACTORY could be the reference for customer’s factory tool design.

1.2 Scope

The document provide the programming details of the META.

1.3 Who should read this document

This document is primarily intended for:

- Users who want to understand what META is.
- Users who need to do RF testing, NVRAM access testing, speech related testing of advanced feature or develop their factory tool optimized for their production line and phone design.

2 References

N/A

3 Definitions

For the purposes of the present document, the following terms and definitions apply:

META (Mobile Engineering Testing Architecture): META is designed to provide the functionality of RF testing, NVRAM access testing, speech related testing of advanced feature – melody and iMelody. Regarding the architecture of META, it is composed of META-TARGET and META-LAB or META-Factory.

META-TARGET: it is MediaTek Inc. hardware platform with conventional full image but operated in test mode, which only TST task, FT task, NVRAM task and L1SP task are spawn.

4 Abbreviations

Please note the abbreviations and their explanations provided in Table 4-1. They are used in many fundamental definitions and explanations in this document and are specific to the information that this document contains.

Table 4-1. Abbreviations

Abbreviations	Explanation
META	Mobile engineering testing architecture.

5 Overview

In META application design, META DLL plays the heart in the operation of testing application that written by programmer to test mobile station based on MediaTek solution, and it takes the responsibility that communicates with META-TARGET, and testing command handling. So, META-DLL can help testing application to leave META-TARGET alone, no matter the implementation change or chip revision in META-TARGET, except for the addition of new testing function.

META-DLL contains the mechanism that adapts the heterogeneity between testing application and META-TARGET. Testing application uses API exported by META-DLL to test mobile phone. Regarding exported API, RF testing/calibration, NVRAM access, and L1SP testing are supported. In order to provide these API for versatile testing and make testing application independent of META-TARGET, exported API provided by META-DLL handle the packing of testing command, and each API handles specific command packing and unpacking. For detail API definition, please refer to chapter 6.

META-DLL also provides testing application with a facility for editing the content of NVRAM, which is a proprietary file system in target. Testing application must provide the information that is necessary for META-DLL to decode and encode the content of NVRAM. This information is actually a file that should be generated in the build process of target image. For detail about how to provide META-DLL with this file, please refer to chapter 6.4.

5.1 META-DLL Architecture

META-DLL is a dynamic link library. User application written for testing mobile phone based on MediaTek solution uses this META-DLL as a mean to test the target. This chapter introduces the common aspect of META_DLL and describes the architecture of MEAT-DLL callback mechanism. The restrictions of using META-DLL callback mechanism are also mentioned in this chapter.

5.1.1 META-DLL Software Architecture and Callback Mechanism

The communication paradigm between META-DLL and META-TARGET uses 3 different commands, Request, Confirm, and Indication commands. META-DLL uses Request command to instruct META-TARGET, and META-TARGET uses Confirm command to inform META-DLL with the completion of the instructed testing. Except Confirm command, META-TARGET also uses Indication command to inform META-DLL with some unpredicted events.

Due to non-predicted processing time consumed in META-TARGET, META-DLL uses callback functions to avoid the user application being pended until response gotten from META-TARGET. There are 2 different types of callback functions that should be registered to META-DLL. The first type of callback function is a global and unique error-handler function. This function is registered to META-DLL in the META-DLL initialization function. The second type of callback function is registered to META-DLL whenever user application uses API exported by META-DLL. These API actually send Request command to META-TARGET. When META-TARGET finishes the indicated testing, META-TARGET will send back Confirm and/or Indication command to META-DLL. META-DLL uses the registered corresponding callback function to inform user application with the arrival of these commands.

Figure 5-1 shows the architecture that META-DLL uses for implementing this callback mechanism. In this architecture, META-DLL always monitors the RS-232 port that initialized in META_Init function to receive any incoming command. If the received command is corrupted, the global and unique error-handler callback function installed when user application calls META_Init function is called. If the received command is correct, and there is a corresponding callback function installed for this command, this corresponding callback function is called; otherwise, the global and unique error-handler is called.

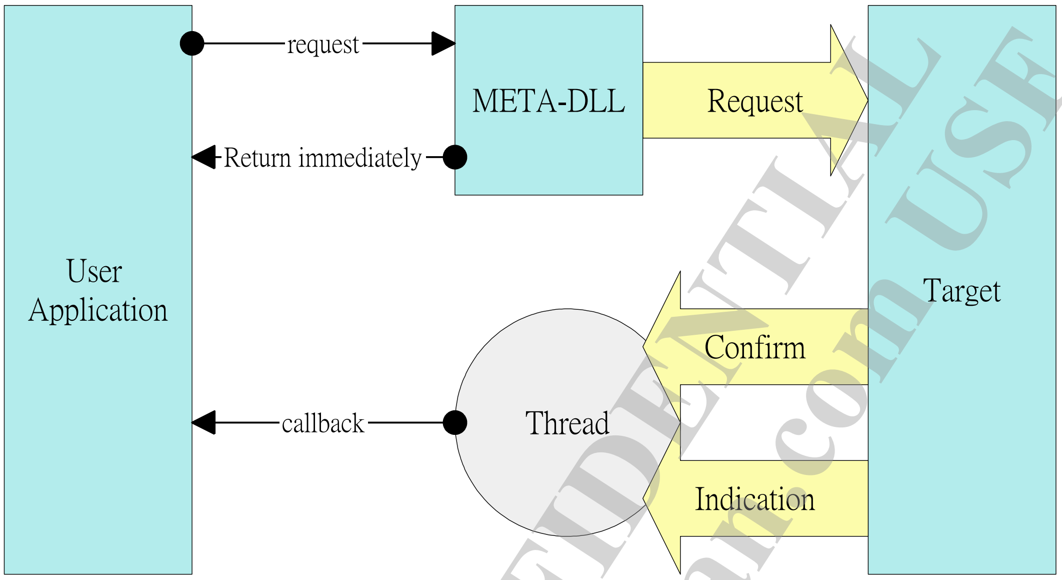

Figure 5-1 META-DLL callback mechanism

It's obvious that the callback function is called and executes in the context of Thread, whose responsibility is to monitor and receive all the incoming commands sent by META-TARGET. Therefore, the following notes are important.

- DO NOT block the thread in the callback for long. The thread has to receive command sent from META-TARGET. If the thread is blocked, it will not be able to monitor and receive any incoming commands, and therefore the other callback functions will be also blocked.
- Since the callback is called and executes in the context of the thread, race condition may occur in the callback function, if this callback function uses some resources that are also used by other threads. Please refer to related material for programming multi-threaded application.
- If user application is written by using Borland C++ Builder, DO NOT access GUI functionality in the callback function. This will introduce some unexpected errors. For the sake, please refer to Borland C++ Builder documentation.
- It's the best only sending messages or setting events in the callback function. Do the other job in the message handler. This way will avoid the race condition of multi-threading.

5.1.2 Internal Token Counter for Callback Mechanism

META-DLL maintains an internal counter for callback mechanism. Every time you issue a command to target, META-DLL will return a unique token value to indicate of this command as a timestamp. There are two purposes: 1. You can use token value to cancel any callback before target sending confirmation. 2. You can tell from two confirmations with the same type, for example, when you're doing the Gain Sweep test; you can continuously issue META_Rf_PM command without waiting for receiving previous confirmation. META-DLL will choose corresponding callback function for you according to token value and command type of confirmation.

5.1.3 META_RESULT

META_RESULT is an enumeration type, which is defined as following. If an exported function of META-DLL has returned value, the type of this returned value is always META_RESULT.

```
typedef enum
{
   // META_DLL received a corrupted frame
   META_CNF_FRAME_ERROR = 0,

   // META_DLL received a confirm or indication from target,
   // but there is not corresponding call back function
   // installed for this confirm or indication.
   META_CNF_NO_CALLBACK = 1,

   // META_DLL received a corrupted primitive.
   META_CNF_PRIMITIVE_ERROR = 2,

   // META_DLL received a confirm or indication from
   // target, but there is no sufficient memory to process.
   META_CNF_NO_MEMORY = 3,

   // META_DLL retrieved a callback function, however,
   // the user input arguments are invalid.
   META_CNF_CALLBACK_PARAMETER_ERROR = 4,

   // META_DLL received a confirm with peer msg, however,
    // the peer msg is corrupted.
    META_CNF_PEER_MSG_ERROR = 5,

    // META_DLL received a confirm and successfully executed
    // the callback function.
    META_CNF_OK = 6
} META_CNF_ERR_CODE;

// The magic value to stop usb enumerate process
#define ENUM_USB_STOP   9876
#define ENUM_ANY_STOP   9876

typedef enum
{
    META_SUCCESS                    // 0
    ,META_FAILED                    // 1
    ,META_COMM_FAIL                 // 2
    ,META_NORESPONSE                // 3
    ,META_EBOOT_FAILED              // 4
    ,META_BUFFER_LEN                // 5
    ,META_FILE_BAD                  // 6
    ,META_LID_INVALID               // 7
    ,META_INTERNAL_DB_ERR           // 8
    ,META_NO_MEMORY                 // 9
    ,META_INVALID_ARGUMENTS         // 10
    ,META_TIMEOUT                   // 11
    ,META_BUSY                      // 12
    ,META_INVALID_HANDLE            // 13
    ,META_FAT_ERROR                 // 14
```

```
,META_FAT_DISK_FULL                    // 15
,META_FAT_ROOT_DIR_FULL                // 16
,META_FAT_INVALID_FILENAME             // 17
,META_FAT_INVALID_FILE_HANDLE          // 18
,META_FAT_FILE_NOT_FOUND               // 19
,META_FAT_DRIVE_NOT_FOUND              // 20
,META_FAT_PATH_NOT_FOUND               // 21
,META_FAT_ACCESS_DENIED                // 22
,META_FAT_TOO_MANY_FILES               // 23
,META_INCORRECT_TARGET_VER             // 24
,META_COM_ERROR                        // 25
,META_BROM_CMD_ERROR                   // 26
,META_INCORRECT_BBCHIP_TYPE            // 27
,META_BROM_ERROR                       // 28
,META_STOP_BOOTUP_PROCEDURE            // 29
,META_CANCEL                           // 30
,META_FUNC_NOT_IMPLEMENT_YET           // 31
,META_FAT_APP_QUOTA_FULL               // 32
,META_IMEI_CD_ERROR                    // 33
,META_RFID_MISMATCH                    // 34
,META_NVRAM_DB_IS_NOT_LOADED_YET       // 35
,META_WAIT_FOR_TARGET_READY_TIMEOUT    // 36
,META_ERR_EXCEED_MAX_PEER_BUF_SIZE     // 37
,META_BROM_SECURITY_CHECK_FAIL         // 38
,META_MAUI_DB_INCONSISTENT             // 39
,META_FAT_FILEPATH_TOO_LONG            // 40
,META_FAT_RESTRICTED_FILEPATH          // 41
,META_FAT_DIR_NOT_EXIST                // 42
```

```
,META_FAT_DISK_SPACE_IS_NOT_ENOUGH                    // 43
,META_TDMB_ERR_BAND_NOT_EXIST                         // 44
,META_TDMB_ERR_FREQ_NOT_EXIST                         // 45
,META_TDMB_ERR_ENSM_NOT_EXIST                         // 46
,META_TDMB_ERR_SERV_NOT_EXIST                         // 47
,META_TDMB_ERR_SUB_CHAN_NOT_EXIST                     // 48
,META_TDMB_ERR_DEMOD_STATE                            // 49
,META_ENUMERATE_USB_FAIL                              // 50
,META_STOP_ENUM_USB_PROCEDURE                         // 51
,META_MISC_TARGET_LOAD_NEED_TO_BE_PATCHED             // 52
,META_MISC_INI_FILE_SETTINGS_WRONG                    // 53
,META_MISC_FAIL_TO_READ_IMEI                          // 54
,META_MISC_FAIL_TO_BACKUP_FILE                        // 55
,META_MISC_FAIL_TO_WRITE_BACKUP_RESULT                // 56
,META_MISC_FAIL_TO_GET_NVRAM_FOLDER_PATH              // 57
,META_MISC_FAIL_TO_GET_NVRAM_MUST_LIST                // 58
,META_STOP_CURRENT_PROCEDURE                          // 59
,META_MISC_CUSTOMIZED_NVRAM_ERROR                     // 60
,META_MISC_FOLDER_EMPTY_CHECKING_FAIL                 // 61
,META_MISC_TOO_MANY_BACKUP_RESULT_FILE                // 62
,META_MISC_TOO_MANY_RESTORE_RESULT_FILE               // 63
,META_MISC_RESTORE_RESULT_FILE_NOT_EXIST              // 64
,META_MISC_RESTORE_RESULT_FILE_INCOMPLETE             // 65
,META_FAIL_TO_CELAR_ALL_IN_BACUP_FOLDER               // 66
,META_MISC_BACKUP_RESULT_FILE_NOT_EXIST               // 67
,META_MISC_BACKUP_RESULT_FILE_INCOMPLETE              // 68
,META_MISC_IMEI_MISMATCH                              // 69
,META_MISC_SML_FILE_VERIFY_FAIL                       // 70
,META_MISC_BACKUP_RESULT_NOT_ENOUGH_FOR_NEW_LOAD      // 71
```

```
    ,META_MISC_FAIL_TO_RESTORE_FILE                   // 72
    ,META_MISC_FAIL_TO_WRITE_RESTORE_RESULT           // 73
    ,META_MISC_USE_WRONG_API_FOR_NEW_LOAD             // 74
    ,META_MISC_QUERY_TARGET_CAPABILITY_FAIL           // 75
    ,META_MISC_INI_SETTINGS_ERR_IN_NVRAM_SEC          // 76
    ,META_MISC_INI_SETTINGS_ERR_IN_TARGET_SEC         // 77
    ,META_MISC_INI_SETTINGS_ERR_IN_PC_SEC             // 78
    ,META_MISC_NO_FILES_NEED_TO_BE_UPLOAD             // 79
    ,META_FAT_ACTION_NOT_SUPPORT                      // 80
    ,META_MISC_EMPTY_UPLOADFILES_AND_IMEI_SEC         // 81
    ,META_MISC_INI_SETTINGS_ERR_IN_MORE_SEC           // 82
    ,META_MISC_INI_SETTINGS_ERR_IN_DELETE_SEC         // 83
    ,META_MISC_CHECK_TARGET_NVRAM_FILES_FAIL          // 84
    ,META_MISC_FAIL_TO_GET_NVRAM_FOLDER_AMOUNT        // 85
    ,META_AUDIO_CHECK_WAVE_FILE_FAIL                  // 86
    ,META_MISC_COLLECT_NVRAM_FOLDER_FILES_FAILED      // 87
    ,META_MISC_COLLECT_NVRAM_FOLDER_FILES_FIRST       // 88
    ,META_MISC_BACKUP_FILE_NOT_FOUND_IN_NVRAM         // 89
    ,META_MISC_BACKUP_MORE_FILE_NOT_FOUND_IN_NVRAM    // 90
    ,META_MISC_LOCAL_FS_UNKNOWN_ERROR                 // 91
    ,META_MISC_RETORE_FILE_NOT_FOUND_IN_BACKUP_RESULT // 92
    ,META_MISC_LEGACY_ADC_FILE_NOT_FOUND              // 93
    ,META_MISC_LEGACY_BARCODE_FILE_NOT_FOUND          // 94
    ,META_MISC_FILE_SIZE_MISMATCH                     // 95
    ,META_MISC_RESTORE_TARGET_NOT_FOUND_IN_NVRAM      // 96
  ,META_LAST_RESULT

} META_RESULT;
```

5.1.4 Error Handler

To use META_DLL, user application is recommended to install an error handler function when user initializes META_DLL by calling META_Init function, which will be mentioned in the following chapter. When META_DLL receives a bad frame or command from META-TARGET, this error handler function is called. See the following prototype of this error handler.

5.1.5 META_Error_CallBack

Definition:

```
typedef void (__stdcall *META_Error_CallBack)(META_CNF_ERR_CODE   mr);
```

Description:

Definition of error handler.

5.1.6 META_CNF_ERR_CODE

```
typedef enum
{
    // META_DLL received a corrupted frame
    META_CNF_FRAME_ERROR = 0,

    // META_DLL received a confirm or indication from target,
    // but there is not corresponding call back function
    // installed for this confirm or indication.
    META_CNF_NO_CALLBACK = 1,

    // META_DLL received a corrupted primitive.
    META_CNF_PRIMITIVE_ERROR = 2,

    // META_DLL received a confirm or indication from
    // target, but there is no sufficient memory to process.
    META_CNF_NO_MEMORY = 3,
        // META_DLL retrieved a callback function, however,
        // the user input arguments are invalid.
        META_CNF_CALLBACK_PARAMETER_ERROR = 4,

        // META_DLL received a confirm with peer msg, however,
        // the peer msg is corrupted.
        META_CNF_PEER_MSG_ERROR = 5,

        // META_DLL received a confirm and successfully executed
        // the callback function.
        META_CNF_OK = 6
} META_CNF_ERR_CODE;
```

5.2 Programming Convention

The META DLL is not thread safe, the user cannot use the API with the same META handle in different thread context. However, different META handle in different thread context is allowed.

The META handle represents a session handle ID to a connection channel to a target.

Example 1 (one connection is not able to handle multi-thread request)

Table 5-1 Programming convention example 1

Thread context 1	Thread context 2	Result
META_GetTargetVerInfoEx_r(0, &cnf);	META_GetTargetVerInfoEx_r(0, &cnf);	the result could no be guaranteed

Example 2 (multiple connections can run concurrently)

Table 5-2 Programming convention example 2

Thread context 1	Thread context 2	Result
META_GetTargetVerInfoEx_r(0, &cnf);	META_GetTargetVerInfoEx_r(1, &cnf);	OK

6 Exported Functions

This chapter mentions the functions exported by META-DLL, and their prototypes.

6.1 The Terminology of Function Descriptions

6.1.1 The Meaning of Parameter Table:

Parameter:

Table 6-1 The meaning of parameter table

Parameter	IN/OUT	Description

Parameter:

The name of parameter.

IN/OUT:

IN: It means this parameter is used for input value.

OUT: It means this parameter is used for output value. You have to pass the address pointer of container.

Description:

The description of that parameter.

6.2 Reentrant Functions

The function with _r postfix means it is reentrant API. If you want to use multi-thread to connect with different targets concurrently via META_DLL, you MUST create different META_DLL handle for each thread and use reentrant API call instead.

6.3 Exported General Functions

6.3.1 META_GetVersion

Definition:

void __stdcall META_ GetVersion (unsigned int *ver)

Description:

Get version number of this META_DLL.

Parameter:

Table 6-2 The parameter of META_GetVersion

Parameter	IN/OUT	Description

Ver	OUT	Pointer to an unsigned integer, which will contain the version number of META_DLL. The version number is x.y.z;
		x = ((*ver) & 0xFF000000) >> 24
		y = ((*ver) & 0x00FF0000) >> 16
		z = ((*ver) & 0x0000FFFF)

6.3.2 META_Cancel

Definition:

void __stdcall META_ Cancel (short token)

Description:

Uninstall an installed callback function.

Parameter:

Table 6-3 The parameter of META_Cancel

Parameter	IN/OUT	Description
token	IN	Some exported functions of META_DLL need user to provide one or two callback function. The function is called by META_DLL when META_DLL receives a confirmation or indication from target. User can uninstall a callback by calling this function.

6.3.3 META_GetTargetVerInfo

Definition:

META_RESULT __stdcall META_GetTargetVerInfo(const META_GET_VERSION_INFO_CNF cb, short *token, void *usrData);

META_RESULT __stdcall META_GetTargetVerInfo_r(const int meta_handle, const META_GET_VERSION_INFO_CNF cb, short *token, void *usrData);

META_RESULT __stdcall META_GetTargetVerInfoEx(VerInfo_Cnf *cnf);

META_RESULT __stdcall META_GetTargetVerInfoEx_r(const int meta_handle, VerInfo_Cnf *cnf);

```
typedef struct {
        char    BB_CHIP[64];            // BaseBand chip version
        char    ECO_VER[4];             // ECO version
        char      DSP_FW[64];             // DSP firmware version
        char      DSP_PATCH[64];  // DSP patch version
        char      SW_VER[64];             // S/W version
        char      HW_VER[64];             // H/W board version
        char      MELODY_VER[64];         // Melody version
} VerInfo_Cnf;
```

Description:

This function will retrieve S/W and H/W version information from target.

Callback:

typedef void (__stdcall *META_GET_VERSION_INFO_CNF)(const VerInfo_Cnf *cnf, const short token, void *usrData);

Return Value:

Table 6-4 The return value of META_GetTargetVerInfo

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.

Parameter:

Table 6-5 The parameter of META_GetTargetVerInfo

Parameter	IN/OUT	Description
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
UsrData	IN	Parameter used by user.

6.3.4 META_GetErrorString

Definition:

const char * __stdcall META_ GetErrorString(META_RESULT ErrCode)

Description:

Translate error code the error message string.

Return Value:

Table 6-6 The return value of META_GetErrorString

Return value	Description
const char *	Return a char pointer to the const error message string inside META_DLL. DO NOT free this point, because you don't have to.

Parameter:

Table 6-7 The parameter of META_GetErrorString

Parameter	IN/OUT	Description
ErrCode	IN	META result code.

6.3.5 META_BaudrateEnumToName

Definition:

const char * __stdcall META_BaudrateEnumToName(META_COMM_BAUDRATE baudrate)

Description:

Translate baud rate enum code to string.

Return Value:

Table 6-8 The return value of META_BaudrateEnumToName

Return value	Description
const char *	Return a char pointer to the const baud rate string inside META_DLL. DO NOT free this point, because you don't have to.

Parameter:

Table 6-9 The parameter of META_BaudrateEnumToName

Parameter	IN/OUT	Description
baudrate	IN	META_COMM_BAUDRATE

6.3.6 META_CancelAllBlockingCall

Definition:

void __stdcall META_CancelAllBlockingCall(void)

Description:

This function will release all the previous blocking function call.

Return Value:

Table 6-10 The return value of META_CancelAllBlockingCall

Return value	Description
N/A	

Parameter:

Table 6-11 The parameter of META_CancelAllBlockingCall

Parameter	IN/OUT	Description
N/A	N/A	

6.3.7 META_QueryIfFunctionSupportedByTarget

Definition:

```
META_RESULT __stdcall META_QueryIfFunctionSupportedByTarget(
                        unsigned int ms_timeout,
                        const char *query_func_name)
```

Description:

This function will query if the inquired function was supported by target.

Return Value:

Table 6-12 The return value of META_QueryIfFunctionSupportedByTarget

Return value	Description
META_SUCCESS	The inquired function is supported by target.
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-13 The parameter of META_QueryIfFunctionSupportedByTarget

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
query_func_name	IN	The inquired function name string. eg: Input "META_Rf_SetAfcSinWaveDetection" to query if META_Rf_SetAfcSinWaveDetection function was supported.

6.3.8 META_EnableWatchDogTimer

Definition:

```
META_RESULT  __stdcall META_EnableWatchDogTimer (
                                unsigned int ms_timeout,
                                FtWatchDog *req)
```

Description:

This function will reset baseband after specific time, the FtWatchDog uses baseband clock unit.

Return Value:

Table 6-14 The return value of META_EnableWatchDogTimer

Return value	Description
None	If the EnableWatchDogTimer function works successfully, baseband reset, it returns nothing.
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-15 The parameter of META_EnableWatchDogTimer

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

req	IN	The struct of FtWatchDog is
		typedef struct {
		unsigned int ms_timeout_interval;
		} FtWatchDog;
		The ms_timeout_interval is baseband chip watchdog timeout count down value.
		Take the following for example:
		FtWatchDog req;
		req. ms_timeout_interval =5000; // reset after 5 secs
		META_EnableWatchDogTimer(1500,&req);

6.3.9 META_QueryPMICID

Definition:

META_QueryPMICID (unsigned int ms_timeout, PMIC_ID *cnf)

Description:

The users could quary the power measurement IC in cellphone by calling this function.

CallBack:

NA

Return Value:

Table 6-16 The return value of META_QueryPMICID

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-17 The parameter of META_QueryPMICID

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

PMIC_ID	IN/OUT	#define FT_MT_UNKNOWN 0
		#define FT_MT6305 1
		#define FT_MT6318 2
		typedef struct {
		unsigned char id;
		} PMIC_ID;

6.3.10 META_DebugOn_ex

Definition:

META_DebugOn_ex (const int meta_handle)

Description:

Different meta_handle uses different com port to connect with target, this API provide the users different com port log name, such as you meta_handle as 2, using com port6, then the log file will be META_DLL6.log

CallBack:

NA

Return Value:

Table 6-18 The return value of META_DebugOn_ex

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-19 The parameter of META_DebugOn_ex

Parameter	IN/OUT	Description
meta_handle	IN	In multi-thread architecture, there are many meta handles which stand for different threads.

6.3.11 META_DebugOn_With_Handle_FilePath

Definition:

META_RESULT __stdcall META_DebugOn_With_Handle_FilePath (const int meta_handle, const char* filename)

Description:

To open log file and set file path and name for each META Handle

CallBack:

NA

Return Value:

Table 6-20 The return value of META_DebugOn_With_Handle_FilePath

Return value	Description
META_SUCCESS	Success in transmitting the modem log filter to target
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-21 The parameter of META_DebugOn_With_Handle_FilePath

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
filename	IN	The path of modem log filter

6.3.12 META_DebugOff_With_Handle

Definition:

META_RESULT __stdcall META_DebugOff_With_Handle (const int meta_handle)

Description:

To close log file for each META Handle

CallBack:

NA

Return Value:

Table 6-22 The return value of META_DebugOff_With_Handle

Return value	Description
META_SUCCESS	Success in transmitting the modem log filter to target
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-23 The parameter of META_DebugOff_With_Handle

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
filename	IN	The path of modem log filter

6.3.13 META_DebugClear_With_Handle

Definition:

META_RESULT __stdcall META_DebugClear_With_Handle (const int meta_handle)

Description:

To Clear log file for each META Handle

CallBack:

NA

Return Value:

Table 6-24 The return value of META_DebugClear_With_Handle

Return value	Description
META_SUCCESS	Success in transmitting the modem log filter to target
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-25 The parameter of META_DebugClear_With_Handle

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
filename	IN	The path of modem log filter

6.3.14 META_SetLEDLightLevel

Definition:

META_SetLEDLightLevel(unsigned int ms_timeout, FtLEDLevel *req)

Description:

Users could set the LEDLightLevel by calling this API.

CallBack:

NA

Return Value:

Table 6-26 The return value of META_SetLEDLightLevel

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-27 The parameter of META_SetLEDLightLevel

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

FtLEDLevel *req	IN	Specified the LED Light Level
		typedef struct {
		unsigned char led_light_level;
		} FtLEDLevel;
		MAX LEVEL is 5
		MIN LEVEL is 0 which implies dark

6.3.15 META_SetVibratorOnOff

Definition:

META_SetVibratorOnOff(unsigned int ms_timeout, FtVibratorOnOff *req)

Description:

Users could turn ON/OFF the Vibrator by calling this API.

CallBack:

NA

Return Value:

Table 6-28 The return value of META_SetVibratorOnOff

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-29 The parameter of META_SetVibratorOnOff

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

FtVibratorOnOff *req	IN	typedef struct {
		unsigned char onoff;

Parameter	IN/OUT	Description
		} FtVibratorOnOff; 0 is OFF 1 is ON

6.3.16 META_QueryLocalTime

Definition:

META_QueryLocalTime(unsigned int ms_timeout, T_Rtc *cnf)

Description:

Users could get the the RTC time of cell phone by calling this API.

CallBack:

NA

Return Value:

Table 6-30 The return value of META_QueryLocalTime

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-31 The parameter of META_QueryLocalTime

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

T_Rtc *cnf	IN/OUT	typedef struct {
		unsigned char m_rtc_sec;
		/* seconds after the minute - [0,59] */
		unsigned char m_rtc_min;
		/* minutes after the hour - [0,59] */
		unsigned char m_rtc_hour;
		/* hours after the midnight - [0,23] */
		unsigned char m_rtc_day;
		/* day of the month - [1,31] */
		unsigned char m_rtc_mon;
		/* months - [1,12] */
		unsigned char m_rtc_wday;
		/* days in a week - [1,7] */
		unsigned char m_rtc_year;

Parameter	IN/OUT	Description
		/* year of 2XXX, such XXX is between 0 to 255 */ } T_Rtc;

6.3.17 META_QueryITC_PCL

Definition:

META_QueryITC_PCL(unsigned int ms_timeout, RF_GetITC_PCL *cnf)

Description:

Users could get MT6140 RF ITC PCL value

```
typedef struct {
        unsigned int        pcl;
} RF_GetITC_PCL;
```

CallBack:

NA

Return Value:

Table 6-32 The return value of META_QueryITC_PCL

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-33 The parameter of META_QueryITC_PCL

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
RF_GetITC_PCL *cnf	IN/OUT	Get MT6140D PCL value

6.3.18 META_SetMainSubLCDLightLevel

Definition:

META_SetMainSubLCDLightLevel_r(const int meta_handle, unsigned int ms_timeout, FtLCDLevel *req);

Description:

lcd_type is set to be 0, implies MAIN_LCD, SUB_LCD implies 1, but most MAIN_LCD and Sub_LCD have the same LCM module, so the lcd_type is always set to be 0, the value of lcd_light_level is between 0 and 5. while you want to turn off LCD, you set lcd_light_level to be 0.

```
typedef struct {
        unsigned char           lcd_type;
        unsigned char           lcd_light_level;
} FtLCDLevel;
```

CallBack:

NA

Return Value:

Table 6-34 The return value of META_SetMainSubLCDLightLevel

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-35 The parameter of META_SetMainSubLCDLightLevel

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	FtLCDLevel

6.3.19 META_QueryIfTargetSupportDRC

Definition:

META_QueryIfTargetSupportDRC(unsigned int ms_timeout);

Description:

Query if Target Support DRC

CallBack:

NA

Return Value:

Table 6-36 The return value of META_QueryIfTargetSupportDRC

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-37 The parameter of META_QueryIfTargetSupportDRC

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

6.3.20 META_StartTimer

Definition:

META_RESULT __stdcall META_StartTimer();

META_RESULT __stdcall META_StartTimer_r(const int meta_handle);

Description:

Start to record each API's time consumption in target.

CallBack:

NA

Return Value:

Table 6-38 The return value of META_StartTimer

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

6.3.21 META_GetProcessTime

Definition:

META_RESULT __stdcall META_GetProcessTime(unsigned int *pProcessTime, unsigned short *pNumAPIs);

META_RESULT __stdcall META_GetProcessTime_r(const int meta_handle, unsigned int *pProcessTime, unsigned short *pNumAPIs);

Description:

Get the total amount of process time (in milliseconds) of all APIs and the number of APIs after we call META_StartTimer( ).

CallBack:

NA

Return Value:

Table 6-39 The return value of META_GetProcessTime

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-40 The parameter of META_GetProcessTime

Parameter	IN/OUT	Description
*pProcessTime	IN/OUT	The total amount of time consumption (in milliseconds)
*pNumAPIs	IN/OUT	The number of APIs we record.

6.3.22 META_StopTimer

Definition:

META_RESULT __stdcall META_StopTimer();

META_RESULT __stdcall META_StopTimer_r(const int meta_handle);

Description:

Stop to record each API's time consumption in target.

CallBack:

NA

Return Value:

Table 6-41 The return value of META_StopTimer

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

6.3.23 META_MISC_GetIMEILocation

Definition:

META_RESULT __stdcall META_MISC_GetIMEILocation(const unsigned int ms_timeout, META_IMEI_LOC_enum *storagetype);

META_RESULT __stdcall META_MISC_GetIMEILocation_r(const int meta_handle, const unsigned int ms_timeout, META_IMEI_LOC_enum *storagetype);

```
typedef enum
{
   META_STORAGE_TYPE_FAT = 0
  ,META_STORAGE_TYPE_OTP
  ,META_STORAGE_TYPE_SECRO
  ,META_STORAGE_TYPE_END

}META_IMEI_LOC_enum;
```

Description:

Get the target's IMEI storage location. (so far, OTP, SEC_RO, and FAT)

CallBack:

NA

Return Value:

Table 6-42 The return value of META_MISC_GetIMEILocation

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

6.3.24 META_MISC_GetIMEIRecNum

Definition:

META_RESULT __stdcall META_MISC_GetIMEIRecNum(const unsigned int ms_timeout, unsigned short *rec_num);

META_RESULT __stdcall META_MISC_GetIMEIRecNum_r(const int meta_handle, const unsigned int ms_timeout, unsigned short *rec_num);

Description:

Get the target's IMEI Record Number (1 on single SIM card target, 2 on dual SIM card target)

CallBack:

NA

Return Value:

Table 6-43 The return value of META_MISC_GetIMEILocation

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-44 The parameter of META_MISC_GetIMEILocation

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
rec_num	OUT	Total IMEI record number supported on target.

6.3.25 META_MISC_QueryNVRAMFolderAmount

Definition:

META_RESULT __stdcall META_MISC_QueryNVRAMFolderAmount(const unsigned int ms_timeout, unsigned char* folder_amount);

META_RESULT __stdcall META_MISC_QueryNVRAMFolderAmount_r(const int meta_handle, const unsigned int ms_timeout, unsigned char* folder_amount);

Description:

Get the number of target's NVRAM folder.

CallBack:

NA

Return Value:

Table 6-45 The return value of META_MISC_QueryNVRAMFolderAmount

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-46 The parameter of META_MISC_QueryNVRAMFolderAmount

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
folder_amount	OUT	Total number of NVRAM folders.

6.3.26 META_MISC_CheckSIM1Inserted

Definition:

META_RESULT __stdcall META_MISC_CheckSIM1Inserted(const unsigned int ms_timeout,unsigned char* inserted);

META_RESULT __stdcall META_MISC_CheckSIM1Inserted_r(const int meta_handle, const unsigned int ms_timeout, unsigned char* inserted);

Description:

Get the stauts of the SIM card module1 to detect whether SIM card module1 is inserted SIM card or not.

CallBack:

NA

Return Value:

Table 6-47 The return value of META_MISC_CheckSIM1Inserted

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-48 The parameter of META_MISC_CheckSIM1Inserted

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
inserted	OUT	SIM card inserted or not

6.3.27 META_MISC_CheckSIM2Inserted

Definition:

META_RESULT __stdcall META_MISC_CheckSIM2Inserted(const unsigned int ms_timeout,unsigned char* inserted);

META_RESULT __stdcall META_MISC_CheckSIM2Inserted_r(const int meta_handle, const unsigned int ms_timeout, unsigned char* inserted);

Description:

Get the stauts of the SIM card module2 to detect whether SIM card module2 is inserted SIM card or not.

CallBack:

NA

Return Value:

Table 6-49 The return value of META_MISC_CheckSIM2Inserted

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-50 The parameter of META_MISC_CheckSIM2Inserted

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
inserted	OUT	SIM card inserted or not

6.3.28 META_MISC_GetADCFromEFuse

Definition:

META_RESULT __stdcall META_MISC_GetADCFromEFuse(const unsigned int ms_timeout, META_MISC_GET_ADC_FROM_EFUSE_CNF_T *cnf);

META_RESULT __stdcall META_MISC_GetADCFromEFuse_r(const int meta_handle, const unsigned int ms_timeout, META_MISC_GET_ADC_FROM_EFUSE_CNF_T *cnf);

#define META_MISC_SUPPORTED_MAX_ADC_CHN_NUM 20

typedef struct

{

bool bADCStoredInEfuse; // true: ADC is stored in EFUSE, not in NVRAM data.

int i4ADCChnNum; // specify the adc channel number supported by this phone

int i4ADCSlope[META_MISC_SUPPORTED_MAX_ADC_CHN_NUM]; // [0 ~ iADCChnNum-1] is valid when bADCStoredInEfuse = true

int i4ADCOffset[META_MISC_SUPPORTED_MAX_ADC_CHN_NUM];// [0 ~ iADCChnNum-1] is valid when bADCStoredInEfuse = true

}META_MISC_GET_ADC_FROM_EFUSE_CNF_T;

Description:

Get the ADC information from EFUSE.

CallBack:

NA

Return Value:

Table 6-51 The return value of META_MISC_GetADCFromEFuse

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-52 The parameter of META_MISC_GetADCFromEFuse

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
cnf	OUT	ADC information.

6.3.29 META_MISC_SetMuicChargerMode

Definition:

```
META_RESULT __stdcall META_MISC_SetMuicChargerMode(const unsigned int ms_timeout, const unsigned char* req_mode);

META_RESULT __stdcall META_MISC_SetMuicChargerMode_r(const int meta_handle, const unsigned int ms_timeout, const unsigned char* req_mode);
```

```
#define MUIC_MODE_CHARGE_ON    0
#define MUIC_MODE_CHARGE_OFF   1
#define MUIC_MODE_USB_500      2
#define MUIC_MODE_ISET_PROGRAM 3
#define MUIC_MODE_USB_100      4
#define MUIC_MODE_TEST_MODE    5
#define MUIC_MODE_USB_100_2    6
```

Description:

Set the MUIC charger mode.

CallBack:

NA

Return Value:

Table 6-53 The return value of META_MISC_SetMuicChargerMode

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-54 The parameter of META_MISC_SetMuicChargerMode

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
req_mode	OUT	The Requested mode operation to MUIC charger.

6.3.30 META_MISC_CalDataIntegrity_StartRec

Definition:

META_RESULT __stdcall META_MISC_CalDataIntegrity_StartRec(const unsigned int ms_timeout);

META_RESULT __stdcall META_MISC_CalDataIntegrity_StartRec_r(const int meta_handle, const unsigned int ms_timeout);

Description:

Start monitoring the NVRAM item changes.

CallBack:

NA

Return Value:

Table 6-55 The return value of META_MISC_CalDataIntegrity_StartRec

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-56 The parameter of META_MISC_CalDataIntegrity_StartRec

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

6.3.31 META_MISC_CalDataIntegrity_StopRec

Definition:

META_RESULT __stdcall META_MISC_CalDataIntegrity_StopRec(const unsigned int ms_timeout, int *rec_num);

META_RESULT __stdcall META_MISC_CalDataIntegrity_StopRec_r(const int meta_handle, const unsigned int ms_timeout, int *rec_num);

Description:

Stop monitoring the NVRAM item changes.

CallBack:

NA

Return Value:

Table 6-57 The return value of META_MISC_CalDataIntegrity_StopRec

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-58 The parameter of META_MISC_CalDataIntegrity_StopRec

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
rec_num	OUT	Number of NVRAM items are changed during the start recoring and stop recording indication.

6.3.32 META_MISC_CalDataIntegrity_AddOne

Definition:

META_RESULT __stdcall META_MISC_CalDataIntegrity_AddOne(const unsigned int ms_timeout, META_MISC_CAL_DATA_INTEGRITY_ENTRY *req);

META_RESULT __stdcall META_MISC_CalDataIntegrity_AddOne_r(const int meta_handle, const unsigned int ms_timeout, META_MISC_CAL_DATA_INTEGRITY_ENTRY *req);

typedef struct

```
{
        const char                              *LID;               // The name of logical data item ID
        //signed short    u2LIDEnumVal;
        unsigned short              u2RID;   // Record ID (the first record is 1)
} META_MISC_CAL_DATA_INTEGRITY_ENTRY;
```

Description:

Add on NVRAM items to the calibration data integrity check list.

CallBack:

NA

Return Value:

Table 6-59 The return value of META_MISC_CalDataIntegrity_AddOne

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-60 The parameter of META_MISC_CalDataIntegrity_AddOne

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
req	IN	Requested NVRAM items for adding to the calibration data integrity check list.

6.3.33 META_MISC_CalDataIntegrity_DelOne

Definition:

META_RESULT __stdcall META_MISC_CalDataIntegrity_DelOne(const unsigned int ms_timeout, META_MISC_CAL_DATA_INTEGRITY_ENTRY *req);

META_RESULT __stdcall META_MISC_CalDataIntegrity_DelOne_r(const int meta_handle, const unsigned int ms_timeout, META_MISC_CAL_DATA_INTEGRITY_ENTRY *req);

```
typedef struct
{
        const char                    *LID;           // The name of logical data item ID
        //signed short   u2LIDEnumVal;
        unsigned short          u2RID;   // Record ID (the first record is 1)
} META_MISC_CAL_DATA_INTEGRITY_ENTRY;
```

Description:

Add on NVRAM items to the calibration data integrity check list.

CallBack:

NA

Return Value:

Table 6-61 The return value of META_MISC_CalDataIntegrity_DelOne

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-62 The parameter of META_MISC_CalDataIntegrity_DelOne

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
req	IN	Requested NVRAM items for adding to the calibration data integrity check list.

6.3.34 META_MISC_CalDataIntegrity_DelAll

Definition:

META_RESULT __stdcall META_MISC_CalDataIntegrity_DelAll(const unsigned int ms_timeout);

META_RESULT __stdcall META_MISC_CalDataIntegrity_DelAll_r(const int meta_handle, const unsigned int ms_timeout);

Description:

Delete all the NVRAM items from the calibration data integrity check list.

CallBack:

NA

Return Value:

Table 6-63 The return value of META_MISC_CalDataIntegrity_DelAll

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-64 The parameter of META_MISC_CalDataIntegrity_DelAll

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

6.3.35 META_MISC_CalDataIntegrity_CheckOne

Definition:

META_RESULT __stdcall META_MISC_CalDataIntegrity_CheckOne(const unsigned int ms_timeout, META_MISC_CAL_DATA_INTEGRITY_ENTRY *req);

META_RESULT __stdcall META_MISC_CalDataIntegrity_CheckOne_r(const int meta_handle, const unsigned int ms_timeout, META_MISC_CAL_DATA_INTEGRITY_ENTRY *req);

```
typedef struct
{
        const char                  *LID;           // The name of logical data item ID
        //signed short      u2LIDEnumVal;
        unsigned short               u2RID;    // Record ID (the first record is 1)
} META_MISC_CAL_DATA_INTEGRITY_ENTRY;
```

Description:

Check the calibration data integrity of the specified NVRAM item.

CallBack:

NA

Return Value:

Table 6-65 The return value of META_MISC_CalDataIntegrity_CheckOne

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-66 The parameter of META_MISC_CalDataIntegrity_CheckOne

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
req	IN	Requested NVRAM items for checking the calibration data integrity.

6.3.36 META_MISC_CalDataIntegrity_CheckAll

Definition:

META_RESULT __stdcall META_MISC_CalDataIntegrity_CheckAll(const unsigned int ms_timeout, META_MISC_CAL_DATA_INTEGRITY_CHECK_CNF_T *cnf);

META_RESULT __stdcall META_MISC_CalDataIntegrity_CheckAll_r(const int meta_handle, const unsigned int ms_timeout, META_MISC_CAL_DATA_INTEGRITY_CHECK_CNF_T *cnf);

typedef struct

```
{

        bool            bAllPass;  // true: check pass, false: no items or check fail
        unsigned short    u2LastLID;  // valid when bAllPass == false
        unsigned short    u2LastRID;  // valid when bAllPass == false
} META_MISC_CAL_DATA_INTEGRITY_CHECK_CNF_T;
```

Description:

Check the calibration data integrity of all the NVRAM items in the calibration data integrity check list.

CallBack:

NA

Return Value:

Table 6-67 The return value of META_MISC_CalDataIntegrity_CheckAll

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-68 The parameter of META_MISC_CalDataIntegrity_CheckAll

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
cnf	OUT	Indication of calibration data integrity check.

6.3.37 META_MISC_GetRID

Definition:

META_RESULT __stdcall META_MISC_GetRID(const unsigned int ms_timeout,unsigned char *u1Rid,const unsigned int ui_RidLen);

META_RESULT __stdcall META_MISC_GetRID_r(const int meta_handle, const unsigned int ms_timeout,unsigned char *u1Rid,const unsigned int ui_RidLen);

Description:

Query the chip RID from the target

CallBack:

NA

Return Value:

Table 6-69 The return value of META_MISC_GetRID

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-70 The parameter of META_MISC_GetRID

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
u1Rid	OUT	RID string queried from the target in HEX format
ui_RidLen	IN	The requested length of RID string 1 ~ 16 (unit: bytes)

6.3.38 META_MISC_CheckGeminiPlusSIMInserted

Definition:

META_RESULT __stdcall META_MISC_CheckGeminiPlusSIMInserted(const unsigned int ms_timeout, unsigned char sim_module_index, unsigned char* inserted);

META_RESULT __stdcall META_MISC_CheckGeminiPlusSIMInserted_r(const int meta_handle, const unsigned int ms_timeout, unsigned char sim_module_index, unsigned char* inserted);

Description:

SIM card module test function.

CallBack:

NA

Return Value:

Table 6-71 The return value of META_MISC_CheckGeminiPlusSIMInserted

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-72 The parameter of META_MISC_CheckGeminiPlusSIMInserted

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
sim_module_index	IN	The specified SIM module index for test (0: SIM1, 1: SIM2 …)
inserted	IN	The SIM module test result (0: SIM card is inserted, 1: SIM card is not inserted)

6.3.39 META_Check_SmartPhoneModem_support

Definition:

META_RESULT __stdcall META_RESULT __stdcall META_Check_SmartPhoneModem_support (unsigned int ms_timeout);

META_RESULT __stdcall META_RESULT __stdcall META_Check_SmartPhoneModem_support_r (const int meta_handle, unsigned int ms_timeout);

Description:

Check to see whether the code base is for smartphone or not.

Return Value:

Table 6-73 The return value of META_Check_SmartPhoneModem_support

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-74 The parameter of META_Check_SmartPhoneModem_support

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

6.3.40 META_MISC_EX_SetCommandToSystem

Definition:

META_RESULT __stdcall META_MISC_EX_SetCommandToSystem(unsigned int ms_timeout, const SYSTEM_EX_CMD command);

META_RESULT __stdcall META_MISC_EX_SetCommandToSystem_r(const int meta_handle, unsigned int ms_timeout, const SYSTEM_EX_CMD command);

```
typedef enum {
   SET_DL_FLAG = 0,  // set download flag = enter download mode when booting.
   CLR_DL_FLAG       // clear download flag = enter normal mode when booting.
}SYSTEM_EX_CMD;
```

Description:

The API command system according to SYSTEM_EX_CMD command.

Return Value:

Table 6-75 The return value of META_MISC_EX_SetCommandToSystem

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_TIMEOUT	Wait for target confirmation timeout.

Parameter:

Table 6-76 The parameter of META_MISC_EX_SetCommandToSystem

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Function timeout value. (in milliseconds)
command	IN	The command to control system.

Sample Code:

```
// Command target to enter download mode when booting

If(META_SUCCESS  != META_MISC_EX_SetCommandToSystem_r(meta_handle, 3000, SET_DL_FLAG))

{

        Error_log("META_MISC_EX_SetCommandToSystem_r failed!");

        Return;

}

// Command target to enter normal mode when booting

If(META_SUCCESS  != META_MISC_EX_SetCommandToSystem_r(meta_handle, 3000, CLR_DL_FLAG))

{

        Error_log("META_MISC_EX_SetCommandToSystem_r failed!");

        Return;

}
```

6.3.41 META_MISC_EX_BackupCalibrationToStorage

Definition:

META_RESULT __stdcall META_MISC_EX_BackupCalibrationToStorage(const unsigned int ms_timeout, unsigned int storage_mode, unsigned int *status);

META_RESULT __stdcall META_MISC_EX_BackupCalibrationToStorage_r(const int meta_handle, const unsigned int ms_timeout, unsigned int storage_mode, unsigned int *status);

Description:

The API triggers NVRAM module on the target side to backup the calibration data to SDS.

Return Value:

Table 6-77 The return value of META_MISC_EX_BackupCalibrationToStorage

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_TIMEOUT	Wait for target confirmation timeout.

Parameter:

Table 6-78 The parameter of META_MISC_EX_BackupCalibrationToStorage

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Function timeout value. (in milliseconds)
storage_mode	IN	The parameter used for NVRAM backup requests. (currently, only 0 is used for backup)
status	OUT	The return status code from NVRAM module.

Sample Code:

```
  unsigned int sds_status;
  if(META_SUCCESS == META_QueryIfFunctionSupportedByTarget_r(0, 1500, "META_MISC_EX_BackupCalibrationToStorage_r"))
  {
    {
      if(META_SUCCESS == META_MISC_EX_BackupCalibrationToStorage_r(0, 20000, 0, &sds_status))
      {
        // success
      }
      else
      {
        // failed
      }
    }
  }
```

6.3.42 META_MISC_EX_BackupNvramItemToStorage

Definition:

META_RESULT __stdcall META_MISC_EX_BackupNvramItemToStorage(const unsigned int ms_timeout, const char* lid, unsigned int *status);

META_RESULT __stdcall META_MISC_EX_BackupNvramItemToStorage_r(const int meta_handle, const unsigned int ms_timeout, const char* lid, unsigned int *status);

Description:

The API triggers NVRAM module on the target side to backup the specified NVRAM item from filesystem to SDS. The NVRAM item must be specified in NVRAM_SDS_SPLIT_LIST defined in nvram_ex_io.c

Return Value:

Table 6-79 The return value of META_MISC_EX_BackupNvramItemToStorage

Return value	Description
META_SUCCESS	The command has been successfully called, but the detailed result must be checked by the output parameter.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	The parameter is invalid either lid or status is NULL pointer.
META_NVRAM_DB_IS_NOT_LOADED_YET	The NVRAM database is not loaded yet.
META_LID_INVALID	The given LID name can not be found in the NVRAM database.

Parameter:

Table 6-80 The parameter of META_MISC_EX_BackupNvramItemToStorage

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Function timeout value. (in milliseconds)
lid	IN	A null terminated string representing the given LID to be backup to SDS. (eg. “NVRAM_EF_IMEI_IMEISV_LID”);
status	OUT	The return status code from NVRAM module. (0: successfully done, otherwise: failed)

Sample Code:

```
unsigned int sds_status;
```

```
  unsigneg                                   long                                   addr;
  if(META_SUCCESS != META_NVRAM_Init_r(0, “NVRAM_DB_PATH”, &addr))
  {
     // error handling
  }
  if(META_SUCCESS          !=          META_QueryIfFunctionSupportedByTarget_r(0,          1500,
"META_MISC_EX_BackupNvramItemToStorage_r"))
  {
     // error handling
  }
     {
         if(META_SUCCESS          ==          META_MISC_EX_BackupNvramItemToStorage_r(0,          20000,
"NVRAM_EF_IMEI_IMEISV_LID", &sds_status))
         {
            // success
         }
         else
         {
            // failed
         }
     }
  }
```

6.3.43 META_MISC_EX_RestoreNvramItemFromStorage

Definition:

META_RESULT __stdcall META_MISC_EX_RestoreNvramItemFromStorage(const unsigned int ms_timeout, const char* lid, unsigned int *status);

META_RESULT __stdcall META_MISC_EX_RestoreNvramItemFromStorage_r(const int meta_handle, const unsigned int ms_timeout, const char* lid, unsigned int *status);

Description:

The API triggers NVRAM module on the target side to restore the specified NVRAM item from SDS to filesystem. The NVRAM item must be specified in NVRAM_SDS_SPLIT_LIST defined in nvram_ex_io.c

Return Value:

Table 6-81 The return value of META_MISC_EX_RestoreNvramItemFromStorage

Return value	Description
META_SUCCESS	The command has been successfully called, but the detailed result must be checked by the output parameter.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	The parameter is invalid either lid or status is NULL pointer.
META_NVRAM_DB_IS_NOT_LOADED_YET	The NVRAM database is not loaded yet.
META_LID_INVALID	The given LID name can not be found in the NVRAM database.

Parameter:

Table 6-82 The parameter of META_MISC_EX_RestoreNvramItemFromStorage

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Function timeout value. (in milliseconds)
lid	IN	A null terminated string representing the given LID to be backup to SDS. (eg. “NVRAM_EF_IMEI_IMEISV_LID”);
status	OUT	The return status code from NVRAM module. (0: successfully done, otherwise: failed)

Sample Code:

```
unsigned int sds_status;
unsigneg long addr;
if(META_SUCCESS != META_NVRAM_Init_r(0, “NVRAM_DB_PATH”, &addr))
{
    // error handling
}
if(META_SUCCESS != META_QueryIfFunctionSupportedByTarget_r(0, 1500,
"META_MISC_EX_RestoreNvramItemFromStorage_r"))
{
    // error handling
```

```
  }
    {
      if(META_SUCCESS == META_MISC_EX_RestoreNvramItemFromStorage_r(0, 20000,
"NVRAM_EF_IMEI_IMEISV_LID", &sds_status))
      {
        // success
      }
      else
      {
        // failed
      }
    }
  }
```

6.4 Exported Utility Functions

6.4.1 META_Util_CheckTargetRequiredVersion

Definition:

```
META_RESULT __stdcall META_Util_CheckTargetRequiredVersion(unsigned int ms_timeout, const
META_UTIL_CHECK_TARGET_VER_REQ_T *req, META_UTIL_CHECK_TARGET_VER_CNF_T *cnf );

META_RESULT __stdcall META_Util_CheckTargetRequiredVersion_r(const int meta_handle, unsigned int
ms_timeout, const META_UTIL_CHECK_TARGET_VER_REQ_T *req, META_UTIL_CHECK_TARGET_VER_CNF_T
*cnf );


typedef enum
{
  META_VERSION_USER_DEFINE
 ,META_VERSION_META_DLL_UTIL_VER
 ,VER_TYPE_END
}META_VERSION_TYPE;

typedef struct
{

        META_VERSION_TYPE   m_eVerType;

        bool            b_AssertWhenVerCheckFail;  // a flag to enable/disable target assert when version check
fail

        unsigned int       m_u4MainVersion;  // valid when m_eVerType = META_VERSION_USER_DEFINE
        unsigned int       m_u4MinorVersion; // valid when m_eVerType = META_VERSION_USER_DEFINE
        unsigned int       m_u4BuildNum;    // valid when m_eVerType = META_VERSION_USER_DEFINE

}META_UTIL_CHECK_TARGET_VER_REQ_T;

typedef struct
{
        bool       m_bCheckPass;
        unsigned int m_u4TargetMainVersion;
        unsigned int m_u4TargetMinorVersion;
        unsigned int m_u4TargetBuildNum;
}META_UTIL_CHECK_TARGET_VER_CNF_T;
```

Description:

Perform version check between PC-side's tool or META DLL version with Target's FT task.

.

Return Value:

Table 6-83 The return value of META_Util_CheckTargetRequiredVersion

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-84 The parameter of META_Util_CheckTargetRequiredVersion

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
req	IN	PC-sides tool version information or META DLL version information for version check between tools and phones.
cnf	IN/OUT	The version check result between phones and PC-side tools.

6.4.2 META_Util_SetTargetAssertCheckParas

Definition:

META_RESULT __stdcall META_Util_SetTargetAssertCheckParas(unsigned int ms_timeout, const META_UTIL_SET_ASSERT_CHECK_PARAs_REQ_T *req);

META_RESULT __stdcall META_Util_SetTargetAssertCheckParas_r(const int meta_handle, unsigned int ms_timeout, const META_UTIL_SET_ASSERT_CHECK_PARAs_REQ_T *req);

```
typedef struct
{
        bool  b_TargetAssertCheckFlag;
        bool  b_SetCurRecvMsgTimes;
        unsigned char m_u1CurRecvMsgTimes;  // valid when b_SetCurRecvMsgTimes = true
}META_UTIL_SET_ASSERT_CHECK_PARAs_REQ_T;
```

Description:

Enable/Disable the target-assert-related parameters of the phone. This API is flexible for

PC-side tools to make target FT task assert for version control. Note: A message counter is adopted in some projects (w0918 MAUI/09A later) when the assert check flag of phones' FT task is ON.

.

Return Value:

Table 6-85 The return value of META_Util_SetTargetAssertCheckParas

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-86 The parameter of META_Util_SetTargetAssertCheckParas

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
req	IN	The phone's assert check parameter settings.

6.4.3 META_Util_CheckIfTargetNVSecOn

Definition:

META_RESULT __stdcall META_Util_CheckIfTargetNVSecOn(unsigned int ms_timeout, bool *bOn);

META_RESULT __stdcall META_Util_CheckIfTargetNVSecOn_r(const int meta_handle, unsigned int ms_timeout, bool *bOn);

Description:

Check whether the NVRAM security is turned on in the target.

.

Return Value:

Table 6-87 The return value of META_Util_CheckIfTargetNVSecOn

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-88 The parameter of META_Util_CheckIfTargetNVSecOn

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
bOn	OUT	The indication of whether the NVRAM security is turned on the target.

6.4.4 META_Util_RebootToNormalMode

Definition:

META_RESULT __stdcall META_Util_RebootToNormalMode(unsigned int ms_timeout, unsigned short timeout);

META_RESULT __stdcall META_Util_RebootToNormalMode_r(const int meta_handle, unsigned int ms_timeout, unsigned short timeout);

Description:

Reboot the target from META mode to Normal mode after timeout ms.

.

Return Value:

Table 6-89 The return value of META_Util_RebootToNormalMode

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-90 The parameter of META_Util_RebootToNormalMode

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
timeout	IN	The timeout value to reboot.

6.4.5 META_Util_QueryBTWiFiSingleAntennaCap

Definition:

META_RESULT __stdcall META_Util_QueryBTWiFiSingleAntennaCap(unsigned int ms_timeout, unsigned short timeout);

META_RESULT __stdcall META_Util_QueryBTWiFiSingleAntennaCap_r(const int meta_handle, unsigned int ms_timeout, unsigned short timeout);

Description:

Query the target whether it supports BT/WiFi Single Antenna capability.

.

Return Value:

Table 6-91 The return value of META_Util_QueryBTWiFiSingleAntennaCap

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-92 The parameter of META_Util_QueryBTWiFiSingleAntennaCap

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
bOn	OUT	Indication of the BT/WiFi Single Antenna capability is On or Not.

6.4.6 META_Util_SetAntennaPathToBT

Definition:

META_RESULT __stdcall META_Util_SetAntennaPathToBT(unsigned int ms_timeout, unsigned short timeout);

META_RESULT __stdcall META_Util_SetAntennaPathToBT_r(const int meta_handle, unsigned int ms_timeout, unsigned short timeout);

Description:

Switch the BT/WiFi antenna path for BT RF TX/RX.

.

Return Value:

Table 6-93 The return value of META_Util_SetAntennaPathToBT

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-94 The parameter of META_Util_SetAntennaPathToBT

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

6.4.7 META_Util_SetAntennaPathToWiFi

Definition:

META_RESULT __stdcall META_Util_SetAntennaPathToWiFi(unsigned int ms_timeout, unsigned short timeout);

META_RESULT __stdcall META_Util_SetAntennaPathToWiFi_r(const int meta_handle, unsigned int ms_timeout, unsigned short timeout);

Description:

Switch the BT/WiFi antenna path for WiFi RF TX/RX.

.

Return Value:

Table 6-95 The return value of META_Util_SetAntennaPathToWiFi

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-96 The parameter of META_Util_SetAntennaPathToWiFi

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

6.4.8 META_Util_QueryVpaVoltageList

Definition:

META_RESULT __stdcall META_Util_QueryVpaVoltageList(const unsigned int ms_timeout, MetaVpaVoltageList* vpaVoltageList);

META_RESULT __stdcall META_Util_QueryVpaVoltageList_r(const int meta_handle, const unsigned int ms_timeout, MetaVpaVoltageList* vpaVoltageList);

```
typedef struct
{
    /// number of elements in the list
    unsigned int validNumber;
    /// voltage list (unit: micro volt 10^-6)
    unsigned int voltageList[255];
    /// register value of each voltageList
    unsigned int registerValue[255];
}MetaVpaVoltageList;
```

Description:

Query usable VPA voltage list from the target.

.

Return Value:

Table 6-97 The return value of META_Util_QueryVpaVoltageList

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-98 The parameter of META_Util_QueryVpaVoltageList

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

Parameter	IN/OUT	Description
vpaVoltageList	OUT	The usable VPA voltage setting on the target

6.5 Exported Functions for Initialization

6.5.1 META_Init

Definition:

META_RESULT __stdcall META_Init(const META_Error_CallBack cb)

Description:

Initialize META-DLL.

Return Value:

Table 6-99 The return value of Exported Functions for Initialization

Return value	Description
META_SUCCESS	Success
META_FAILED	Initialization failed due to allocate resource failed.

Parameter:

Table 6-100 The parameter of Exported Functions for Initialization

Parameter	IN/OUT	Description
Cb	IN	Function pointer of error handler. The error handler is called when META_DLL finds some error.

6.5.2 META_Init_Ex_2_r

Definition:

META_RESULT __stdcall META_Init_Ex_2_r(const int meta_handle, const META_Error_CallBack err_cb, const META_MD_Query_CallBack md_query_cb, void* md_query_arg, const META_MD_Switch_CallBack md_switch_cb, void* md_switch_arg, const META_MDTYPE_Switch_CallBack mdtype_switch_cb, void* mdtype_switch_arg)

Description:

Initialize META-DLL. This API is enhanced for world phone feature (multiple SW image/MD type feature.)

* MUST assign mdtype_switch_cb as a valid pointer to init the com port state of meta handler, if user want to use the mate handler in world phone feature.

Return Value:

Table 6-101 The return value of META_Init_Ex_2_r

Return value	Description
META_SUCCESS	Success
META_FAILED	Initialization failed due to allocate resource failed.

Parameter:

Table 6-102 The parameter of META_Init_Ex_2_r

Parameter	IN/OUT	Description
err_cb	IN	Function pointer of error handler. The error handler is called when META_DLL finds some error.
md_query_cb	IN	Callback function pointer of modem information query handler. The handler is called when META_DLL query modem capability.
md_query_arg	IN	Arguments of modem information query handler. The handler is called when META_DLL query modem capability.
md_switch_cb	IN	Callback function pointer of modem switch handler. The handler is called if there's registered callback function.
md_switch_arg	IN	Arguments of modem switch handler.
mdtype_switch_cb	IN	Callback function pointer of modem type switch handler. The handler is called if there's registered callback function.
mdtype_switch_arg	IN	Arguments of modem type switch handler.

Note:

Parameter of md_query_cb needs be implemented with bring some modem information during AP connection period. And structure of META_MD_Query_Result_T will be set in this callback function.

For dual-talk nad worldphone developers, here is an example pseudo code to implement this callback function (Ex, MdQueryHandler()).

Example:

```
typedef struct
{
   unsigned int number_of_md:8;
   unsigned int active_md_idx:8;
   unsigned int multi_talk:1;
```

```
    unsigned int multi_frame_type:1;
    unsigned int number_of_mdSwImg:4;
    unsigned int active_mdtype_idx:4;
    unsigned int multi_mdtype:1;
    unsigned int reserved:5;
} META_MD_Query_Result_T;

META_MD_Query_Result_T __stdcall MdQueryHandler(void* MdQuery_CB_Arg)
{
    META_MD_Query_Result_T result;
    result.number_of_md = Number Of Modem (these information comes from AP connection period);
    result.active_md_idx = Active Modem Index (these information comes from AP connection period);
    result.number_of_mdSwImg = Number Of Modem Type (these information comes from AP connection period);
    int active_mdtype = Active Modem Type Index (these information comes from AP connection period);
    result.active_mdtype_idx = Active Modem Type Index (these information comes from AP connection period);
    result.multi_talk  = (result.active_md_idx!=0||result.number_of_md>=2)?true:false;
    result.multi_frame_type =  (these information comes from AP connection period);
    result.multi_mdtype = (active_mdtype!=0||result.number_of_mdSwImg>=2)?true:false;
    return result;
}
int __stdcall MdTypeSwitchHandler(META_MDTYPE_Switch_Param_T mdtype_switch_param, void* MdTypeSwitch_CB_Arg)
{
    return 1;  //Not be NULL
}

MetaResult = META_Init_Ex_2_r( meta_handle, NULL, ::MdQueryHandler, NULL, NULL, NULL, ::MdTypeSwitchHandler, NULL);
```

6.5.3

6.5.4 META_SetSysTraceCallback

Definition:

META_RESULT __stdcall META_SetSysTraceCallback(const META_SysTrace_CallBack sys_cb)

Description:

Register a callback function to receive system trace information. It's very useful when target assert.

Note:

You must call META_Init before calling this function.

Callback:

typedef void (__stdcall *META_SysTrace_CallBack)(const char *sys_trace);

Return Value:

Table 6-103 The return value of META_SetSysTraceCallback

Return value	Description
META_SUCCESS	Success
META_INVALID_ARGUMENTS	sys_cb is NULL.

Parameter:

Table 6-104 The parameter of META_SetSysTraceCallback

Parameter	IN/OUT	Description
sys_cb	IN	Function pointer of system trace callback. The system trace callback is called when target sent system trace frame to PC side.

6.5.5 META_Deinit

Definition:

void __stdcall META_Deinit()

Description:

Deinitialize META-DLL.

6.5.6 META_ConnectWithTarget

Definition:

```
        META_RESULT  __stdcall META_ConnectWithTarget(
                                        const META_Connect_Req  *req,
                                        int *p_bootstop,
                                        META_Connect_Report  *p_report)
```

Structure Definition:

```
        typedef enum  {
            META_BAUD2400 = 0,
            META_BAUD4800,
            META_BAUD9600,
            META_BAUD14400,
            META_BAUD19200,
            META_BAUD57600,
            META_BAUD115200,
            META_BAUD230400,
            META_BAUD460800,
            META_BAUD921600,
            META_BAUD_END = 0xFF
        } META_COMM_BAUDRATE;

        typedef enum {
            META_NO_FLOWCTRL = 0,// no flow control
            META_SW_FLOWCTRL,             // enable S/W flow control
            META_FLOWCTRL_END
        } META_FLOWCTRL;

        #define META_BOOT_INFINITE          0xFFFFFFF
```

```
typedef struct {

    BBCHIP_TYPE              m_bbchip_type;
    EXT_CLOCK                m_ext_clock;

    unsigned int      m_ms_boot_timeout;
    unsigned int      m_max_start_cmd_retry_count;

    // This callback function will be invoke after COM port is opened
    // You can do some initialization by using this callback function.
    CALLBACK_COM_INIT_STAGE                  m_cb_com_init_stage;
    void *                                          m_cb_com_init_stage_arg;

    // This callback function will be invoke after BootROM start cmd is passed.
    // You can issue other BootROM command by brom_handle and hCOM which provides callback
arguments,
    // or do whatever you want otherwise.
    CALLBACK_IN_BROM_STAGE                   m_cb_in_brom_stage;
    void *                                          m_cb_in_brom_stage_arg;

    // speed-up BootROM stage baudrate
    _BOOL   m_speedup_brom_baudrate;

    // Application's window handle to send WM_BROM_READY_TO_POWER_ON_TGT message
    HWND   m_ready_power_on_wnd_handle;
    void *    m_ready_power_on_wparam;
    void *    m_ready_power_on_lparam;
    // Serial Link Authentication
    AUTH_HANDLE_T                               m_auth_handle;  // AUTH file handle
    CALLBACK_SLA_CHALLENGE                      m_cb_sla_challenge;
    void *                                           m_cb_sla_challenge_arg;
    CALLBACK_SLA_CHALLENGE_END             m_cb_sla_challenge_end;
    void *                                           m_cb_sla_challenge_end_arg;

   // Security Certificate
    SCERT_HANDLE_T                                   m_scert_handle; // SCERT file handle

    // use USB Cable
    _BOOL                                            m_usb_enable;

} BOOT_META_ARG;

typedef struct {
    int                    com_port;
    META_COMM_BAUDRATE   baudrate[11];
    META_FLOWCTRL          flowctrl;
    BOOT_META_ARG          boot_meta_arg;
    unsigned int           ms_connect_timeout;
} META_Connect_Req;

typedef struct {

    BBCHIP_TYPE            m_bbchip_type;
    char                   m_bbchip_name[32];
    unsigned short    m_bbchip_hw_ver;
```

```
        unsigned short    m_bbchip_sw_ver;
        unsigned short    m_bbchip_hw_code;
        EXT_CLOCK                 m_ext_clock;
        unsigned char     m_bbchip_secure_ver;
        unsigned char     m_bbchip_bl_ver;
        unsigned int      m_fw_ver_len;
        char                      m_fw_ver[64];

        unsigned char     m_msp_err_code;

    } BOOT_RESULT;
    typedef struct {
        META_COMM_BAUDRATE  final_baudrate;
        unsigned int              meta_ver_required_by_target;
        BOOT_RESULT               boot_result;
        STATUS_E                  boot_meta_ret;
    } META_Connect_Report;
```

Description:

This function will open COM port and boot up target to META mode.

* MUST call META_DisconnectWithTarget or META_COMM_Stop to init the com port state of meta handler, If you want to reuse the mate handler. Otherwise, the next connect operation will fail.

Return Value:

Table 6-105 The return value of META_ConnectWithTarget

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-106 The parameter of META_ConnectWithTarget

Parameter	IN/OUT	Description
m_bbchip_type	IN	Baseband chip type; refer to BBCHIP_TYPE in mtk_mcu.h To enable baseband chip auto detection mechanism, set to AUTO_DETECT_BBCHIP
m_ext_clock	IN	External clock rate; refer to EXT_CLOCK in mtk_mcu.h To enable external clock auto detection mechanism, set to AUTO_DETECT_EXT_CLOCK
m_ms_boot_timeout	IN	Boot until timeout with a unit of ms(millisecond) .
m_max_start_cmd_retry_count	IN	When the download cable is plugged in or removed, the UART TX and RX channels may cross over, resulting in temporary interference and causing BROM_DLL to send the boot ROM start command prematurely when the target boot ROM has not yet powered up. To avoid this problem, set the max_start_cmd_retry_count to the number of retry attempts for the boot ROM start command. For the default value, set to the defined constant DEFAULT_BROM_START_CMD_RETRY_COUNT. If set to zero, the start command is not reattempted.
m_cb_in_brom_stage	IN	CALLBACK_IN_BROM_STAGE callback is invoked after BootROM start cmd is passed. Other boot ROM commands can be issued the brom_handle and hCOM commands, which provide the callback arguments.
m_cb_in_brom_stage_arg	IN	User argument for this callback function.
m_speedup_brom_baudrate	IN	_TRUE: The BROMDLL doubles the boot ROM stage baud rate to speed up downloading DA into the target's internal SRAM. _FALSE: The boot ROM stage baud rate remains the same.
m_ready_power_on_wnd_handle	IN	Application's window handle to send WM_BROM_READY_TO_POWER_ON_TGT message When Boot_META is starting to polling BOOT ROM start command, it sends this message to notify application to power on target.
m_ready_power_on_wparam	IN	WPARAM is a type of Windows messages. WPARAM is typically used to store small pieces of information, such as flags.
m_ready_power_on_lparam	IN	LPARAM is a type of Windows messages. LPARAM is typically used to store an object if it is needed by the message.
m_auth_handle	IN	refer to BROM_DLL Development Kit User Manual for detailed usage.
m_cb_sla_challenge	IN	refer to BROM_DLL Development Kit User Manual for detailed usage.

Parameter	IN/OUT	Description
m_cb_sla_challenge_arg	IN	User argument for this callback function.
m_cb_sla_challenge_end	IN	refer to BROM_DLL Development Kit User Manual for detailed usage.
m_cb_sla_challenge_end_arg	IN	User argument for this callback function.
m_scert_handle	IN	The securitye certificate handle.
m_usb_enable	IN	USB connection enable flag.

Table 6-107 The parameter of META_ConnectWithTarget

Parameter	IN/OUT	Description
m_bbchip_type	OUT	Target’s baseband chip type; refer to BBCHIP_TYPE in mtk_mcu.h
m_bbchip_name[32]	OUT	Target’s baseband chip name with limited length of 32 bytes.
m_bbchip_hw_ver	OUT	Target’s baseband chip hardware version.
m_bbchip_sw_ver	OUT	Target’s baseband chip software version.
m_bbchip_hw_code	OUT	Target’s baseband chip hardware code.
m_ext_clock	OUT	Target’s external clock rate; refer to EXT_CLOCK in mtk_mcu.h
m_bbchip_secure_ver	OUT	Target’s secure platform version.
m_bbchip_bl_ver	OUT	Target’s bootloader version.
m_fw_ver_len	OUT	Target’s firmware version string length.
m_fw_ver	OUT	Target’s firmware version string.
m_msp_err_code	OUT	MTK Secure Platform (MSP) return code.

Table 6-108 The parameter of META_ConnectWithTarget

Parameter	IN/OUT	Description
req->com_port	IN	COM port number.
req->baudrate	IN	Baud rate array. META_DLL will enumerate target’s baud rate according to this array. The last element of array must be META_BAUD_END. For example, if you want to enumerate 115200 and 921600. You have to fill like this:

Parameter	IN/OUT	Description
		Baudrate[11] = { META_BAUD115200, META_BAUD921600, META_BAUD_END, …, } The rest of elements after META_BAUD_END are ignored, you can just leave them alone.
req->flowctrl	IN	UART flow control type. It should be META_SW_FLOWCTRL normally.
req-> boot_meta_arg	IN	Refer to BOOT_META_ARG
req->ms_connect_timeout	IN	Sync with target timeout value. When target passed BootROM and entered META mode, then META_ConnectWithTarget will keep sending message to query if target is ready to accept META command. Only when target has response or reach this timeout value, the query operation will stop.
p_bootstop	IN	The pointer to an integer variable. You can forcedly stop the BootROM polling by set the variable to BOOT_STOP. Please refer BOOT_STOP in brom.h
p_report->final_baudrate	IN/OUT	The current baud rate of target.
p_report-> meta_ver_required_by_target	IN/OUT	The META_DLL version required by target.
p_report->boot_result	IN/OUT	Refer to BOOT_RESULT.
P_report->boot_meta_ret	IN/OUT	Return code from BROM_DLL. Please use StatusToString function to convert error code to error string.

6.5.7 META_DisconnectWithTarget

Definition:

META_RESULT __stdcall META_DisconnectWithTarget( )

Description:

This function will send META_ShutDownTarget command to target and them close the COM port.

Return Value:

Table 6-109 The return value of META_DisconnectWithTarget

Parameter	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

6.5.8 META_ShutDownTarget

Definition:

META_RESULT __stdcall META_ShutDownTarget ( )

Description:

This function will send command to shutdown target by pull down BB WakeUp.

Return Value:

Table 6-110 The return value of META_ShutDownTarget

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

6.5.9 META_ConnectWithTargetByUSB

Definition:

```
META_RESULT __stdcall META_ConnectWithTargetByUSB(
                                        const META_ConnectByUSB_Req  *req,
                                        int *p_bootstop,
                                        META_ConnectByUSB_Report                    *p_report);
```

Structure Definition:

```
typedef struct {
    int                 com_port;
    BOOT_META_ARG       boot_meta_arg;   // [BootROM] please refer to brom.h
    unsigned int        ms_connect_timeout;
      // [META] META stage sync timeout value (after BootROM negotiation pass)
} META_ConnectByUSB_Req;


typedef struct {
      unsigned int      meta_ver_required_by_target;      // [META] Target required META_DLL version.
      BOOT_RESULT       boot_result;                // [BootROM] boot-up result.
      STATUS_E          boot_meta_ret;              // [BROM_DLL] The return code of Boot_META function.
} META_ConnectByUSB_Report;
```

Description:

This function will open USB COM port and boot up target to META mode.
* MUST call META_DisconnectWithTarget or META_COMM_Stop to init the com port state of meta handler, If you want to reuse the mate handler. Otherwise, the next connect operation will fail.

Return Value:

Table 6-111 The return value of META_ConnectWithTargetByUSB

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-112 The parameter of META_ConnectWithTargetByUSB

Parameter	IN/OUT	Description
req-	IN	META_ConnectByUSB_Req specifies the connection settings
p_bootstop	IN	The pointer to an integer variable. You can forcedly stop the BootROM polling by set the variable to BOOT_STOP. Please refer BOOT_STOP in brom.h
p_report	IN/OUT	META_ConnectByUSB_Report specfies the connection result.

6.5.10 META_GetDynamicUSBComPort

Definition:

META_RESULT __stdcall META_GetDynamicUSBComPort(unsigned int ms_scan_timeout, unsigned short *com_port, int *p_scanstop);

Structure Definition:

#define ENUM_USB_STOP 9876

Description:

This function will continuously query the registry ("HARDWARE\\DEVICEMAP\\SERIALCOMM") to see if there is any new USB com port.

Return Value:

Table 6-113 The return value of META_GetDynamicUSBComPort

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-114 The parameter of META_GetDynamicUSBComPort

Parameter	IN/OUT	Description
com_port	IN/OUT	The com port is found
p_scanstop	IN	A flag to stop the function, if (*p_scanstop) == ENUM_USB_STOP

6.5.11 META_ConnectInMetaModeByUSB

Definition:

```
META_RESULT  __stdcall META_ConnectInMetaModeByUSB (
                                    const META_ConnectByUSB_Req  *req,
                                    int *p_bootstop,
                                    META_ConnectByUSB_Report                      *p_report);
```

Structure Definition:

```
typedef struct {
    int                  com_port;
    BOOT_META_ARG        boot_meta_arg;   // don't care
    unsigned int         ms_connect_timeout;
      // [META] META stage sync timeout value (after BootROM negotiation pass)
} META_ConnectByUSB_Req;


typedef struct {
      unsigned int      meta_ver_required_by_target;        // [META] Target required META_DLL version.
      BOOT_RESULT     boot_result;                      // [BootROM] boot-up result.
      STATUS_E          boot_meta_ret;                  // [BROM_DLL] The return code of Boot_META function.
} META_ConnectByUSB_Report;
```

Description:

This function will open USB COM port and assume the target is already in META mode.

* MUST call META_DisconnectWithTarget or META_COMM_Stop to init the com port state of meta handler, If you want to reuse the mate handler. Otherwise, the next connect operation will fail.

Return Value:

Table 6-115 The return value of META_ConnectInMetaModeByUSB

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-116 The parameter of META_ConnectInMetaModeByUSB

Parameter	IN/OUT	Description
req-	IN	META_ConnectByUSB_Req specifies the connection settings
p_bootstop	IN	The pointer to an integer variable. You can forcedly stop the BootROM polling by set the variable to BOOT_STOP. Please refer BOOT_STOP in brom.h
p_report	IN/OUT	META_ConnectByUSB_Report specfies the connection result.

6.5.12 META_ConnectWithMultiModeTarget

Definition:

```
META_RESULT  __stdcall META_ META_ConnectWithMultiModeTarget (
                                  META_Connect_Ex_Req*  req,
                                  const unsigned int requestLengthlength,
                                  int *p_bootstop,
                                  META_Connect_Report                    *p_report);
```

Structure Definition:

```
    typedef struct
    {
      int            com_port;
      META_COMM_BAUDRATE    baudrate[12]; // [META] META stage baudrate polling array, it must end
    with META_BAUD_END.
      META_FLOWCTRL       flowctrl;    // [META] META stage uart flow control type.
```

```
   BOOT_META_ARG        boot_meta_arg;   // [BootROM] please refer to brom.h
   unsigned int          ms_connect_timeout; // [META] META stage sync timeout value (after BootROM negotiation pass)
   unsigned int          usb_enable: 1;   // [META] Connect target with UART or USB, 0: UART 1: USB others:reserved
   unsigned int          InMetaMode: 1;  // [META] Decide that need boot META or not 0:need boot META 1:already in meta mode
   unsigned int          escape: 1;      // [META] Force to connect target with escaping
   unsigned int          close_com_port: 1;  // [META] Choose to close com port or handle
   META_MODE_TRACE_PARA_T trace_para;    // [META] META mode trace parameters
   unsigned int          protocol: 4;          // [META] Only for MultiMode connection API. When InMetaMode==true, connect target with different protocol 0||1:TST 2:DHL
   unsigned int          channel_type: 4;   // [META] Only for MultiMode connection API. Connect target with different channel type, 0||1: native channel, 2: tunneling, 3: tunneling with check sum ignored
} META_Connect_Ex_Req;


typedef struct {
    META_COMM_BAUDRATE  final_baudrate;
    unsigned int            meta_ver_required_by_target;
    BOOT_RESULT             boot_result;
    STATUS_E                boot_meta_ret;
} META_Connect_Report;
```

Description:

This function will integrate multiple mode for opening USB port and COM port, and boot up target to META mode.

Return Value:

Table 6-117 The return value of META_ConnectWithMultiModeTarget

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-118 The parameter of META_ConnectWithMultiModeTarget

Parameter	IN/OUT	Description
req-	IN	META_Connect_Ex_Req specifies the connection settings
requestLengthlength	IN	Length of META_Connect_ Ex_Req structure
p_bootstop	IN	The pointer to an integer variable. You can forcedly stop the BootROM polling by set the variable to BOOT_STOP. Please refer BOOT_STOP in brom.h
p_report	IN/OUT	META_Connect_Report specfies the connection result.

6.5.13 META_SwitchCurrentModem

Definition:

META_RESULT __stdcall META_SwitchCurrentModem(const unsigned int ms_timeout, const unsigned int md_index);

META_RESULT __stdcall META_SwitchCurrentModem_r(const int meta_handle, const unsigned int ms_timeout, const unsigned int md_index);

Description:

This function will switch the current connection protocol to the specific MODEM.

Return Value:

Table 6-119 The return value of META_SwitchCurrentModem

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-120 The parameter of META_SwitchCurrentModem

Parameter	IN/OUT	Description
md_index	IN	The index of MODEM handle

6.5.14 META_SwitchCurrentModemEx

Definition:

META_RESULT __stdcall META_SwitchCurrentModemEx(const unsigned int ms_timeout, const unsigned int md_index, const unsigned int protocol, const unsigned int channel_type, const META_MODE_TRACE_PARA_T* trace_para);

META_RESULT __stdcall META_SwitchCurrentModemEx_r(const int meta_handle, const unsigned int ms_timeout, const unsigned int md_index, const unsigned int protocol, const unsigned int channel_type, const META_MODE_TRACE_PARA_T* trace_para);

Description:

This function will switch the current connection protocol to the specific MODEM with given protocol and channel type information.

Return Value:

Table 6-121 The return value of META_SwitchCurrentModemEx

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-122 The parameter of META_SwitchCurrentModemEx

Parameter	IN/OUT	Description
md_index	IN	The index of MODEM handle
protocol	IN	The protocol used on this channel 0/1: TST; 2: DHL
channel_type	IN	The type of the communication channel 0/1: Native 2: tunneling 3: tunneling without checksum

6.6 Exported Functions for RF Testing

6.6.1 META_Rf_PM

Definition:

```
        META_RESULT  __stdcall META_Rf_PM(
                                        const RfPm_Req *req,
                                        const META_RF_PM_CNF cb,
                                        short *token, void *usrData)
typedef short ARFCN;
typedef short Gain;
typedef struct
{
            ARFCN           arfcn;                  // Absolute radio frequency channel number
            char            sampleNoPerFrame;       // number of samples per frame
            Gain            gain;                   // Gain that should be used in power management
            short           frames;                 // number of frames
} RfPm_Req;

typedef struct
{
            int             power;                  // average power
            int             deviation;              // deviation of power measurement
            Gain            usedGain;               // Gain that is used
            unsigned char   ok;                     // status
} RfPm_Cnf;
```

Description:

Commands mobile station to do power measurement.

Callback:

```
        typedef void (__stdcall *META_RF_PM_CNF)(const RfPm_Cnf *cnf, const short token, void *usrData);
```

Return Value:

Table 6-123 The return value of META_Rf_PM

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-124 The parameter of META_Rf_PM

Parameter	IN/OUT	Description
req	IN	Testing command.
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.6.2 META_Rf_AFC

Definition:

```
    META_RESULT  __stdcall META_Rf_AFC(
                                const RfAfc_Req *req,
                                const META_RF_AFC_CNF cb,
                                short *token, void *usrData)
typedef struct
{
        ARFCN        arfcn;          // absolute radio frequency channel number
        short        dacValue;       // AFC DAC value
        Gain         gain;           // gain used for AFC testing
        short        testNumber;     // test number
} RfAfc_Req;

typedef struct
{
    short           fcb_ok_number;      // successful number of FCB decoded
    int             freqOffset;         // average frequency error
    int             deviation;          // deviation of frequency error
    unsigned char   ok;                 // status
} RfAfc_Cnf;
```

Description:

Commands mobile station to do AFC testing.

Callback:

```
typedef void (__stdcall *META_RF_AFC_CNF)(const RfAfc_Cnf *cnf, const short token, void *usrData);
```

Return Value:

Table 6-125 The return value of META_Rf_AFC

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-126 The parameter of META_Rf_AFC

Parameter	IN/OUT	Description
req	IN	Testing command.
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.6.3 META_Rf_NB_TX

Definition:

```
META_RESULT __stdcall META_Rf_NB_TX(
                        const RfNbtx_Req *req,
                        const META_RF_NB_TX_CNF cb,
                        short *token, void *usrData)
```

```
typedef char BSIC;
typedef short Power;

typedef enum
{
        AB_TX_RANDOM_WITH_SYNC_SEQ,
        NB_TX_ALL_ZEROS_WITHOUT_TSC,
        NB_TX_ALL_ONES_WITHOUT_TSC,
        NB_TX_ALTER_BITS_WITHOUT_TSC,
        NB_TX_RANDOM_WITH_TSC
} APCTxPattern;

typedef struct
{
        ARFCN           arfcn;          // Absolute radio frequency channel number
        BSIC            bsic;           // bsic value used in transmission
        Power           power;          // Tx power in the unit of PCL
        short           frames;         // the number of frames NB should transmit
        short           dacValue;       // AFC DAC value
        APCTxPattern    burstTypeNB;
} RfNbtx_Req;
```

Description:

Commands mobile station to transmit normal burst.

Callback:

typedef void (__stdcall *META_RF_NB_TX_CNF)(const unsigned char cnf, const short token, void *usrData);

Note:

This function will send RF_TEST_CMD_NB_TX command, which is an actual structure in C language, to target. In this command, there is a field whose name is bitmask. The parameters of this function do not contain any value for this field. The implementation of this function will automatically fill this field in the command, and the value is now always 0x01. For users of this function, they don't have any information about this field, and they don't have to care about the value of this field now.

Return Value:

Table 6-127 The return value of META_Rf_NB_TX

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-128 The parameter of META_Rf_NB_TX

Parameter	IN/OUT	Description
req	IN	Testing command.
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.6.4 META_Rf_CONTINUE_RX

Definition:

```
        META_RESULT  __stdcall META_Rf_CONTINUE_RX(
                                        const RfCnRx_Req *req,
                                        const META_RF_CONT_RX_CNF cb,
                                        short *token, void *usrData)
typedef struct
{
            ARFCN            arfcn;                 // Absolute radio frequency channel number
            Gain             gain;                  // Gain that should be used
            unsigned char    OnOff;                 // On or off
} RfCnRx_Req;
```

Description:

Commands the mobile station to toggle radio receive operation, which is used to test RF.

Callback:

typedef void (__stdcall *META_RF_CONT_RX_CNF)(const unsigned char cnf, const short token, void *usrData);

Return Value:

Table 6-129 The return value of META_Rf_CONTINUE_RX

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-130 The parameter of META_Rf_CONTINUE_RX

Parameter	IN/OUT	Description
req	IN	Testing command.
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.6.5 META_Rf_CONTINUE_TX

Definition:

```
        META_RESULT  __stdcall META_Rf_CONTINUE_TX(
                                        const RfCnTx_Req *req,
                                        const META_RF_CONT_TX_CNF cb,
                                        short *token, void *usrData)
typedef enum
{
            CONT_TX_ALL_ZEROS,
            CONT_TX_ALL_ONES,
            CONT_TX_ALTERNATE_BITS,
            CONT_TX_PSEUDO_RANDOM
} ContTxPattern;

typedef struct
{
            ARFCN           arfcn;          // Absolute radio frequency channel number
            ContTxPattern   pattern;
            unsigned char   OnOff;          // On or off
} RfCnTx_Req;
```

Description:

Commands mobile station to toggle transmission operation, except for PA module.

Callback:

typedef void (__stdcall *META_RF_CONT_TX_CNF)(const unsigned char cnf, const short token, void *usrData);

Return Value:

Table 6-131 The return value of META_Rf_CONTINUE_TX

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-132 The parameter of META_Rf_CONTINUE_TX

Parameter	IN/OUT	Description
req	IN	Testing command.
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.6.6 META_Rf_SetBBTXCfg

Definition:

```
META_RESULT __stdcall META_Rf_SetBBTXCfg(
                const RfSetBBTXCfg_Req *req,
                const META_RF_SETBBTX_CFG_CNF cb,
                short *token, void *usrData)
typedef strut
{
    char    TxTrimI;
    char    TxTrimQ;
    char    TxOffsetI;
```

char TxOffsetQ;

} RfSetBBTXCfg_Req;

Description:

Commands mobile station to set TX trim I/Q and Offset I/Q.

Callback:

typedef void (__stdcall *META_RF_SETBBTX_CFG_CNF)(const unsigned char cnf, const short token, void *usrData);

Return Value:

Table 6-133 The return value of META_Rf_SetBBTXCfg

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-134 The parameter of META_Rf_SetBBTXCfg

Parameter	IN/OUT	Description
Req	IN	Testing command.
Cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
Token	IN/OUT	Token used by user to uninstall the callback function.
UsrData	IN	Parameter used by user.

6.6.7 META_Rf_SelectFrequencyBand1900

Definition:

```
META_RESULT __stdcall META_Rf_SelectFrequencyBand1900(
                        const unsigned char selectPCS1900,
                        const META_RF_SELBAND_CNF cb,
                        short *token, void *usrData)
```

Description:

Commands mobile station to select band between PCS1900 and DCS1800.

Callback:

typedef void (__stdcall *META_RF_SELBAND_CNF)(const unsigned char cnf, const short token, void *usrData);

Return Value:

Table 6-135 The return value of META_Rf_SelectFrequencyBand1900

Return value	Description
META_SUCCESS	Success
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-136 The parameter of META_Rf_SelectFrequencyBand1900

Parameter	IN/OUT	Description
selectPCS1900	IN	1 → select PCS1900 0 → select DCS1800
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
UsrData	IN	Parameter used by user.

6.6.8 META_Rf_Stop

Definition:

META_RESULT __stdcall META_Rf_Stop(const META_RF_STOP_CNF cb, short *token, void *usrData)

Description:

Command mobile station to cease all running tests about Rf.

Callback:

typedef void (__stdcall *META_RF_STOP_CNF)(const unsigned char cnf, const short token, void *usrData);

Return Value:

Table 6-137 The return value of META_Rf_Stop

Return value	Description
META_SUCCESS	Success
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-138 The parameter of META_Rf_Stop

Parameter	IN/OUT	Description
Cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
Token	IN/OUT	Token used by user to uninstall the callback function.
UsrData	IN	Parameter used by user.

6.6.9 META_Rf_MultiSlot_TX

Definition:

```
    META_RESULT __stdcall META_Rf_MultiSlot_TX(
                              const RfMultiSlotTX_Req *req,
                              const META_RF_MULTISLOT_TX_CNF  cb,
                              short *token, void *usrData)


typedef unsigned char       TimingAdvance;


typedef enum {
        CodingSchemeCS1 = 1,
        CodingSchemeCS2,
        CodingSchemeCS3,
        CodingSchemeCS4,
        CodingSchemePRACh8,
        CodingSchemePRACh11,
        CodingSchemeMCS1,
        CodingSchemeMCS2,
        CodingSchemeMCS3,
        CodingSchemeMCS4,
        CodingSchemeMCS5,
        CodingSchemeMCS6,
        CodingSchemeMCS7,
        CodingSchemeMCS8,
        CodingSchemeMCS9
} CodingScheme;

typedef struct {
        ARFCN           arfcn;          // absolute radio frequency channel number
        BSIC            bsic;           // training sequence
        char            timeSlotmask;   // time slot mask, slot_1: 0x01, slot_2: 0x02, slot_3: 0x04, slot_4:
0x08
        Power           powerLev[4];    // power level for each time slot
        CodingScheme    cs[4];          // coding scheme for each time slot
        TimingAdvance   ta;             // time advance
        int             frames;         // the number of frames should transmit
        short           dacValue;       // AFC DAC value
} RfMultiSlotTX_Req;
```

Description:

Commands mobile station to transmit multi-slot burst.

Callback:

typedef void (__stdcall *META_RF_MULTISLOT_TX_CNF)(const unsigned char cnf, const short token, void *usrData);

Return Value:

Table 6-139 The return value of META_Rf_MultiSlot_TX

Return value	Description
META_SUCCESS	Success
META_NO_MEMORY	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-140 The parameter of META_Rf_MultiSlot_TX

Parameter	IN/OUT	Description
Req	IN	Testing command.
Cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
Token	IN/OUT	Token used by user to uninstall the callback function.
UsrData	IN	Parameter used by user.

6.6.10 META_Rf_SetRampApcLevel

Definition:

```
        META_RESULT  __stdcall META_Rf_SetRampApcLevel(
                                const RfSetRampApcLevel_Req *req,
                                const META_RF_SET_RAMPAPCLEVEL_CNF  cb,
                                short *token, void *usrData)


typedef struct {
         FrequencyBand    rf_band;
         int              power_level;
         int              apc_dac;
} RfSetRampApcLevel_Req;
```

Description:

Directly change power level without update calibration data.

Callback:

typedef void (__stdcall *META_RF_SET_RAMPAPCLEVEL_CNF)(const unsigned char cnf, const short token, void *usrData);

Return Value:

Table 6-141 The return value of META_Rf_SetRampApcLevel

Return value	Description
META_SUCCESS	Success
META_NO_MEMORY	Memory is not enough.

Return value	Description
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-142 The parameter of META_Rf_SetRampApcLevel

Parameter	IN/OUT	Description
req	IN	Testing command.
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.6.11 META_Rf_EPSK_SetRampApcLevel

Definition:

```
        META_RESULT  __stdcall META_Rf_EPSK_SetRampApcLevel(
                               unsigned int ms_timeout, const RfSetRampApcLevel_Req *req)


typedef struct {
        FrequencyBand    rf_band;
        int              power_level;
        int              apc_dac;
} RfSetRampApcLevel_Req;
```

Description:

Directly change power level without update calibration data in EDGE.

Return Value:

Table 6-143 The return value of META_Rf_EPSK_SetRampApcLevel

Return value	Description
META_SUCCESS	Success
META_NO_MEMORY	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-144 The parameter of META_Rf_EPSK_SetRampApcLevel

Parameter	IN/OUT	Description
req	IN	Testing command.
ms_timeout	IN	Function timeout value. (in milliseconds)

6.6.12 META_Rf_SetAfcDacValue

Definition:

```
        META_RESULT  __stdcall META_Rf_SetAfcDacValue(
                                const RfSetAfcDacValue_Req *req,
                                const META_RF_SET_AFCDACVALUE_CNF  cb,
                                short *token, void *usrData)


typedef struct {
        short               dacValue;       // AFC DAC value
} RfSetAfcDacValue_Req;
```

Description:

Update AFC DAC value.

Callback:

typedef void (__stdcall * META_RF_SET_AFCDACVALUE_CNF)(const unsigned char cnf, const short token, void *usrData);

Return Value:

Table 6-145 The return value of META_Rf_SetAfcDacValue

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-146 The parameter of META_Rf_SetAfcDacValue

Parameter	IN/OUT	Description
req->dacValue	IN	AFC DAC value.
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.6.13 META_Rf_SetBBTxCfg2

Definition:

```
        META_RESULT  __stdcall META_Rf_SetBBTxCfg2(
                                    unsigned int ms_timeout,
                                    const RfBBTXCfg2  *tx_cfg_req,
                                    RfBBTXCfg2  *tx_cfg_cnf)


typedef struct {
        char        TxTrimI;
        char        TxTrimQ;
        char        TxOffsetI;
        char        TxOffsetQ;
        char        TxCalbias;
        char        TxIQSwap;
        char        TxCMV;
        char        TxGain;
        char        TxCalrcsel;
} RfBBTXCfg2;
```

Description:

Set baseband TX config.

Return Value:

Table 6-147 The return value of META_Rf_SetBBTxCfg2

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-148 The parameter of META_Rf_SetBBTxCfg2

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
tx_cfg_req	IN	TX config.
tx_cfg_cnf	IN/OUT	Read back TX config for your confirmation. If you don’t want to confirm, just assign NULL.

6.6.14 META_Rf_GetBBTxCfg2

Definition:

```
META_RESULT  __stdcall META_Rf_SetBBTxCfg2(
                    unsigned int ms_timeout,
                    RfBBTXCfg2  *tx_cfg_cnf)
```

Description:

Get current baseband TX config.

Return Value:

Table 6-149 The return value of META_Rf_GetBBTxCfg2

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-150 The parameter of META_Rf_GetBBTxCfg2

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
tx_cfg_cnf	IN/OUT	Current baseband TX config.

6.6.15 META_Rf_BBTXAutoCal

Definition:

```
META_RESULT __stdcall META_Rf_BBTXAutoCal(unsigned int ms_timeout);
```

Description:

Trigger target L! module perform baseband TX auto-calibration.

Return Value:

Table 6-151 The return value of META_Rf_BBTXAutoCal

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-152 The parameter of META_Rf_BBTXAutoCal

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

6.6.16 META_Rf_QueryMSCapability

Definition:

```
META_RESULT __stdcall META_Rf_QueryMSCapability(
                                        unsigned int ms_timeout,
                                        RfMsCapability_S  *p_type)
typedef enum {
  MS_GSM = 0
  ,MS_GPRS
  ,MS_EGPRS_RX_ONLY
  ,MS_EGPRS_FULL_FUNCTION
} MS_CAPABILITY_E;
```

```
typedef struct {
  unsigned int        GSM400;          // Zero: not support, Non-zero: support
  unsigned int        GSM850;
  unsigned int        GSM900;
  unsigned int        DCS1800;
  unsigned int        PCS1900;
} RFBandSupport_S;

typedef struct {
  MS_CAPABILITY_E                      capability;
  RFBandSupport_S                      band_support;
} RfMsCapability_S;
```

Description:

Query mobile station capability of target.

Return Value:

Table 6-153 The return value of META_Rf_QueryMSCapability

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-154 The parameter of META_Rf_QueryMSCapability

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
p_type	IN/OUT	Return value of RfMsCapability_S structure.

6.6.17 META_Rf_SetAfcSinWaveDetection

Definition:

```
META_RESULT __stdcall META_Rf_SetAfcSinWaveDetection(
                                    unsigned int ms_timeout,
                                    AFC_SINWAVE_DETECTION_E  bIsAfcSinWaveOn)
        typedef enum {
          AFC_SINWAVE_OFF = 0
          ,AFC_SINWAVE_ON
        } AFC_SINWAVE_DETECTION_E;
```

Description:

Configure L1 to use sin wave input for AFC detection.

Return Value:

Table 6-155 The return value of META_Rf_SetAfcSinWaveDetection

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-156 The parameter of META_Rf_SetAfcSinWaveDetection

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
bIsAfcSinWaveOn	IN	Return value of RfMsCapability_S structure.

6.6.18 META_Rf_QueryIfTwoApcDCOffsetSupport

Definition:

```
META_RESULT  __stdcall META_Rf_QueryIfTwoApcDCOffsetSupport(unsigned int ms_timeout)
```

Description:

Query if target supported two APC DC offset configuration.

Return Value:

Table 6-157 The return value of META_Rf_QueryIfTwoApcDCOffsetSupport

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-158 The parameter of META_Rf_QueryIfTwoApcDCOffsetSupport

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

6.6.19 META_Rf_SetRampTable

Definition:

```
META_RESULT  __stdcall META_Rf_SetRampTable(
                    unsigned int ms_timeout,
                    FrequencyBand  band,
                    const l1cal_rampTable_T  *ramp)
```

Description:

Directly change ramp table setting without updating NVRAM.

Return Value:

Table 6-159 The return value of META_Rf_SetRampTable

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-160 The parameter of META_Rf_SetRampTable

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
band	IN	Selecting which band to be updated.
ramp	IN	Ramp table setting.

6.6.20 META_Rf_SetBBTxCfg4

Definition:

META_RESULT __stdcall META_Rf_SetBBTxCfg4(unsigned int ms_timeout, const RfBBTXCfg4 *tx_cfg_req, RfBBTXCfg4 *tx_cfg_cnf);

META_RESULT __stdcall META_Rf_SetBBTxCfg4_r(const int meta_handle, unsigned int ms_timeout, const RfBBTXCfg4 *tx_cfg_req, RfBBTXCfg4 *tx_cfg_cnf);

```
typedef struct {
        char    TxTrimI;
        char    TxTrimQ;
        char    TxOffsetI;
        char    TxOffsetQ;
        char    TxCalbias;
        char    TxIQSwap;
        char    TxCMV;
        char    TxGain;
        char    TxCalrcsel;
        char    TxPhasesel;
        char    TxCoarseI;
        char    TxCoarseQ;
}RfBBTXCfg4;
```

Description:

Set baseband TX config4.

Return Value:

Table 6-161 The return value of META_Rf_SetBBTxCfg4

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-162 The parameter of META_Rf_SetBBTxCfg4

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
tx_cfg_req	IN	TX config4.
tx_cfg_cnf	IN/OUT	Read back TX config4 for your confirmation. If you don't want to confirm, just assign NULL.

6.6.21 META_Rf_GetBBTxCfg4

Definition:

META_RESULT __stdcall META_Rf_GetBBTxCfg4(unsigned int ms_timeout, RfBBTXCfg4 *tx_cfg_cnf);

META_RESULT __stdcall META_Rf_GetBBTxCfg4_r(const int meta_handle, unsigned int ms_timeout, RfBBTXCfg4 *tx_cfg_cnf);

```
typedef struct {
        char    TxTrimI;
        char    TxTrimQ;
        char    TxOffsetI;
        char    TxOffsetQ;
        char    TxCalbias;
        char    TxIQSwap;
        char    TxCMV;
        char    TxGain;
        char    TxCalrcsel;
        char    TxPhasesel;
        char    TxCoarseI;
        char    TxCoarseQ;
}RfBBTXCfg4;
```

Description:

Get current baseband TX config4.

Return Value:

Table 6-163 The return value of META_Rf_GetBBTxCfg4

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-164 The parameter of META_Rf_GetBBTxCfg4

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
tx_cfg_cnf	IN/OUT	Current baseband TX config4.

6.6.22 META_Rf_SetBBTxCfg5

Definition:

META_RESULT __stdcall META_Rf_SetBBTxCfg5(unsigned int ms_timeout, const RfBBTXCfg4 *tx_cfg_req, RfBBTXCfg4 *tx_cfg_cnf);
META_RESULT __stdcall META_Rf_SetBBTxCfg5_r(const int meta_handle, unsigned int ms_timeout, const RfBBTXCfg4 *tx_cfg_req, RfBBTXCfg4 *tx_cfg_cnf);

```
typedef struct {
        char   TxTrimI;
        char   TxTrimQ;
        char   TxOffsetI;
        char   TxOffsetQ;
        char   TxCalbias;
        char   TxIQSwap;
        char   TxCMV;
        char   TxGain;
        char   TxCalrcsel;
        char   TxPhasesel;
        char    TxCoarseI;
        char    TxCoarseQ;
}RfBBTXCfg4;
```

Description:

Set baseband TX config5.

Return Value:

Table 6-165 The return value of META_Rf_SetBBTxCfg5

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-166 The parameter of META_Rf_SetBBTxCfg5

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
tx_cfg_req	IN	TX config5.
tx_cfg_cnf	IN/OUT	Read back TX config5 for your confirmation. If you don't want to confirm, just assign NULL.

6.6.23 META_Rf_GetBBTxCfg5

Definition:

META_RESULT __stdcall META_Rf_GetBBTxCfg5(unsigned int ms_timeout, RfBBTXCfg4 *tx_cfg_cnf);

META_RESULT __stdcall META_Rf_GetBBTxCfg5_r(const int meta_handle, unsigned int ms_timeout, RfBBTXCfg4 *tx_cfg_cnf);

```
typedef struct {
        char    TxTrimI;
        char    TxTrimQ;
        char    TxOffsetI;
        char    TxOffsetQ;
        char    TxCalbias;
        char    TxIQSwap;
        char    TxCMV;
        char    TxGain;
        char    TxCalrcsel;
        char    TxPhasesel;
        char    TxCoarseI;
        char    TxCoarseQ;
}RfBBTXCfg4;
```

Description:

Get current baseband TX config5.

Return Value:

Table 6-167 The return value of META_Rf_GetBBTxCfg5

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-168 The parameter of META_Rf_GetBBTxCfg5

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
tx_cfg_cnf	IN/OUT	Current baseband TX config5.

6.6.24 META_Rf_32kCalibration

Definition:

META_RESULT __stdcall META_Rf_32kCalibration(unsigned int ms_timeout, int *p_result);

META_RESULT __stdcall META_Rf_32kCalibration_r(const int meta_handle, unsigned int ms_timeout, int *p_result);

Description:

Ask target to do 32k clock calibration, and return the result.

Return Value:

Table 6-169 The return value of META_Rf_32kCalibration

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-170 The parameter of META_Rf_32kCalibration

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
p_result	IN/OUT	Int address of 32k clock calibration result pointer.

6.6.25 META_Rf_AD6546_SetSpecialCoef

Definition:

META_RESULT __stdcall META_Rf_AD6546_SetSpecialCoef(unsigned int ms_timeout, const ad6546tx *rf_mod_coef, const char *buf, const int buf_len);
META_RESULT __stdcall META_Rf_AD6546_SetSpecialCoef_r(const int meta_handle, unsigned int ms_timeout, const ad6546tx *rf_mod_coef, const char *buf, const int buf_len)

```
typedef struct
{
  unsigned char REFDET_SLOPE_SKEW ;
  unsigned char AM_FB_DAC;
}ad6546txcoef;

typedef  struct
{
    ad6546txcoef   CalData[4];
} ad6546tx;
```

Description:

Ask target to do runtime settings of RF special coefficients. Will call META_NVRAM_Compose_ad6546tx( ) to fill the rf_mod_coef to buf, then send the content to target side.

Return Value:

Table 6-171 The return value of META_Rf_AD6546_SetSpecialCoef

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-172 The parameter of META_Rf_AD6546_SetSpecialCoef

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
rf_mod_coef	IN	RF special coefficient we want to store in target
buf	IN	The buffer stores the original target entire settings
Buf_len	IN	Buffer length

6.6.26 META_Rf_StartFdtDL

Definition:

META_RESULT __stdcall META_Rf_StartFdtDL(unsigned int ms_timeout, const Rf_DTS_REQ_T *fdt_dl_req, Rf_DTS_CNF_T *fdt_dl_cnf);

META_RESULT __stdcall META_Rf_StartFdtDL_r(const int meta_handle, unsigned int ms_timeout, const Rf_DTS_REQ_T *fdt_dl_req, Rf_DTS_CNF_T *fdt_dl_cnf);

```
#define MAX_STEP_CNT   50

typedef struct
{
  bool                afc_cal;
  bool                    pl_cal;      // Control whether Path loss calibration is needed or not
  char                      sync_sb_num;
                                                 // the SB frame numbers needed for sync process before path loss calibration
  short                     power;           // the power level expected to measure from test set
  Rf_DSSAFC_T      AfcDSS;
  char                       step_cnt;
  Rf_DSSPL_T           PathLossDSS[MAX_STEP_CNT-2];  // because sync process will need 2 steps

}Rf_DTS_REQ_T;

typedef struct{
  FrequencyBand      band;
  ARFCN      arfcn;
  short                     dac_value[33];
  Gain                      gain;
  short                     repeat_cnt;    // repetitive test counts (frames) for each AFC DAC value
  bool                      capid_cal;       // capid calibration ctrl
  bool                      linear_cal;      // 33 stages calibration ctrl
  int                        capid_min;      // min value for capid range when capid_cal is True; capid when capid_cal is
False
  int                        capid_max;     // max value for capid range

}Rf_DSSAFC_T;

typedef struct
{
  FrequencyBand              band;
  ARFCN              arfcn;
  Gain                              gain[6];               // gain for rx slot 0/1/2/3/4/5
  short                         repeat_cnt;         // repetitive test counts (frames) for each ARFCN value
} Rf_DSSPL_T;


typedef struct
{
  int              power[MAX_STEP_CNT-2];  // because sync process will need 2 steps
  short            valid_sample[MAX_STEP_CNT-2];
  bool              ok;
} Rf_DSSPL_RESULT_T;


typedef struct
{
  int        freq_offset[33];                  // only valid when 33 stage calibration is ON
  int        deviation[33];          // only valid when 33 stage calibration is ON
  short      fcb_ok_number[33];  // only valid when 33 stage calibration is ON
  int        capid;                       // only valid when capid calibration is ON
  short      init_dac_value;         // only valid when 33 stage calibration is OFF
  int        slope;                       // only valid when 33 stage calibration is OFF
  bool       ok;
} Rf_DSSAFC_RESULT_T;


#define FHC_PRE_CAPID_SEARCH_NUM   9
#define FHC_MAX_CAPID_SEARCH_NUM   (7 + FHC_PRE_CAPID_SEARCH_NUM)


typedef struct
{
  int          path_loss_cnt;
  int          freq_offset;
  int          capid_freq_offset_min;
```

```
    int          capid_freq_offset[FHC_MAX_CAPID_SEARCH_NUM];
    int          capid_search_order[FHC_MAX_CAPID_SEARCH_NUM];
    int          capid;
    int          capid_high;
    int          capid_low;
    int          capid_best;
    short         afc_dac;
    short         arfcn;
    short         capid_cnt;
    short         repeat_index;
    char         state;
    char         capid_index;
    char         capid_okay_cnt;
    char         afc_dac_index;
    char         sb_okay_cnt;
    unsigned char    sb_fail_cnt;
    unsigned char    fb_fail_cnt;
    bool         pl_started;
    bool         pre_capid_cal_ok[FHC_PRE_CAPID_SEARCH_NUM];

}Rf_FHC_DTSM_INFO_T;

typedef enum  {
        DTS_RESULT_READY = 0,          // DTS results is ready to get back
        DTS_RESULT_NOT_READY,          // DTS result is still in progress and not ready to get back
        DTS_RESULT_NOT_REQUESTED       // Haven't called the META_Rf_StartFdtDL() in advance.
        DTS_FATAL_ERROR                   // Unexpected behavior happen.
}RF_DTS_GET_RESULT_STATUS;
```

```
typedef struct
{
  RF_DTS_GET_RESULT_STATUS      status;
  Rf_DSSPL_RESULT_T                         PLResult;
  Rf_DSSAFC_RESULT_T            AfcResult;
  Rf_FHC_DTSM_INFO_T                        DtsmInfo;
} Rf_DTS_CNF_T;
```

Description:

Fast Device Tuning (FDT) Downlink calibration (AFC and RX path loss) in a synchronous way. Therefore, it will wait for result back from the target. If the asynchronous way

Return Value:

Table 6-173 The return value of META_Rf_StartFdtDL

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.
	[Error] RCT common initialize failed - Please check environment setting, (ex.cfg file, instrument setting) - Maybe GPIB is not work normally, please check NI tool to check whether if PC can detect GPIB
	[GPIB] Please lock the waveform xxxx.wfm' - it means the waveform of instructment become overdue, please lock waveform on instructment

Parameter:

Table 6-174 The parameter of META_Rf_StartFdtDL

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
fdt_dl_req	IN	Downlink calibration parameter
fdt_dl_cnf	IN/OUT	Downlink calibration result

6.6.27 META_Rf_StartFdtDLNotWaitResult

Definition:

META_RESULT __stdcall META_Rf_StartFdtDLNotWaitResult (unsigned int ms_timeout, const Rf_DTS_REQ_T *fdt_dl_req);

META_RESULT __stdcall META_Rf_StartFdtDLNotWaitResult _r(const int meta_handle, unsigned int ms_timeout, const Rf_DTS_REQ_T *fdt_dl_req);

Description:

Fast Device Tuning (FDT) Downlink calibration (AFC and RX path loss) in an asynchronous way. Therefore, it won't wait for result and should use META_Rf_GetFdtDL() to query the result.

Return Value:

Table 6-175 The return value of META_Rf_StartFdtDLNotWaitResult

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.
	[Error] RCT common initialize failed - Please check environment setting, (ex.cfg file, instrument setting) - Maybe GPIB is not work normally, please check NI tool to check whether if PC can detect GPIB
	[GPIB] Please lock the waveform xxxx.wfm' - it means the waveform of instructment become overdue, please lock waveform on instructment

Parameter:

Table 6-176 The parameter of META_Rf_StartFdtDLNotWaitResult

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
fdt_dl_req	IN	Downlink calibration parameter

6.6.28 META_Rf_GetFdtDL

Definition:

META_RESULT __stdcall META_Rf_GetFdtDL (unsigned int ms_timeout, Rf_DTS_CNF_T *fdt_dl_get_result_cnf);
META_RESULT __stdcall META_Rf_GetFdtDL_r (const int meta_handle, unsigned int ms_timeout, Rf_DTS_CNF_T *fdt_dl_get_result_cnf);

Description:

This is a query function to get the Fast Device Tuning (FDT) Downlink calibration (AFC and RX path loss) in an asynchronous way.

Return Value:

Table 6-177 The return value of META_Rf_GetFdtDL

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.
	[*Error*] FAIL: RX Path Loss Calibration failed DTSMInfo:sb_fail_cnt = 255 - Sync burst can not found, please provide ELT L1 Log for us to analysis
	[*Error*] FAIL: RX Path Loss Calibration failed DTSMInfo:fb_fail_cnt = 10 - Frequency burst can not found, please provide ELT L1 Log for us to analysis
	[*Error*] META_Rf_GetFdtDL_r time out! - Maybe CMD sequence is not right or L1 process error, please provide ELT L1 Log for us to analysis

Parameter:

Table 6-178 The parameter of META_Rf_GetFdtDL

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
fdt_dl_get_result_cnf	IN/OUT	Downlink calibration result
fdt_dl_get_result_cnf.status	OUT	0: DTS results is ready to get back 1: DTS result is still in progress and not ready to get back 2: Haven't called the META_Rf_StartFdtDL() in advance. 3: Unexpected behavior happen.

6.6.29 META_Rf_StartFdtUL

Definition:

META_RESULT __stdcall META_Rf_StartFdtUL(unsigned int ms_timeout, const Rf_UTS_REQ_T *fdt_ul_req);

META_RESULT __stdcall META_Rf_StartFdtUL_r(const int meta_handle, unsigned int ms_timeout, const Rf_UTS_REQ_T *fdt_ul_req);

```
#define MAX_STEP_CNT   50
typedef struct
{
  char                    step_cnt;
  short                 high_apc_dcoffset[FrequencyBandCount];
  Rf_USSAPC_T           ApcUSS[MAX_STEP_CNT];
}Rf_UTS_REQ_T;

typedef struct
{
  FrequencyBand band;
  ARFCN             arfcn;
  char                          timeslot_per_frame;
  char                          apc_dac_pcl_sel; // 1: apc_dac, 0: apc_pcl
  short                         apc_dac_pcl_value[4];
   unsigned char              pa_vbias_val[4];
  unsigned char               is_low_pcl[4];
  CodingScheme                 cs[4];
  int                           repeat_cnt;
  short                        afc_dac_value;
```

```
    char                    tsc;
    APCTxPattern            pattern;
    unsigned short          pattern_data;
} Rf_USSAPC_T;
```

Description:

Fast Device Tuning (FDT) Uplink calibration (APC calibration).

Return Value:

Table 6-179 The return value of META_Rf_StartFdtUL

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.
	[Error] RCT common initialize failed - Please check environment setting, (ex.cfg file, instrument setting) - Maybe GPIB is not work normally, please check NI tool to check whether if PC can detect GPIB
	[GPIB] Please lock the waveform xxxx.wfm' - it means the waveform of instructment become overdue, please lock waveform on instructment

Parameter:

Table 6-180 The parameter of META_Rf_StartFdtUL

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
fdt_ul_req	IN	Uplink calibration parameter

6.6.30 META_Rf_QueryMSCapabilityEx2

Definition:

META_RESULT __stdcall META_Rf_QueryMSCapabilityEx2(unsigned int ms_timeout, RfMsCapabilityEx2_S *p_ms_cap);

```
META_RESULT  __stdcall META_Rf_QueryMSCapabilityEx2_r(const int meta_handle, unsigned int ms_timeout,
RfMsCapabilityEx2_S  *p_ms_cap);

typedef struct {
        unsigned int       GSM:1;
        unsigned int       GPRS:1;
        unsigned int       EDGE_RX:1;
        unsigned int       EDGE_8PSK_TX:1;
        unsigned int       Calibration_8PM:1;
        unsigned int       Calibration_FDT:1;     // new
        unsigned int       Calibration_33Steps:1;  // new
} RfMsCapabilityBits_2;


typedef struct {
        RfMsCapabilityBits_2                capability;
        RfMsBandSupportBits                 band_support;
} RfMsCapabilityEx2_S;
```

Description:

An ehanced function to query target RF capability such as FDT and 33 steps capability

Return Value:

Table 6-181 The return value of META_Rf_QueryMSCapabilityEx2

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-182 The parameter of META_Rf_QueryMSCapabilityEx2

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
p_ms_cap	IN/OUT	The result of target RF capability.

6.6.31 META_Rf_GetAFCDacTRxOffset

Definition:

META_RESULT __stdcall META_Rf_GetAFCDacTRxOffset(unsigned int ms_timeout, RF_GET_AFC_DAC_OFFSET_CNF_T *cnf);

META_RESULT __stdcall META_Rf_GetAFCDacTRxOffset_r(const int meta_handle, unsigned int ms_timeout,

RF_GET_AFC_DAC_OFFSET_CNF_T *cnf);

```
typedef struct
{
    short afc_offset[FrequencyBandCount];

}RF_GET_AFC_DAC_OFFSET_CNF_T;

typedef enum
{
   FrequencyBand400=0,                  // GSM 450/480 band
   FrequencyBand850,                    // GSM 850 band
   FrequencyBand900,                    // GSM 900 band
   FrequencyBand1800,                   // DCS 1800 band
   FrequencyBand1900,                   // PCS 1900 band
   FrequencyBandCount                   // count of supported bands
} FrequencyBand;
```

Description:

Query AFC DAC offset of all bands.

Return Value:

Table 6-183 The return value of META_Rf_GetAFCDacTRxOffset

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-184 The parameter of META_Rf_GetAFCDacTRxOffset

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
cnf	IN/OUT	AFC DAC Offset array.

6.6.32 META_Rf_SetAFCDacTRxOffset

Definition:

META_RESULT __stdcall META_Rf_SetAFCDacTRxOffset(unsigned int ms_timeout, const RF_SET_AFC_DAC_OFFSET_REQ_T *req);

META_RESULT __stdcall META_Rf_SetAFCDacTRxOffset_r(const int meta_handle, unsigned int ms_timeout, const RF_SET_AFC_DAC_OFFSET_REQ_T *req);

```
typedef struct
{
    short afc_offset[FrequencyBandCount];

}RF_SET_AFC_DAC_OFFSET_REQ_T;



typedef enum
{
  FrequencyBand400=0,                    // GSM 450/480 band
  FrequencyBand850,                      // GSM 850 band
  FrequencyBand900,          // GSM 900 band
  FrequencyBand1800,         // DCS 1800 band
  FrequencyBand1900,         // PCS 1900 band
  FrequencyBandCount         // count of supported bands
} FrequencyBand;
```

Description:

Set AFC DAC offset of all bands.

Return Value:

Table 6-185 The return value of META_Rf_SetAFCDacTRxOffset

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-186 The parameter of META_Rf_SetAFCDacTRxOffset

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	AFC DAC Offset array.

6.6.33 META_Rf_EPSK_SetRampTable

Definition:

META_RESULT __stdcall META_Rf_EPSK_SetRampTable(unsigned int ms_timeout, FrequencyBand band, const l1cal_rampTable_T *ramp);

META_RESULT __stdcall META_Rf_EPSK_SetRampTable_r(const int meta_handle, unsigned int ms_timeout, FrequencyBand band, const l1cal_rampTable_T *ramp);

M

typedef enum

```
{
  FrequencyBand400=0,                     // GSM 450/480 band
  FrequencyBand850,                       // GSM 850 band
  FrequencyBand900,                       // GSM 900 band
  FrequencyBand1800,                      // DCS 1800 band
  FrequencyBand1900,                      // PCS 1900 band
  FrequencyBandCount                      // count of supported bands
} FrequencyBand;

typedef struct
{
  sRAMPDATA     rampData;                                          // apc ramp profile of
all bands
}l1cal_rampTable_T;

typedef  struct
{
  int                  lowest_power;      // The lower apc power of the indicated band
  unsigned short       power[16];    // The mapping of power level to apc dac value
  sRAMPAREADATA      ramp[ PROFILE_NUM ];             // ramp profile
  sARFCN_SECTION    arfcn_weight[ ARFCN_SECTION_NUM ];
                                           // profile of weighting power level by PCL and sub-
band
  unsigned short       battery_compensate[3][3];        // [volt][temp]
  short                  tx_afc_offset;
} sRAMPDATA;

#define PROFILE_NUM           16
#define ARFCN_SECTION_NUM       12
```

```
typedef  struct
{
   unsigned char   point[2][16];        // ramp up/down profile

} sRAMPAREADATA;

 typedef  struct
{
   short                          max_arfcn;                    // sub-band boundary of this PCL weighting area
   unsigned short   mid_level;                  // PCLboundary level to apply high/low weighting
   unsigned short   hi_weight;                  // scale factor of PCLs higher than mid_level
   unsigned short   low_weight;                 // scale factor of PCLs lower than mid_level

} sARFCN_SECTION;
```

Description:

Runtime set EPSK ramp table.

Return Value:

Table 6-187 The return value of META_Rf_EPSK_SetRampTable

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-188 The parameter of META_Rf_EPSK_SetRampTable

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
ramp	IN	EPSK ramp table structure.

6.6.34 META_Rf_SetBBTxCfg6

Definition:

META_RESULT __stdcall META_Rf_SetBBTxCfg6(unsigned int ms_timeout, const RfBBTXCfg4 *tx_cfg_req, RfBBTXCfg4 *tx_cfg_cnf);
META_RESULT __stdcall META_Rf_SetBBTxCfg6_r(const int meta_handle, unsigned int ms_timeout, const RfBBTXCfg4 *tx_cfg_req, RfBBTXCfg4 *tx_cfg_cnf);

```
typedef struct {
        char    TxTrimI;
        char    TxTrimQ;
        char    TxOffsetI;
        char    TxOffsetQ;
        char    TxCalbias;
        char    TxIQSwap;
        char    TxCMV;
        char    TxGain;
        char    TxCalrcsel;
        char    TxPhasesel;
        char    TxCoarseI;
        char    TxCoarseQ;
}RfBBTXCfg4;
```

Description:

Set baseband TX config5.

Return Value:

Table 6-189 The return value of META_Rf_SetBBTxCfg6

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-190 The parameter of META_Rf_SetBBTxCfg6

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
tx_cfg_req	IN	TX config6.
tx_cfg_cnf	IN/OUT	Read back TX config6 for your confirmation. If you don't want to confirm, just assign NULL.

6.6.35 META_Rf_GetBBTxCfg6

Definition:

META_RESULT __stdcall META_Rf_GetBBTxCfg6(unsigned int ms_timeout, RfBBTXCfg4 *tx_cfg_cnf);

META_RESULT __stdcall META_Rf_GetBBTxCfg6_r(const int meta_handle, unsigned int ms_timeout, RfBBTXCfg4 *tx_cfg_cnf);

```
typedef struct {
        char    TxTrimI;
        char    TxTrimQ;
        char    TxOffsetI;
        char    TxOffsetQ;
        char    TxCalbias;
        char    TxIQSwap;
        char    TxCMV;
        char    TxGain;
        char    TxCalrcsel;
        char    TxPhasesel;
        char    TxCoarseI;
        char    TxCoarseQ;
}RfBBTXCfg4;
```

Description:

Get current baseband TX config6.

Return Value:

Table 6-191 The return value of META_Rf_GetBBTxCfg6

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-192 The parameter of META_Rf_GetBBTxCfg6

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
tx_cfg_cnf	IN/OUT	Current baseband TX config6.

6.6.36 META_Rf_NSFT_Start

Definition:

META_RESULT __stdcall META_Rf_GetBBTxCfg6(unsigned int ms_timeout, const Rf_NSFT_REQ_T *req);

META_RESULT __stdcall META_Rf_GetBBTxCfg6_r(const int meta_handle, unsigned int ms_timeout, const Rf_NSFT_REQ_T *req);

```
typedef struct{
        FrequencyBand   band;
        ARFCN           BCH_ARFCN;
        ARFCN           TCH_ARFCN;
        Gain            BCH_gain;
        Gain            TCH_gain;
        TSC             tsc;
        TimeSlot        TCH_slot;
        Power           tx_power_level;
        bool            is_EPSK_tx;
        CodingScheme     epsk_cs;
}Rf_NSFT_REQ_T;
```

Description:

Start NSFT process with given configuration.

Return Value:

Table 6-193 The return value of META_Rf_NSFT_Start

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.
	NSFT Start Fail - please check environment setting (ex.cfg file, UI instructment setting, waveform lock) - If environment setting is right, but problem still happen, please provide us with ELT L1 log to analysis

Parameter:

Table 6-194 The parameter of META_Rf_NSFT_Start

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	NSFT start configuration.

6.6.37 META_Rf_NSFT_ChangeSettings

Definition:

META_RESULT __stdcall META_Rf_NSFT_ChangeSettings(unsigned int ms_timeout, const Rf_NSFT_REQ_T *req);

META_RESULT __stdcall META_Rf_NSFT_ChangeSettings_r(const int meta_handle, unsigned int ms_timeout, const Rf_NSFT_REQ_T *req);

```
typedef struct{
        FrequencyBand   band;
        ARFCN           BCH_ARFCN;
        ARFCN           TCH_ARFCN;
        Gain            BCH_gain;
        Gain            TCH_gain;
        TSC             tsc;
        TimeSlot        TCH_slot;
        Power           tx_power_level;
        bool            is_EPSK_tx;
       CodingScheme     epsk_cs;
}Rf_NSFT_REQ_T;
```

Description:

Change NSFT process configuration.

Return Value:

Table 6-195 The return value of META_Rf_NSFT_ChangeSettings

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-196 The parameter of META_Rf_NSFT_ChangeSettings

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	NSFT change configuration.

6.6.38 META_Rf_NSFT_ConfigSBER

Definition:

META_RESULT __stdcall META_Rf_NSFT_ConfigSBER(unsigned int ms_timeout, const unsigned int test_frame_count);

META_RESULT __stdcall META_Rf_NSFT_ConfigSBER_r(const int meta_handle, unsigned int ms_timeout, const unsigned int test_frame_count);

Description:

Configures the SBER (single-end BER) test frame count (must be 4x, eg. 20, 40…)

Return Value:

Table 6-197 The return value of META_Rf_NSFT_ConfigSBER

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-198 The parameter of META_Rf_NSFT_ConfigSBER

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
test_frame_count	IN	SBER test frame count.

6.6.39 META_Rf_NSFT_GetSBER

Definition:

META_RESULT __stdcall META_Rf_NSFT_GetSBER(unsigned int ms_timeout, RF_NSFT_SBERResult_T* sber_result);

META_RESULT __stdcall META_Rf_NSFT_GetSBER_r(const int meta_handle, unsigned int ms_timeout, RF_NSFT_SBERResult_T* sber_result);

```
typedef struct
{
  unsigned int m_u4NSFTSBERSum; // sum of the bit-error in the SBER test (m_u4NSFTSBERSum/ m_u4NSFTSBERCurrentCount/1000 = BER)
  unsigned int m_u4NSFTSBERCurrentCount; // SBER test progress (unit: frames)
}RF_NSFT_SBERResult_T;
```

Description:

Query the current SBER test result. Must be called after META_Rf_NSFT_ConfigSBER

Return Value:

Table 6-199 The return value of META_Rf_NSFT_GetSBER

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-200 The parameter of META_Rf_NSFT_GetSBER

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
test_frame_count	IN	SBER test frame count.

6.6.40 META_Rf_NSFT_StartRxLevel

Definition:

META_RESULT __stdcall META_Rf_NSFT_StartRxLevel(unsigned int ms_timeout);

META_RESULT __stdcall META_Rf_NSFT_StartRxLevel_r(const int meta_handle, unsigned int ms_timeout);

Description:

Start the RX level calculation in target side

Return Value:

Table 6-201 The return value of META_Rf_NSFT_StartRxLevel

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-202 The parameter of META_Rf_NSFT_StartRxLevel

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

6.6.41 META_Rf_NSFT_GetRxLevel

Definition:

META_RESULT __stdcall META_Rf_NSFT_GetRxLevel(unsigned int ms_timeout, unsigned short *rx_level);

META_RESULT __stdcall META_Rf_NSFT_GetRxLevel_r(const int meta_handle, unsigned int ms_timeout, unsigned short *rx_level);

Description:

Get the RX level indicator from target side

Return Value:

Table 6-203 The return value of META_Rf_NSFT_GetRxLevel

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-204 The parameter of META_Rf_NSFT_GetRxLevel

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
Rx_level	OUT	RX level indicator

6.6.42 META_Rf_NSFT_GetRxQual

Definition:

META_RESULT __stdcall META_Rf_NSFT_GetRxQual(unsigned int ms_timeout, const unsigned short ber_decile, unsigned char *rx_qual);

META_RESULT __stdcall META_Rf_NSFT_GetRxQual_r(const int meta_handle, unsigned int ms_timeout, const unsigned short ber_decile, unsigned char *rx_qual);

Description:

Get the RX quality indicator from the target side

Return Value:

Table 6-205 The return value of META_Rf_NSFT_GetRxQual

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-206 The parameter of META_Rf_NSFT_GetRxQual

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
Rx_qual	OUT	RX quality indicator

6.6.43 META_Rf_List_Mode_NSFT_Start_r

Definition:

META_RESULT __stdcall META_Rf_List_Mode_NSFT_Start(unsigned int ms_timeout, const Rf_LIST_MODE_NSFT_REQ_T* req, Rf_LIST_MODE_NSFT_RPT_CNF_T* cnf);

META_RESULT __stdcall META_Rf_List_Mode_NSFT_Start_r(const int meta_handle, unsigned int ms_timeout, const Rf_LIST_MODE_NSFT_REQ_T* req, Rf_LIST_MODE_NSFT_RPT_CNF_T* cnf);

```
typedef struct
{
    unsigned char            ucCmdCount;
    Rf_LIST_MODE_NSFT_COMMAND_T  command[RF_MAX_LIST_MODE_COMMAND_COUNT];
} Rf_LIST_MODE_NSFT_REQ_T;

typedef struct
{
    unsigned char            ucCnfCount;
    Rf_LIST_MODE_NSFT_RPT_T     report[RF_MAX_LIST_MODE_COMMAND_COUNT];
} Rf_LIST_MODE_NSFT_RPT_CNF_T;
```

Description:

Start NSFT list mode process with given configuration.

Return Value:

Table 6-207 The return value of META_Rf_List_Mode_NSFT_Start_r

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.
	NSFT list mode Start Fail - please check environment setting (ex.cfg file, UI instructment setting, waveform lock) - If environment setting is right, but problem still happen, please provide us with ELT L1 log to analysis

Parameter:

Table 6-208 The parameter of META_Rf_List_Mode_NSFT_Start_r

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
Rx_qual	OUT	RX quality indicator

6.6.44 META_Rf_PmEx

Definition:

META_RESULT __stdcall META_Rf_PmEx(unsigned int ms_timeout, const RfPm_Req *req, RfPm_Cnf *cnf);

META_RESULT __stdcall META_Rf_PmEx_r(const int meta_handle, unsigned int ms_timeout, const RfPm_Req *req, RfPm_Cnf *cnf);

```
typedef struct
{
    ARFCN       arfcn;          // Absolute radio frequency channel number
    char        sampleNoPerFrame;   // number of samples per frame
    Gain        gain;           // Gain that should be used in power management
    short       frames;         // number of frames
} RfPm_Req;
```

```
typedef struct
{
   int          power;          // average power
   int          deviation;       // deviation of power measurement
   Gain          usedGain;         // Gain that is used
   unsigned char   ok;              // status
   RfPmExtraInfo_T extra_info;       // extra info
} RfPm_Cnf;


typedef struct {
   unsigned char   valid;            // if valid != zero, it means the extra info is meaningful.
                              // otherwise, the extra info should be ignore.
   int          iOffset;
   int          qOffset;
   int          validSamples;
} RfPmExtraInfo_T;
```

Description:

Get the power measurement result from the target

Return Value:

Table 6-209 The return value of META_Rf_PmEx

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-210 The parameter of META_Rf_PmEx

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

Parameter	IN/OUT	Description
req	IN	Request parameter
Cnf	OUT	Confirm parameter

6.6.45 META_Rf_IfPm

Definition:

META_RESULT __stdcall META_Rf_IfPm(unsigned int ms_timeout, const RfIfPm_Req *req, RfPm_Cnf *cnf);

META_RESULT __stdcall META_Rf_IfPm_r(const int meta_handle, unsigned int ms_timeout, const RfIfPm_Req *req, RfPm_Cnf *cnf);

```
typedef struct
{
    /// original power scan request
    RfPm_Req        m_Pm;
    /// if flag used for specifying the if flag in power scan (override the if flag setting)
    char   m_IfFlag;
} RfIfPm_Req;

typedef struct
{
    ARFCN          arfcn;           // Absolute radio frequency channel number
    char          sampleNoPerFrame;  // number of samples per frame
    Gain          gain;            // Gain that should be used in power management
    short         frames;           // number of frames
} RfPm_Req;

typedef struct
{
    int          power;           // average power
    int          deviation;        // deviation of power measurement
    Gain          usedGain;         // Gain that is used
    unsigned char   ok;              // status
    RfPmExtraInfo_T extra_info;        // extra info
} RfPm_Cnf;


typedef struct {
    unsigned char   valid;            // if valid != zero, it means the extra info is meaningful.
                                   // otherwise, the extra info should be ignore.
    int          iOffset;
    int          qOffset;
    int          validSamples;
} RfPmExtraInfo_T;
```

Description:

Get the power measurement result from the target

Return Value:

Table 6-211 The return value of META_Rf_IfPm

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-212 The parameter of META_Rf_IfPm

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	Request parameter
Cnf	OUT	Confirm parameter

6.6.46 META_Rf_GetTXPCDetectorValueByPCLGMSK

Definition:

META_RESULT __stdcall META_Rf_GetTXPCDetectorValueByPCLGMSK(unsigned int ms_timeout, const Rf_GET_TXPC_PD_REQ_T* req, unsigned short * PDValue);

META_RESULT __stdcall META_Rf_GetTXPCDetectorValueByPCLGMSK_r(const int meta_handle, unsigned int ms_timeout, const Rf_GET_TXPC_PD_REQ_T* req, unsigned short * PDValue);

```
typedef struct
{
    unsigned char  band;
    short pcl;
}Rf_GET_TXPC_PD_REQ_T;
```

Description:

Get the closed-loop detector measurement result

Return Value:

Table 6-213 The return value of META_Rf_GetTXPCDetectorValueByPCLGMSK

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-214 The parameter of META_Rf_GetTXPCDetectorValueByPCLGMSK

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	Request parameter
PDvalue	OUT	Detector measurement result

6.6.47 META_Rf_GetTXPCDetectorValueByPCLEPSK

Definition:

META_RESULT __stdcall META_Rf_GetTXPCDetectorValueByPCLEPSK(unsigned int ms_timeout, const Rf_GET_TXPC_PD_REQ_T* req, unsigned short * PDValue);

META_RESULT __stdcall META_Rf_GetTXPCDetectorValueByPCLEPSK_r(const int meta_handle, unsigned int ms_timeout, const Rf_GET_TXPC_PD_REQ_T* req, unsigned short * PDValue);

```
typedef struct
{
    unsigned char  band;
    short pcl;
}Rf_GET_TXPC_PD_REQ_T;
```

Description:

Refer to META_Rf_GetTXPCDetectorValueByPCLGMSK

Return Value:

Table 6-215 The return value of META_Rf_GetTXPCDetectorValueByPCLEPSK

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-216 The parameter of META_Rf_GetTXPCDetectorValueByPCLEPSK

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	Request parameter
PDvalue	OUT	Detector measurement result

6.6.48 META_Rf_GetTXPCDetectorValueGMSK

Definition:

META_RESULT __stdcall META_Rf_GetTXPCDetectorValueGMSK(unsigned int ms_timeout, l1cal_txpc_T *table);

META_RESULT __stdcall META_Rf_GetTXPCDetectorValueGMSK_r(const int meta_handle, unsigned int ms_timeout, l1cal_txpc_T *table);

```
typedef struct
{
  char          is_calibrated;
  sTXPC_ADCDATA  adc[FrequencyBandCount];
  short         temperature;
  sTXPC_TEMPDATA temp[FrequencyBandCount];
} sTXPC_L1CAL;

typedef sTXPC_L1CAL l1cal_txpc_T;
```

Description:

Get the detector measurement result of all PCL in all supported frequency band. (GMSK)

Return Value:

Table 6-217 The return value of META_Rf_GetTXPCDetectorValueGMSK

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-218 The parameter of META_Rf_GetTXPCDetectorValueGMSK

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	Request parameter
table	OUT	Detector measurement result of all PCL in all supported frequency band

6.6.49 META_Rf_GetTXPCDetectorValueEPSK

Definition:

META_RESULT __stdcall META_Rf_GetTXPCDetectorValueEPSK(unsigned int ms_timeout, l1cal_txpc_T *table);

META_RESULT __stdcall META_Rf_GetTXPCDetectorValueEPSK_r(const int meta_handle, unsigned int ms_timeout, l1cal_txpc_T *table);

```
typedef struct
{
  char          is_calibrated;
  sTXPC_ADCDATA  adc[FrequencyBandCount];
  short         temperature;
  sTXPC_TEMPDATA temp[FrequencyBandCount];
} sTXPC_L1CAL;

typedef sTXPC_L1CAL l1cal_txpc_T;
```

Description:

Get the detector measurement result of all PCL in all supported frequency band. (EPSK)

Return Value:

Table 6-219 The return value of META_Rf_GetTXPCDetectorValueEPSK

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-220 The parameter of META_Rf_GetTXPCDetectorValueEPSK

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	Request parameter
table	OUT	Detector measurement result of all PCL in all supported frequency band

6.6.50 META_Rf_GetTXPCSubbandCompensationGMSK

Definition:

META_RESULT __stdcall META_Rf_GetTXPCSubbandCompensationGMSK(unsigned int ms_timeout, const unsigned char band, l1cal_rampTable_T *ramp);

```
META_RESULT  __stdcall META_Rf_GetTXPCSubbandCompensationGMSK_r(const int meta_handle, unsigned int
ms_timeout, const unsigned char band, l1cal_rampTable_T  *ramp);


typedef struct
{
   sRAMPDATA        rampData;                  // apc ramp profile of all bands
}l1cal_rampTable_T;


#define PROFILE_NUM           16
#define ARFCN_SECTION_NUM       12
#define ARFCN_SECTION_NUM_Ex      64
typedef  struct
{
   unsigned char   point[2][16];   // ramp up/down profile

} sRAMPAREADATA;


typedef  struct
{
   short          max_arfcn;     // sub-band boundary of this PCL weighting area
   unsigned short   mid_level;      // PCLboundary level to apply high/low weighting
   unsigned short   hi_weight;      // scale factor of PCLs higher than mid_level
   unsigned short   low_weight;     // scale factor of PCLs lower than mid_level

} sARFCN_SECTION;


typedef  struct
{
   int          lowest_power;                   // The lower apc power of the indicated band
```

```
  unsigned short   power[16];                    // The mapping of power level to apc dac value
  sRAMPAREADATA    ramp[ PROFILE_NUM ];              // ramp profile
  sARFCN_SECTION   arfcn_weight[ ARFCN_SECTION_NUM ];  // profile of weighting power level by PCL and sub-
band
  unsigned short   battery_compensate[3][3];          // [volt][temp]
  short            tx_afc_offset;
} sRAMPDATA;
```

Description:

Get the detector subband measurement result

Return Value:

Table 6-221 The return value of META_Rf_GetTXPCSubbandCompensationGMSK

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-222 The parameter of META_Rf_GetTXPCSubbandCompensationGMSK

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
req	IN	Request parameter
ramp	OUT	Detector measurement result of all subband

6.6.51 META_Rf_GetSpecialCoef

Definition:

META_RESULT __stdcall META_Rf_GetSpecialCoef(unsigned int ms_timeout, unsigned int rfid, void *coef_struct);

META_RESULT __stdcall META_Rf_GetSpecialCoef_r(const int meta_handle, unsigned int ms_timeout, unsigned int rfid, void *coef_struct);

typedef struct

```
{
   short w_re[19];
   short w_im[19];
}RF_AvgW_Coef_T;
```

Description:

Get the rf special coefficient from target

Return Value:

Table 6-223 The return value of META_Rf_GetSpecialCoef

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-224 The parameter of META_Rf_GetSpecialCoef

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
coef_struct	IN/OUT	The pointer of the special coefficient structure to store the read back data

Sample Code:

```
RF_AvgW_Coef_T  sWCoef;
META_Rf_GetSpecialCoef(3000, RF_ID_MT6255RF, (void *) &sWCoef);
```

6.6.52 META_Rf_StartFdtDL_Big

Definition:

META_RESULT __stdcall META_Rf_StartFdtDL_Big(unsigned int ms_timeout, const Rf_DTS_REQ_BIG_T *fdt_dl_req, Rf_DTS_CNF_BIG_T *fdt_dl_cnf);

META_RESULT __stdcall META_Rf_StartFdtDL_Big_r(const int meta_handle, unsigned int ms_timeout, const Rf_DTS_REQ_BIG_T *fdt_dl_req, Rf_DTS_CNF_BIG_T *fdt_dl_cnf);

```
/**
 * Description:
```

```
 *  Extenstion DTS interface for gain mode combine
 **/
#define MAX_STEP_EX_CNT   100
typedef struct
{
   bool              afc_cal;
   bool              pl_cal;       // Control whether Path loss calibration is needed or not
   char              sync_sb_num; // the SB frame numbers needed for sync process before path loss calibration
   short              power;             // the power level expected to measure from test set
   Rf_DSSAFC_T        AfcDSS;
   char              step_cnt;
   Rf_DSSPL_T         PathLossDSS[MAX_STEP_EX_CNT-2]; // because sync process will need 2 steps
}Rf_DTS_REQ_BIG_T;

typedef struct
{
   RF_DTS_GET_RESULT_STATUS     status;
   Rf_DSSPL_RESULT_BIG_T        PLResult;
   Rf_DSSAFC_RESULT_T          AfcResult;
   Rf_FHC_DTSM_INFO_T           DtsmInfo;
} Rf_DTS_CNF_BIG_T;
```

Description:

This is an extension function of META_Rf_StartFdtDL. The function is available only when following capability condition is satisfied.

```
        typedef struct
        {
          //....
          RfFactoryModeCalItem dts_gain_cmb;
          RfFactoryModeCalItem uts_band_cmb;
          //-----------------------------------------------------------------------
          // parameter: 0, capable: 1, mandatory: 1 ==> support FHC double-band combine, but only 50
        steps (orignal interface)
          // parameter: 1, capable: 1, mandatory: 1 ==> support FHC double-band combine, but only 100
        steps (Big interface)
          // ...
}RfCalibrationItem;
```

Return Value:

Table 6-225 The return value of META_Rf_StartFdtDL_Big

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-226 The parameter of META_Rf_StartFdtDL_Big

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
fdt_dl_req	IN	Downlink calibration parameter
fdt_dl_cnf	IN/OUT	Downlink calibration result

6.6.53 META_Rf_StartFdtDLNotWaitResult_Big

Definition:

META_RESULT __stdcall META_Rf_StartFdtDLNotWaitResult_Big(unsigned int ms_timeout, const Rf_DTS_REQ_BIG_T *fdt_dl_req);

META_RESULT __stdcall META_Rf_StartFdtDLNotWaitResult_Big_r(const int meta_handle, unsigned int ms_timeout, const Rf_DTS_REQ_BIG_T *fdt_dl_req);

Description:

This is an extension function of META_Rf_StartFdtDLNotWaitResult. The function is available only when following capability condition is satisfied.

Return Value:

Table 6-227 The return value of META_Rf_StartFdtDLNotWaitResult_Big

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-228 The parameter of META_Rf_StartFdtDLNotWaitResult_Big

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
fdt_dl_req	IN	Downlink calibration parameter

6.6.54 META_Rf_GetFdtDL_Big

Definition:

META_RESULT __stdcall META_Rf_GetFdtDL_Big(unsigned int ms_timeout, Rf_DTS_CNF_BIG_T *fdt_dl_get_result_cnf);

META_RESULT __stdcall META_Rf_GetFdtDL_Big_r(const int meta_handle, unsigned int ms_timeout, Rf_DTS_CNF_BIG_T *fdt_dl_get_result_cnf);

Description:

This is an extension function of META_Rf_GetFdtDL. The function is available only when following capability condition is satisfied.

Return Value:

Table 6-229 The return value of META_Rf_GetFdtDL_Big

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-230 The parameter of META_Rf_GetFdtDL_Big

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
fdt_dl_get_result_cnf	IN/OUT	Downlink calibration result

6.6.55 META_Rf_StartFdtUL_Big

Definition:

```
META_RESULT  __stdcall  META_Rf_StartFdtUL_Big(unsigned  int  ms_timeout,  const  Rf_UTS_REQ_BIG_T *fdt_ul_req);

META_RESULT  __stdcall META_Rf_StartFdtUL_Big_r(const int meta_handle, unsigned int ms_timeout, const Rf_UTS_REQ_BIG_T  *fdt_ul_req);


#define MAX_STEP_EX_CNT   100


/**
 * Description:
 *  Extenstion UTS interface for middel chanenl tx pcl calibration band combine
 **/
typedef struct
{
   char            step_cnt;
   short           high_apc_dcoffset[FrequencyBandCount];
   Rf_USSAPC_T       ApcUSS[MAX_STEP_EX_CNT];
}Rf_UTS_REQ_BIG_T;
```

Description:

This is an extension function of META_Rf_StartFdtUL. The function is available only when following capability condition is satisfied. Extension interface for the TX quad band combin.

Return Value:

Table 6-231 The return value of META_Rf_StartFdtUL_Big

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-232 The parameter of META_Rf_StartFdtUL_Big

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
fdt_ul_req	IN	Uplink calibration parameter

6.7 Exported Functions for NVRAM Read/Write/Buffer manipulation

6.7.1 META_NVRAM_Init

Definition:

```
META_RESULT  __stdcall META_NVRAM_Init(
                        const char *PathName,
                        unsigned long *p_nvram_CatcherTranAddr)
```

Description:

This function initializes the NVRAM-related functionality of META-DLL.

Return Value:

Table 6-233 The return value of META_NVRAM_Init

Return value	Description
META_SUCCESS	Success
META_FILE_BAD	File doesn’t exist or open file failed.
META_FAILED	Import NVRAM database file failed.

Parameter:

Table 6-234 The parameter of META_NVRAM_Init

Parameter	IN/OUT	Description
PathName	IN	Path of file, which contains NVRAM information.

Parameter	IN/OUT	Description
p_nvram_CatcherTranAddr	OUT	unsigned long address of CatcherTran Pointer, which is used to initialize the ActiveX Control (refers to UI-DLL).

Note:

For multi-thread developers, here is an example pseudo code to init the database for mult-threads.

```
META_Init();
META_NVRAM_Init();
FOR(I=0; I < MAX_THREADS; I++)
{
    META_GetAvailableHandle(&META_HANDLE);
    META_Init_r(META_HANDLE);
    CREATE_THREAD(META_HANDLE);
}
```

In the NVRAM authenitication supported platform, the authenitication key is revieved when connection established. Therfore, developers must have to initial the nvram database at first before connect with targets.

Bug fix:

For multi-thread developers, the NVRAM access will fail when one of the thread disconnect with target. It is because the authenitication key will be erased when disconnection. The bug is fixed in version of v6.1244.04 and the version after v6.1308.1 META DLL.

6.7.2 META_NVRAM_Init_Ex_Mdtype_r

Definition:

META_RESULT __stdcall META_NVRAM_Init_Ex_Mdtype_r(const int meta_handle, const unsigned int md_index, const unsigned int mdtype_index, const char* db_path, unsigned long * p_nvram_CatcherTranAddr)

Description:

This function initializes the NVRAM-related functionality of META-DLL. This API is enhanced for world phone feature (multiple SW image/MD type feature.)

Return Value:

Table 6-235 The return value of META_NVRAM_Init_Ex_Mdtype_r

Return value	Description
META_SUCCESS	Success
META_INVALID_HANDLE	Meta handle is invalid.
META_INVALID_ARGUMENTS	Some NVRAM arguments are invalid.
META_MAUI_DB_INCONSISTENT	NVRAM database is inconsistent.
META_FILE_BAD	File doesn’t exist or open file failed.
META_FAILED	Import NVRAM database file failed.

Parameter:

Table 6-236 The parameter of META_NVRAM_Init_Ex_Mdtype_r

Parameter	IN/OUT	Description
md_index	IN	Modem index in dual-tlak and world phone feature.
mdtype_index	IN	Modem type index (SW image index) in dual-tlak and world phone feature.
db_path	IN	Path of file, which contains NVRAM information.
p_nvram_CatcherTranAddr	OUT	unsigned long address of CatcherTran Pointer, which is used to initialize the ActiveX Control (refers to UI-DLL).

Note:

For parameter md_index, if value equals to 0, it means appointed 1st modem. If value equals to 1, it means appointed 2nd modem. For parameter mdtype_index, if value equals to 0, it means appointed 1st modem type. If value equals to 1, it means appointed 2nd modem type.

6.7.3 META_NVRAM_Reset

Definition:

```
        META_RESULT  __stdcall META_NVRAM_Reset(
                                    const FT_NVRAM_RESET_REQ *req,
                                    const META_NVRAM_Reset_CNF cb,
                                    short *token, void *usrData)
```

```
typedef enum
{
       NVRAM_RESET_ALL,                        // Reset all data items
```

```
        NVRAM_RESET_USER,                        // Reset data items in user category
        NVRAM_RESET_SYSTEM,                // Reset data items in system category
        NVRAM_RESET_CERTAIN                      // Reset certain data item
        NVRAM_RESET_FACTORY                // Reset to factory default value, all the LIDs has
                                                   FACTORY attribute will be reseted
} ResetCategory;
typedef struct
{
        ResetCategory           category;         // Reset category
        const char              *LID;             // The name of logical data item ID , it will be used
                                                 // if and only if ResetCategory = NVRAM_RESET_CERTAIN,
                                                 // in other case you can just assign NULL.
} FT_NVRAM_RESET_REQ;
typedef struct
{
        unsigned char           status;           // The status of Reset
} FT_NVRAM_RESET_CNF;
```

Description:

This function resets the data items.

1. Reset the whole USER category: You must set ResetCategory = NVRAM_RESET_USER. LID is ignored in this case.
2. Reset the whole SYSTEM category: You must set ResetCategory = NVRAM_RESET_SYSTEM. LID is ignored in this case.
3. Reset both of the USER and SYSTEM categories: You must set ResetCategory = NVRAM_RESET_ALL. LID is ignored in this case.
4. Reset one certain LID, you must set ResetCategory = NVRAM_RESET_CERTAIN, and also specify which LID you want to reset.

Note:

NVRAM_RESET_ALL, NVRAM_RESET_USER, NVRAM_RESET_SYSTEM, NVRAM_RESET_CERTAIN are obsolete since W10.12, and all branches have been patched (refer to CR:MAUI_01981104).

Callback:

typedef void (__stdcall *META_NVRAM_Reset_CNF)(FT_NVRAM_RESET_CNF cnf, short token, void *usrData);

Return Value:

Table 6-237 The return value of META_NVRAM_Reset

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target failed.
META_INTERNAL_DB_ERR	Cannot find structure information from InternalDB.

Parameter:

Table 6-238 The parameter of META_NVRAM_Reset

Parameter	IN/OUT	Description
req	IN	Request
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from the target.
token	IN/OUT	Token used by the user to uninstall the callback function.
usrData	IN	Parameter being used by the user.

6.7.4 META_NVRAM_Read

Definition:

```
META_RESULT __stdcall META_NVRAM_Read(
                              const FT_NVRAM_READ_REQ *req,
                              FT_NVRAM_READ_CNF *cnf,
                              const META_NVRAM_Read_CNF cb,
                              short *token, void *usrData)
typedef struct
{
    const char          *LID;          // The name of logical data item ID
    unsigned short      RID;           // Record ID (the first record is 1)
} FT_NVRAM_READ_REQ;
typedef struct
{
            unsigned short          LID;             // Logical data item ID of a EF
            unsigned short          RID;             // Record ID (the first record is 1)
            unsigned char           status;          // 0: read ok; others: read failed.
            unsigned int            len;             // [IN] Length of Buffer, [OUT] Length of read data
            unsigned char           *buf;            // Buffer that will contains the content of record
} FT_NVRAM_READ_CNF;
```

Description:

This function reads the content of a specific record.

Callback:

```
typedef void (__stdcall *META_NVRAM_Read_CNF)(FT_NVRAM_READ_CNF *cnf, short token, void
*usrData);
```

Note:

The "buf" field of FT_NVRAM_READ_CNF is a pointer to a buffer. User should provide this buffer for META_DLL to store the data of read record. The “len” field of FT_NVRAM_READ_CNF indicates the size of “buf” you allocated. When the data is read back, “len” will be replaced with the actual size of the data.

Return Value:

Table 6-239 The return value of META_NVRAM_Read

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_LID_INVALID	Invalid LID.
META_BUFFER_LEN	cnf->buf is not prepared, or cnf->len error.

Parameter:

Table 6-240 The parameter of META_NVRAM_Read

Parameter	IN/OUT	Description
req	IN	Request
cnf	IN/OUT	Pointer to FT_NVRAM_READ_CNF, which will be the parameter of callback function.

Parameter	IN/OUT	Description
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.7.5 META_NVRAM_Read_Ex

Definition:

META_RESULT __stdcall META_NVRAM_Read_Ex(const unsigned int ms_timeout, const FT_NVRAM_READ_REQ *req, FT_NVRAM_READ_CNF *cnf);

META_RESULT __stdcall META_NVRAM_Read_Ex_r(const int meta_handle, unsigned int ms_timeout, const FT_NVRAM_READ_REQ *req, FT_NVRAM_READ_CNF *cnf);

```
typedef struct
{
        const char          *LID;       // The name of logical data item ID
        unsigned short      RID;        // Record ID (the first record is 1)
} FT_NVRAM_READ_REQ;
typedef struct
{
        unsigned short      LID;        // Logical data item ID of a EF
        unsigned short      RID;        // Record ID (the first record is 1)
        unsigned char       status;     // 0: read ok; others: read failed.
        unsigned int        len;        // [IN] Length of Buffer, [OUT] Length of read data
        unsigned char       *buf;       // Buffer that will contains the content of record
} FT_NVRAM_READ_CNF;
```

Description:

This function reads the content of a specific NVRAM record, and returns after the entire transaction completes(send request message, wait for confirm message from target).

Note: Unlike META_NVRAM_Read, META_NVRAM_Read operation asks developers to handle the confirm message sent from target side via the callback they registered.

Note:

The "buf" field of FT_NVRAM_READ_CNF is a pointer to a buffer. User should provide this buffer for META_DLL to store the data of read record. The “len” field of FT_NVRAM_READ_CNF indicates the size of “buf” you allocated. When the data is read back, “len” will be replaced with the actual size of the data.

Return Value:

Table 6-241 The return value of META_NVRAM_Read_Ex

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_LID_INVALID	Invalid LID.
META_BUFFER_LEN	cnf->buf is not prepared, or cnf->len error.

Parameter:

Table 6-242 The parameter of META_NVRAM_Read_Ex

Parameter	IN/OUT	Description
req	IN	Request
cnf	IN/OUT	Pointer to FT_NVRAM_READ_CNF, which will be the parameter of callback function.
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.7.6 META_NVRAM_Write

Definition:

```
META_RESULT __stdcall META_NVRAM_Write(
                    const FT_NVRAM_WRITE_REQ *req,
                    const META_NVRAM_Write_CNF cb,
                    short *token, void *usrData)
```

typedef struct

```
{
        const char          *LID;       // The name of logical data item ID
        unsigned short      RID;        // Record ID (the first record is 1)
        unsigned int        len;        // Length of write data
        unsigned char       *buf;       // Buffer that contains the content of record
} FT_NVRAM_WRITE_REQ;
typedef struct
{
        unsigned short      LID;        // Logical data item ID of a EF
        unsigned short      RID;        // Record ID (the first record is 1)
        unsigned char       status;     // 0: write ok; others: write failed.
} FT_NVRAM_WRITE_CNF;
```

Description:

This function writes the content of a specific record.

Callback:

typedef void (__stdcall *META_NVRAM_Write_CNF)(FT_NVRAM_WRITE_CNF cnf, short token, void *usrData);

Return Value:

Table 6-243 The return value of META_NVRAM_Write

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_LID_INVALID	Invalid LID.
META_BUFFER_LEN	The length of buffer is not enough.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-244 The parameter of META_NVRAM_Write

Parameter	IN/OUT	Description
req	IN	Request

Parameter	IN/OUT	Description
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.7.7 META_NVRAM_Write_Ex

Definition:

```
META_RESULT __stdcall META_NVRAM_Write_Ex(const unsigned int ms_timeout, const FT_NVRAM_WRITE_REQ *req, FT_NVRAM_WRITE_CNF *cnf);

META_RESULT __stdcall META_NVRAM_Write_Ex_r(const int meta_handle,
                                            const unsigned int ms_timeout,
                                            const FT_NVRAM_WRITE_REQ *req,
                                            FT_NVRAM_WRITE_CNF *cnf);

typedef struct
{
        const char          *LID;       // The name of logical data item ID
        unsigned short      RID;        // Record ID (the first record is 1)
        unsigned int        len;        // Length of write data
        unsigned char       *buf;       // Buffer that contains the content of record
} FT_NVRAM_WRITE_REQ;

typedef struct
{
        unsigned short      LID;        // Logical data item ID of a EF
        unsigned short      RID;        // Record ID (the first record is 1)
        unsigned char       status;     // 0: write ok; others: write failed.
} FT_NVRAM_WRITE_CNF;
```

Description:

This function writes the content of a specific NVRAM record, and returns after the entire transaction completes(send request message, wait for confirm message from target).
Note: Unlike META_NVRAM_Write, META_NVRAM_Write operation asks developers to handle the confirm message sent from target side via the callback they registered.

Return Value:

Table 6-245 The return value of META_NVRAM_Write_Ex

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_LID_INVALID	Invalid LID.
META_BUFFER_LEN	The length of buffer is not enough.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-246 The parameter of META_NVRAM_Write_Ex

Parameter	IN/OUT	Description
req	IN	Request
cb	IN	Callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the callback function.
usrData	IN	Parameter used by user.

6.7.8 META_NVRAM_OTP_LockDown

Definition:

META_RESULT __stdcall META_NVRAM_OTP_LockDown(unsigned int ms_timeout);

Description:

Lockdown entire OTP area.

Return Value:

Table 6-247 The return value of META_NVRAM_OTP_LockDown

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-248 The parameter of META_NVRAM_OTP_LockDown

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value (in milliseconds).

6.7.9 META_NVRAM_GetAllLIDNameLength

Definition:

META_RESULT __stdcall META_NVRAM_GetAllLIDNameLength(int *len)

Description:

This function returns the total length of the buffer that is used to store all LID name strings.
len is including '\0' for each string.

Return Value:

Table 6-249 The return value of META_NVRAM_GetAllLIDNameLength

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.

Parameter:

Table 6-250 The parameter of META_NVRAM_GetAllLIDNameLength

Parameter	IN/OUT	Description
len	OUT	The total length of the buffer that is used to store all LID name strings.

6.7.10 META_NVRAM_GetAllLIDName

Definition:

META_RESULT __stdcall META_NVRAM_GetAllLIDName(char *buf, const int buf_len, int *NofLID)

Description:

This function will store all LID strings into buffer and return the total count of LIDs.
All strings are separated by '\0' character.

Return Value:

Table 6-251 The return value of META_NVRAM_GetAllLIDName

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.
META_BUFFER_LEN	Input buf is not enough to store all LID strings.

Parameter:

Table 6-252 The parameter of META_NVRAM_GetAllLIDName

Parameter	IN/OUT	Description
buf	IN/OUT	The buffer used to store all LID name strings.
buf_len	IN	The total length of buffer
NofLID	IN/OUT	The total count of LIDs.

6.7.11 META_NVRAM_GetRecStructNameLength

Definition:

META_RESULT __stdcall META_NVRAM_GetRecStructNameLength(const char *LID, int *len)

Description:

This function is used to get the correspondent structure name by specific LID.
len is including '\0' character.

Return Value:

Table 6-253 The return value of META_NVRAM_GetRecStructNameLength

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.
META_LID_INVALID	LID is not found, or the data type of LID is not structure type.

Parameter:

Table 6-254 The parameter of META_NVRAM_GetRecStructNameLength

Parameter	IN/OUT	Description
LID	IN	LID name.
len	IN/OUT	The length of the correspondent structure.

6.7.12 META_NVRAM_GetRecStructName

Definition:

META_RESULT __stdcall META_NVRAM_GetRecStructName(const char *LID, char *buf, const int buf_len)

Description:

This function will store the correspondent structure name into buffer by LID.

Return Value:

Table 6-255 The return value of META_NVRAM_GetRecStructName

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.
META_BUFFER_LEN	Input buf is not enough to store all LID strings.
META_LID_INVALID	LID is not found, or the data type of LID is not structure type.

Parameter:

Table 6-256 The parameter of META_NVRAM_GetRecStructName

Parameter	IN/OUT	Description
LID	IN	LID name.
buf	IN/OUT	The buffer used to store all LID name strings.
buf_len	IN	The total length of buffer

6.7.13 META_NVRAM_GetAllRecFieldNameLength

Definition:

META_RESULT __stdcall META_NVRAM_GetAllRecFieldNameLength(const char *LID, int *len)

Description:

This function returns the total length of the buffer that is used to store all structure field names. len is including '\0' for each field name.

Return Value:

Table 6-257 The return value of META_NVRAM_GetAllRecFieldNameLength

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.
META_LID_INVALID	LID is not found, or the data type of LID is not structure type.

Return value	Description
META_INTERNAL_DB_ERR	Can't find structure info from NVRAM InternalDB.

Parameter:

Table 6-258 The parameter of META_NVRAM_GetAllRecFieldNameLength

Parameter	IN/OUT	Description
LID	IN	LID name.
len	IN/OUT	The total length of the buffer that is used to store all LID name strings.

6.7.14 META_NVRAM_GetAllRecFieldName

Definition:

META_RESULT __stdcall META_NVRAM_GetAllRecFieldName(const char *LID, char *buf, const int buf_len, int *NofField)

Description:

This function will store all field names into buffer and return the total count of fields.
All strings are separated by '\0' character.

Remark:

For example:
If the structure is:

```
typedef  struct
 {
        int                         lowest_power;
        unsigned short              power[16];
        sRAMPAREADATA  ramp[ PROFILE_NUM ];
        sARFCN_SECTION arfcn_weight[ ARFCN_SECTION_NUM ];
} sRAMPDATA;
```

The buffer will be stored like this: "lowest_power\0power\0ramp\0arfcn_weight\0"
, and return NofField = 4.

Return Value:

Table 6-259 The return value of META_NVRAM_GetAllRecFieldName

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.
META_BUFFER_LEN	Input buf is not enough to store all field names.
META_LID_INVALID	LID is not found, or the data type of LID is not structure type.

Parameter:

Table 6-260 The parameter of META_NVRAM_GetAllRecFieldName

Parameter	IN/OUT	Description
LID	IN	LID name.
buf	IN/OUT	The buffer used to store all LID name strings.
buf_len	IN	The total length of buffer
NofField	IN/OUT	The total count of fields

6.7.15 META_NVRAM_GetRecNum

Definition:

META_RESULT __stdcall META_NVRAM_GetRecNum(const char *LID, int *num)

Description:

This function returns the total record number of specific logical item ID.

Return Value:

Table 6-261 The return value of META_NVRAM_GetRecNum

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.
META_LID_INVALID	Invalid LID.

Parameter:

Table 6-262 The parameter of META_NVRAM_GetRecNum

Parameter	IN/OUT	Description
LID	IN	The name of logical data item ID
num	OUT	The record number of specific LID.

6.7.16 META_NVRAM_GetRecLen

Definition:

META_RESULT __stdcall META_NVRAM_GetRecLen(const char *LID, int *len)

Description:

This function returns the necessary size of buffer, which is capable to contain a specific record.

Return Value:

Table 6-263 The return value of META_NVRAM_GetRecLen

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.
META_LID_INVALID	Invalid LID.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-264 The parameter of META_NVRAM_GetRecLen

Parameter	IN/OUT	Description
LID	IN	The name of logical data item ID
len	OUT	The size of each record.

6.7.17 META_NVRAM_GetLIDVersion

Definition:

META_RESULT __stdcall META_NVRAM_GetLIDVersion(const char *LID,unsigned short *ver)

Description:

This function input LID and returns the version of specific LID.

Return Value:

Table 6-265 The return value of META_NVRAM_GetLIDVersion

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.
META_LID_INVALID	Invalid LID.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-266 The parameter of META_NVRAM_GetLIDVersion

Parameter	IN/OUT	Description
LID	IN	The name of logical data item ID
ver	OUT	The LID version of specific record.

6.7.18 META_NVRAM_CheckFieldNameExist

Definition:

META_RESULT __stdcall META_NVRAM_CheckFieldNameExist(const char *LID, const char *Field, BOOL *result);

Description:

This function input LID to get struct name and input field name of struct member to check if Field exist in specific LID.

Return Value:

Table 6-267 The return value of META_NVRAM_CheckFieldNameExist

Return value	Description
META_SUCCESS	Success
META_FAILED	Input arguments are invalid.
META_LID_INVALID	Invalid LID.
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-268 The parameter of META_NVRAM_CheckFieldNameExist

Parameter	IN/OUT	Description
LID	IN	The name of logical data item ID
Field	IN	The LID version of specific record.
result	OUT	True means Exist, False means nonexist

6.7.19 META_NVRAM_SetRecFieldValue

Definition:

META_RESULT __stdcall META_NVRAM_SetRecFieldValue(

const char *LID,

const char *field,

char *buf, const int buf_len,

void *value, const int value_len)

Description:

This function sets the value of a specific field in a specific record. The record type must be structure type.

Return Value:

Table 6-269 The return value of META_NVRAM_SetRecFieldValue

Return value	Description
META_SUCCESS	Success
META_LID_INVALID	Invalid LID.
META_BUFFER_LEN	The length of buffer is not enough.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-270 The parameter of META_NVRAM_SetRecFieldValue

Parameter	IN/OUT	Description
LID	IN	The name of logical data item ID

field	IN	Field name.
		If the field is an array, you can only use [n] to get the value of element in the array.
		
		For example:
		struct TEST {
		  int a[10];
		};
		
		you can use MEAT_NVRAM_SetRecFieldValue(..., "a[2]", ...);
		to get the 3rd element,

buf	IN/OUT	Buffer that holds the content of a specific record.
buf_len	IN	Length of buf.
value	IN	Pointer to a location in which the value is located.
value_len	IN	The size of the location to which value points.

6.7.20 META_NVRAM_GetRecFieldValue

Definition:

META_RESULT __stdcall META_NVRAM_GetRecFieldValue(

const char *LID,

const char *field,

const char *buf, const int buf_len,

void *value, const int value_len)

Description:

This function gets the value of a specific field in a specific record. The record type must be structure type.

Return Value:

Table 6-271 The return value of META_NVRAM_GetRecFieldValue

Return value	Description
META_SUCCESS	Success
META_LID_INVALID	Invalid LID.
META_BUFFER_LEN	The length of buffer is not enough.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-272 The parameter of META_NVRAM_GetRecFieldValue

Parameter	IN/OUT	Description
LID	IN	The name of logical data item ID

field	IN	Field name.
		If the field is an array, you can only use [n] to get the value of element in the array.
		
		For example:
		struct TEST {
		  int a[10];
		};
		
		You can use MEAT_NVRAM_SetRecFieldValue(…, "a[2]", ….);
		to get the 3rd element,

buf	IN	Buffer that holds the content of a specific record.
buf_len	IN	Length of buf.
value	IN/OUT	Pointer to a location in which the value is located.
value_len	IN	The size of the location to which value points.

6.7.21 META_NVRAM_SetRecFieldBitValue

Definition:

```
META_RESULT  __stdcall META_NVRAM_SetRecFieldBitValue(
                                        const char *LID,
                                        const char *field,
                                        const char *bitname,
                                        char *buf, const int buf_len,
                                        const int bitvalue)
```

Description:

This function sets the bitvalue of a specific field in a specific record. The record type must be structure type.

Return Value:

Table 6-273 The return value of META_NVRAM_SetRecFieldBitValue

Return value	Description
META_SUCCESS	Success
META_LID_INVALID	Invalid LID.
META_BUFFER_LEN	The length of buffer is not enough.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-274 The parameter of META_NVRAM_SetRecFieldBitValue

Parameter	IN/OUT	Description
LID	IN	The name of logical data item ID

field	IN	Field name.
		If the field is an array, you can only use [n] to get the value of element in the array.
		
		For example:
		struct TEST {
		  int a[10];
		};
		
		you can use MEAT_NVRAM_SetRecFieldValue(…, “a[2]”, ….);
		to get the 3rd element,

field	IN	Bit field name.
buf	IN/OUT	Buffer that holds the content of a specific record.
buf_len	IN	Length of buf.
bitvalue	IN	The value for that bit field.

6.7.22 META_NVRAM_GetRecFieldBitValue

Definition:

```
        META_RESULT  __stdcall META_NVRAM_GetRecFieldBitValue(
                                            const char *LID,
                                            const char *field,
                                            const char *bitname,
                                            const char *buf, const int buf_len,
                                            int *bitvalue)
```

Description:

This function gets the bitvalue of a specific field in a specific record. The record type must be structure type.

Return Value:

Table 6-275 The return value of META_NVRAM_GetRecFieldBitValue

Return value	Description
META_SUCCESS	Success
META_LID_INVALID	Invalid LID.
META_BUFFER_LEN	The length of buffer is not enough.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-276 The parameter of META_NVRAM_GetRecFieldBitValue

Parameter	IN/OUT	Description
LID	IN	The name of logical data item ID

field	IN	Field name.
		If the field is an array, you can only use [n] to get the value of element in the array.
		
		For example:
		struct TEST {
		  int a[10];
		};
		
		You can use MEAT_NVRAM_SetRecFieldValue(…, "a[2]", ….);

Parameter	IN/OUT	Description
		to get the 3rd element,
field	IN	Bit field name.
buf	IN	Buffer that holds the content of a specific record.
buf_len	IN	Length of buf.
bitvalue	IN/OUT	Pointer to a location in which the bit value is located.

6.7.23 META_NVRAM_QueryIsLIDExist

Definition:

META_RESULT __stdcall META_NVRAM_QueryIsLIDExist(const char *LID)

Description:

This function is used to query whether if the LID does exist in NVRAM database.

Return Value:

Table 6-277 The return value of META_NVRAM_QueryIsLIDExist

Return value	Description
META_SUCCESS	Success, the LID does exist.
META_LID_INVALID	Invalid LID.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-278 The parameter of META_NVRAM_QueryIsLIDExist

Parameter	IN/OUT	Description
LID	IN	The name of logical data item ID

6.7.24 META_NVRAM_ResetToFactoryDefault

Definition:

```
META_RESULT  __stdcall META_NVRAM_ResetToFactoryDefault(unsigned int ms_timeout)
```

Description:

This function resets NVRAM data to factory default.

Return Value:

Table 6-279 The return value of META_NVRAM_ResetToFactoryDefault

Return value	Description
META_SUCCESS	Success, all NVRAM data that have NVRAM_CATEGORY_FACTORY attribute are reset to default value.
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-280 The parameter of META_NVRAM_ResetToFactoryDefault

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

6.7.25 META_NVRAM_AudioParam_Len

Definition:

```
META_RESULT __stdcall META_NVRAM_AudioParam_Len (int *len)
```

Description:

This function returns the size of audio parameter data.

Return Value:

Table 6-281 The return value of META_NVRAM_ResetToFactoryDefault

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-282 The parameter of META_NVRAM_ResetToFactoryDefault

Parameter	IN/OUT	Description
len	OUT	Size of audio parameter data

6.7.26 META_NVRAM_Compose_AudioParam

Definition:

META_RESULT __stdcall META_NVRAM_Compose_AudioParam(const l1audio_param_T *param, char *buf, const int buf_len)

```
// Speech Coefficient
typedef struct {
        short           Speech_8k_Input_Coeff[30];
        // FIR for input speech (microphone) with 8k sampling rate
        short           Speech_8k_Output_Coeff[30];
        // FIR for output speech (speaker) with 8k sampling rate
        short           Speech_16k_Input_Coeff[62];
        // FIR for input speech (microphone) with 16k sampling rate
        short           Speech_16k_Output_Coeff[62];
        // FIR for output speech (speaker) with 16k sampling rate

        short           Additional_Speech_8k_Output_Coeff[5][30];
        // The additional FIR for output speech (speaker) with 8k sampling rate
```

```
            unsigned short      Speech_8k_Output_Coeff_Index;
            // The active FIR index
            // 0 -> Speech_8k_Output_Coeff
            // 1 -> Additional_Speech_8k_Output_Coeff[0]
            // 2 -> Additional_Speech_8k_Output_Coeff[1]
            // 3 -> Additional_Speech_8k_Output_Coeff[2]
            // 4 -> Additional_Speech_8k_Output_Coeff[3]
            // 5 -> Additional_Speech_8k_Output_Coeff[4]
}L1_SpeechCoeff_T;

// Melody Coefficient
typedef struct {
            short                   Melody_32k_Output_Coeff[45];
}L1_MelodyCoeff_T;

// L1Audio Param
typedef struct{
            L1_SpeechCoeff_T            Speech_FIR[2];
            L1_MelodyCoeff_T            Melody_FIR[2];
            unsigned short              ES_TimeConst;
            unsigned short              ES_VolConst;
            unsigned short              ES_TimeConst2;
            unsigned short              ES_VolConst2;
            unsigned short              Media_Playback_Maximum_Swing;
}l1audio_param_T;
```

Description:

Compose audio parameter data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-283 The return value of META_NVRAM_Compose_AudioParam

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-284 The parameter of META_NVRAM_Compose_AudioParam

Parameter	IN/OUT	Description
param->Speech_FIR[0]	IN	Normal mode speech coefficient.
param->Speech_FIR[1]	IN	Headset mode speech coefficient. NOTE: In headset mode, the Additional_Speech_8k_Output_Coeff and Speech_8k_Output_Coeff_Index are ignored! You can just leave them alone.
param->Melody_FIR[0]	IN	Loudspeaker mode melody coefficient.
param->Melody_FIR[1]	IN	Stereo speaker mode melody coefficient.
param->ES_TimeConst	IN	Time const value.
param->ES_VolConst	IN	Volume const value.
param->ES_TimeConst2	IN	Time const 2 value.
param->ES_VolConst2	IN	Volume const 2 value.
param-> Media_Playback_Maximum_Swing	IN	Media playback maximum swing.
buf	IN/OUT	Output buffer to be composed.
buf_len	IN	Buffer length

6.7.27 META_NVRAM_Decompose_AudioParam

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AudioParam(l1audio_param_T *param, const char *buf, const int buf_len)

Description:

Decompose audio parameter data. Usually, once the buffer of audio coefficient data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1audio_param_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-285 The return value of META_NVRAM_Decompose_AudioParam

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-286 The parameter of META_NVRAM_Decompose_AudioParam

Parameter	IN/OUT	Description
param	IN/OUT	Output audio parameter data
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.7.28 META_NVRAM_Calculate_IMEI_CD

Definition:

META_RESULT __stdcall META_NVRAM_Calculate_IMEI_CD(const char *imei, unsigned short *p_cd)

Description:

This function will calculate Check Digit (15th digit of IMEI) by the given 14 IMEI digits.

Return Value:

Table 6-287 The return value of META_NVRAM_Calculate_IMEI_CD

Return value	Description
META_SUCCESS	Success
META_INVALID_ARGUMENTS	Invalid input arguments. It's possible the IMEI format is incorrect.

Parameter:

Table 6-288 The parameter of META_NVRAM_Calculate_IMEI_CD

Parameter	IN/OUT	Description
imei	IN	The IMEI string that contains 14 digits.
p_cd	OUT	The pointer to the short integer that will be used to store the CD (Check Digit) result. Note: The CD value will be stored in integer, not the character.

6.7.29 META_NVRAM_IMEISV_Len

Definition:

META_RESULT __stdcall META_NVRAM_IMEISV_Len(int *len)

Description:

This function returns the size of IMEISV data.

Return Value:

Table 6-289 The return value of META_NVRAM_IMEISV_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-290 The parameter of META_NVRAM_IMEISV_Len

Parameter	IN/OUT	Description
Len	OUT	Size of IMEISV data

6.7.30 META_NVRAM_Compose_IMEISV_NoCheck

Definition:

META_RESULT __stdcall META_NVRAM_Compose_IMEISV_NoCheck(const IMEISV_struct_T *p_imeisv, char *buf, const int buf_len)

```
typedef struct {
        char            imei[16];          // NULL terminated IMEI string that contains 14 bytes IMEI digit characters.
        unsigned char   svn;
        unsigned char   pad;
} IMEISV_struct_T;
```

Description:

Compose IMEISV data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.
Call META_NVRAM_Compose_IMEISV_ex(p_imeisv,buf,buf_len,false)

Remake:

If you input 14 digits of IMEI string, the Check Digit (15th digit of IMEI) will be automatically calculated and stored into buffer.

If you input 15 digits of IMEI string, it won't Check Digit correctness (15th digit of IMEI), this function will not verify Check Digit.

Return Value:

Table 6-291 The return value of META_NVRAM_Compose_IMEISV_NoCheck

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.
META_IMEI_CD_ERROR	The input Check Digit is incorrect.

Parameter:

Table 6-292 The parameter of META_NVRAM_Compose_IMEISV_NoCheck

Parameter	IN/OUT	Description
p_imeisv	IN	IMEISV data
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.7.31 META_NVRAM_Compose_IMEISV

Definition:

META_RESULT __stdcall META_NVRAM_Compose_IMEISV(const IMEISV_struct_T *p_imeisv, char *buf, const int buf_len)

typedef struct {

char imei[16]; // NULL terminated IMEI string that contains 14 bytes IMEI digit characters.

```
        unsigned char       svn;
        unsigned char       pad;
} IMEISV_struct_T;
```

Description:

Compose IMEISV data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.
Call META_NVRAM_Compose_IMEISV_ex(p_imeisv,buf,buf_len,true)

Remake:

If you input 14 digits of IMEI string, the Check Digit (15th digit of IMEI) will be automatically calculated and stored into buffer.

If you input 15 digits of IMEI string, which means you already have the correct Check Digit (15th digit of IMEI), this function will verify Check Digit, if it’s incorrect, the function will return fail.

Return Value:

Table 6-293 The return value of META_NVRAM_Compose_IMEISV

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.
META_IMEI_CD_ERROR	The input Check Digit is incorrect.

Parameter:

Table 6-294 The parameter of META_NVRAM_Compose_IMEISV

Parameter	IN/OUT	Description
p_imeisv	IN	IMEISV data
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.7.32 META_NVRAM_Decompose_IMEISV

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_IMEISV(IMEISV_struct_T *p_imeisv, const char *buf, const int buf_len)

Description:

Decompose IMEISV data. Usually, once the buffer of IMEISV data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure IMEISV_struct_T, and doesn’t take care the byte alignment problem.

Return Value:

Table 6-295 The return value of META_NVRAM_Decompose_IMEISV

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-296 The parameter of META_NVRAM_Decompose_IMEISV

Parameter	IN/OUT	Description
p_imeisv	IN/OUT	IMEISV data
Buf	IN	Buffer
buf_len	IN	Size of buf

6.7.33 META_NVRAM_SWC_RetrieveChangeList

Definition:

META_RESULT __stdcall META_NVRAM_SWC_RetrieveChangeList( )

Description:

This function is used to retrieve NVRAM LID change list from target. If you download a new load to target, NVRAM task will upgrade all the NVRAM LIDs that have different version number at the 1st time boot up. After LID upgrade process is done, NVRAM task will generate a change log to indicate which LID had been upgraded with new default value.

Return Value:

Table 6-297 The return value of META_NVRAM_SWC_RetrieveChangeList

Return value	Description
META_SUCCESS	Success
META_FAILED	General fail, please see debug log for more information.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.
META_FILE_BAD	Can't open change log from target side.
META_NO_MEMORY	Not enough memory.

6.7.34 META_NVRAM_SWC_UpdateChangeList

Definition:

META_RESULT __stdcall META_NVRAM_SWC_UpdateChangeList( )

Description:

This function is used to update PC side NVRAM LID change list to target. **NVRAM_SWC_RetrieveChangeList** will store a change list in META DLL, if you issue **META_NVRAM_Write** for the LID which is in PC side change list, META DLL will assume you already update that LID, then META_DLL will remove that LID from PC side change list. At the end, the change list between PC side and target side might be different, so you have to issue **META_NVRAM_SWC_UpdateChangeList** to sync PC side change list to target.

Return Value:

Table 6-298 The return value of META_NVRAM_SWC_UpdateChangeList

Return value	Description
META_SUCCESS	Success
META_FAILED	General fail, please see debug log for more information.
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.
META_FILE_BAD	Can't write change log to target side.
META_NO_MEMORY	Not enough memory.

Parameter:

Table 6-299 The parameter of META_NVRAM_SWC_UpdateChangeList

Parameter	IN/OUT	Description
N/A		

6.7.35 META_NVRAM_SWC_GetAllChangedLIDCount

Definition:

META_RESULT __stdcall META_NVRAM_SWC_GetAllChangedLIDCount(int *NofLID)

Description:

This function is used to query how many LIDs in change list.

Return Value:

Table 6-300 The return value of META_NVRAM_SWC_GetAllChangedLIDCount

Return value	Description
META_SUCCESS	Success
META_FAILED	You have not call NVRAM_SWC_RetrieveChangeList yet, or there is problem to get change list from target.

Return value	Description
META_INVALID_ARGUMENTS	NofLID is NULL

Parameter:

Table 6-301 The parameter of META_NVRAM_SWC_GetAllChangedLIDCount

Parameter	IN/OUT	Description
NofLID	IN/OUT	Number of LID inside change list.

6.7.36 META_NVRAM_SWC_GetAllChangedLIDName

Definition:

META_RESULT __stdcall META_NVRAM_SWC_GetAllChangedLIDName(LID_Info *p_ArrayOfLID, const int NofLID)

```
typedef struct {
    int          OldVer;       // The original version of this LID.
    int          NewVer;       // The new version of this LID.
    char         ID[64];       // The LID name string.
}LID_Info;
```

Description:

This function is used to get all the LIDs in change list.

Return Value:

Table 6-302 The return value of META_NVRAM_SWC_GetAllChangedLIDName

Return value	Description
META_SUCCESS	Success
META_FAILED	You have not call NVRAM_SWC_RetrieveChangeList yet, or there is problem to get change list from target.
META_INVALID_ARGUMENTS	p_ArrayOfLID is NULL, or NofLID is less than the change list.

Parameter:

Table 6-303 The parameter of META_NVRAM_SWC_GetAllChangedLIDName

Parameter	IN/OUT	Description
p_ArrayOfLID	IN/OUT	The array to store all LIDs inside change list.
NofLID	IN	Number of LID inside change list.

6.7.37 META_NVRAM_SWC_QueryIfLIDChanged

Definition:

```
META_RESULT  __stdcall META_NVRAM_SWC_QueryIfLIDChanged(const char *LID, LID_STATUS  *result)


typedef enum {
    LID_VER_SAME = 0,
    LID_VER_CHANGED
} LID_STATUS;
```

Description:

This function is used to query if the version of LID that you specify is changed.

Return Value:

Table 6-304 The return value of META_NVRAM_SWC_QueryIfLIDChanged

Return value	Description
META_SUCCESS	Success
META_FAILED	You have not call NVRAM_SWC_RetrieveChangeList yet, or there is problem to get change list from target.
META_INVALID_ARGUMENTS	LID is NULL, or result is NULL.
META_LID_INVALID	The input LID name is invalid, it's not found in NVRAM database.

Parameter:

Table 6-305 The parameter of META_NVRAM_SWC_QueryIfLIDChanged

Parameter	IN/OUT	Description
LID	IN	The LID name string that you want to query.
Result	IN/OUT	The status of LID. LID_VER_SAME indicates this LID doesn't change during NVRAM upgrade. LID_VER_CHANGED means this LID does change.

6.7.38 META_NVRAM_SWC_Database_Compare

Definition:

```
META_RESULT  __stdcall META_NVRAM_SWC_Database_Compare(
                                                       const char *PathName,
                                                       int *p_NumOfNewAddLID,
                                                       int *p_NumOfModifiedLID,
                                                       int *p_NumOfDeletedLID)
```

Description:

This function is used to compare current NVRAM database with the new one.

Return Value:

Table 6-306 The return value of META_NVRAM_SWC_Database_Compare

Return value	Description
META_SUCCESS	Success
META_LID_INVALID	The input LID name is invalid, it's not found in NVRAM database.
META_FILE_BAD	New NVRAM database doesn't exist or fail to open.
META_FAILED	Import new NVRAM database file failed.
META_INTERNAL_DB_ERR	Can't find structure info from NVRAM InternalDB.

Parameter:

Table 6-307 The parameter of META_NVRAM_SWC_Database_Compare

Parameter	IN/OUT	Description
PathName	IN	The full filepath of new NVRAM database.
p_NumOfNewAddLID	IN/OUT	The total count of new-added LIDs in new NVRAM database.
p_NumOfModifiedLID	IN/OUT	The total count of LIDs that have version change between new and old NVRAM database.
p_NumOfDeletedLID	IN/OUT	The total count of LIDs they are deleted in new NVRAM database.

6.7.39 META_NVRAM_SWC_Get_Database_Compare_Result

Definition:

```
META_RESULT __stdcall META_NVRAM_SWC_Get_Database_Compare_Result(
        LID_Info *p_ArrayOfNewAddLID, const int NumOfNewAddLID,
        LID_Info *p_ArrayOfModifiedLID, const int NumOfModifiedLID,
        LID_Info *p_ArrayOfDeletedLID, const int NumOfDeletedLID)
```

Description:

This function is used to get NVRAM database compare result.

Return Value:

Table 6-308 The return value of META_NVRAM_SWC_Get_Database_Compare_Result

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The prepared LID array is too small to store the result.

Parameter:

Table 6-309 The parameter of META_NVRAM_SWC_Get_Database_Compare_Result

Parameter	IN/OUT	Description
p_ArrayOfNewAddLID	IN/OUT	The array to store new-added LIDs in new NVRAM database.
NumOfNewAddLID	IN	The total count of new-added LIDs in new NVRAM database.
p_ArrayOfModifiedLID	IN/OUT	The array to store LIDs that have version change between new and old NVRAM database.
NumOfModifiedLID	IN	The total count of LIDs that have version change between new and old NVRAM database.
p_ArrayOfDeletedLID	IN/OUT	The array to store LIDs that are deleted in new NVRAM database.
NumOfDeletedLID	IN	The total count of LIDs that are deleted in new NVRAM database.

6.7.40 META_NVRAM_SWC_Check_FAT_FreeSpace

Definition:

```
META_RESULT  __stdcall META_NVRAM_SWC_Check_FAT_FreeSpace(
                                        const CB_META_NVRAM_GET_DISK_INFO_CNF  cb,
                                        short *token,
                                        void *usrData)

typedef struct {
  int              target_nvramsize;        // current NVRAM size on target FAT file system
  int              target_freespace;        // current freespace of target FAT file system
  int              target_overhead;         // S/W upgrade operation overhead
  int              newdb_nvramsize;         // new NVRAM size
  unsigned char    status;              // 0 -> [OK] safe to upgrade to new NVRAM
                                        // 1 -> [ERROR] can't retrieve info from target
                                        // 2 -> [ERROR] freespace is not enough to upgrade to new NVRAM
} NVRAM_GetDiskInfo_Cnf;
```

Description:

This function is used to query target FAT freespace information.

Callback:

```
    typedef void (__stdcall *CB_META_NVRAM_GET_DISK_INFO_CNF)(const NVRAM_GetDiskInfo_Cnf
*cnf, const short token, void *usrData);
```

Return Value:

Table 6-310 The return value of META_NVRAM_SWC_Check_FAT_FreeSpace

Return value	Description
META_SUCCESS	Success
META_LID_INVALID	The input LID name is invalid, it's not found in NVRAM database.
META_FILE_BAD	New NVRAM database doesn't exist or fail to open.
META_FAILED	Import new NVRAM database file failed.
META_INTERNAL_DB_ERR	Can't find structure info from NVRAM InternalDB.

Parameter:

Table 6-311 The parameter of META_NVRAM_SWC_Check_FAT_FreeSpace

Parameter	IN/OUT	Description
cb	IN	The callback function will be called when target send query confirmation. If you didn't call META_NVRAM_SWC_Database_Compare, cnf->newdb_nvramsize would be zero.
token	IN/OUT	Token used by user to uninstall the confirmation and indication callback function.
usrData	IN	Parameter used by user.

6.7.41 META_NVRAM_SWC_Enable_ForceUpgrade

Definition:

META_RESULT __stdcall META_NVRAM_SWC_Enable_ForceUpgrade( )

Description:

This function is used to create NVRAM forced upgrade flag on target. If the forced upgrade flag is set, NVRAM task will forced execute upgrade procedure even some important LIDs (L1 calibration data) are changed and need to be backup. Otherwise NVRAM will block system and display warning on target LCD panel.

Return Value:

Table 6-312 The return value of META_NVRAM_SWC_Enable_ForceUpgrade

Return value	Description
META_SUCCESS	Success
META_FILE_BAD	Can’t write forced upgrade flag to the FAT on target.
META_INTERNAL_DB_ERR	Can’t find structure info from NVRAM InternalDB.

Parameter:

Table 6-313 The parameter of META_NVRAM_SWC_Enable_ForceUpgrade

Parameter	IN/OUT	Description
N/A		

6.7.42 META_NVRAM_SWC_Disable_ForceUpgrade

Definition:

META_RESULT __stdcall META_NVRAM_SWC_Disable_ForceUpgrade( )

Description:

This function is used to disable NVRAM forced upgrade flag on target.

Return Value:

Table 6-314 The return value of META_NVRAM_SWC_Disable_ForceUpgrade

Return value	Description
META_SUCCESS	Success
META_FILE_BAD	Can’t write forced upgrade flag to the FAT on target.
META_INTERNAL_DB_ERR	Can’t find structure info from NVRAM InternalDB.

Parameter:

Table 6-315 The parameter of META_NVRAM_SWC_Disable_ForceUpgrade

Parameter	IN/OUT	Description
N/A		

6.7.43 META_NVRAM_Compose_AudioParam _W0712

Definition:

META_RESULT __stdcall META_NVRAM_Compose_AudioParam_W0712 (const l1audio_param_W0712_T *param, char *buf, const int buf_len)

```
typedef struct{
    short speech_input_FIR_coeffs[6][45];
  short speech_output_FIR_coeffs[6][45];
  unsigned short selected_FIR_output_index;
  unsigned short speech_common_para[12];
  unsigned short speech_normal_mode_para[8];
  unsigned short speech_earphone_mode_para[8];
  unsigned short speech_loudspk_mode_para[8];
  unsigned short speech_bt_earphone_mode_para[8];
  unsigned short speech_bt_cordless_mode_para[8];
  unsigned short speech_aux1_mode_para[8];
  unsigned short speech_aux2_mode_para[8];
  unsigned short speech_aux3_mode_para[8];
  unsigned short Media_Playback_Maximum_Swing;
  short Melody_FIR_Output_Coeff_32k_Tbl1[25];
} l1audio_param_W0712_T;
```

Description:

Compose audio parameter data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-316 The return value of META_NVRAM_Compose_AudioParam _W0712

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-317 The parameter of META_NVRAM_Compose_AudioParam _W0712

Parameter	IN/OUT	Description
param->Speech_FIR[0]	IN	Normal mode speech coefficient.
param->Speech_FIR[1]	IN	Headset mode speech coefficient. NOTE: In headset mode, the Additional_Speech_8k_Output_Coeff and Speech_8k_Output_Coeff_Index are ignored! You can just leave them alone.
param->Melody_FIR[0]	IN	Loudspeaker mode melody coefficient.
param->Melody_FIR[1]	IN	Stereo speaker mode melody coefficient.
param->ES_TimeConst	IN	Time const value.
param->ES_VolConst	IN	Volume const value.
param->ES_TimeConst2	IN	Time const 2 value.
param->ES_VolConst2	IN	Volume const 2 value.
param-> Media_Playback_Maximum_Swing	IN	Media playback maximum swing.
param-> Melody_FIR_Output_Coeff_32k_Tbl1	IN	Melody_FIR_Output_Coeff_32k
buf	IN/OUT	Output buffer to be composed.
buf_len	IN	Buffer length

6.7.44 META_NVRAM_Decompose_AudioParam _W0712

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AudioParam_W0712 (l1audio_param_T *param, const char *buf, const int buf_len)

Description:

Decompose audio parameter data. Usually, once the buffer of audio coefficient data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1audio_param_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-318 The return value of META_NVRAM_Decompose_AudioParam _W0712

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-319 The parameter of META_NVRAM_Decompose_AudioParam _W0712

Parameter	IN/OUT	Description
param	IN/OUT	Output audio parameter data
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.7.45 META_NVRAM_Compose_AudioParam_W0740

Definition:

META_RESULT __stdcall META_NVRAM_Compose_AudioParam_W0712 (const l1audio_param_W0712_T *param, char *buf, const int buf_len)

```
typedef struct
{
    short              speech_input_FIR_coeffs[6][45];
    short              speech_output_FIR_coeffs[6][45];
    unsigned short     selected_FIR_output_index;
    unsigned short     speech_common_para[12];
    unsigned short     speech_mode_para[8][8];
    unsigned short     Media_Playback_Maximum_Swing;
    short              Melody_FIR_Coeff_Tbl[25];
    short                   audio_compensation_coeff[2][45];   // new added, so different with others version

} l1audio_param_W0740_T;
```

Description:

Compose audio parameter data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-320 The return value of META_NVRAM_Compose_AudioParam_W0740

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-321 The parameter of META_NVRAM_Compose_AudioParam_W0740

Parameter	IN/OUT	Description
param	IN	l1audio_param_W0740_T
buf	IN/OUT	Output buffer to be composed.
buf_len	IN	Buffer length

6.7.46 META_NVRAM_Decompose_AudioParam_W0740

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AudioParam_W0740(l1audio_param_W0740_T *param, const char *buf, const int buf_len);

Description:

Decompose audio parameter data. Usually, once the buffer of audio coefficient data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1audio_param_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-322 The return value of META_NVRAM_Decompose_AudioParam_W0740

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-323 The parameter of META_NVRAM_Decompose_AudioParam_W0740

Parameter	IN/OUT	Description
param	IN/OUT	Output audio parameter data
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.7.47 META_NVRAM_Compose_AudioParam_W0809

Definition:

META_RESULT __stdcall META_NVRAM_Compose_AudioParam_W0809(const l1audio_param_W0809_T *param, char *buf, const int buf_len)

```
typedef struct

{

   short        speech_input_FIR_coeffs[6][45];

   short        speech_output_FIR_coeffs[6][45];

   unsigned short selected_FIR_output_index;

   unsigned short speech_common_para[8];

   unsigned short speech_mode_para[8][16];

   unsigned short speech_volume_para[3][7][4];

   unsigned short Media_Playback_Maximum_Swing;

   short        Melody_FIR_Coeff_Tbl[25];

   short        audio_compensation_coeff[2][45];  // new added, so different with others structure


} l1audio_param_W0809_T;
```

Description:

Compose audio parameter data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-324 The return value of META_NVRAM_Compose_AudioParam_W0809

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-325 The parameter of META_NVRAM_Compose_AudioParam_W0809

Parameter	IN/OUT	Description
param	IN	l1audio_param_W0809_T
buf	IN/OUT	Output buffer to be composed.
buf_len	IN	Buffer length

6.7.48 META_NVRAM_Decompose_AudioParam_W0809

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AudioParam_W0748(l1audio_param_W0809_T *param, const char *buf, const int buf_len)

Description:

Decompose audio parameter data. Usually, once the buffer of audio coefficient data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1audio_param_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-326 The return value of META_NVRAM_Decompose_AudioParam_W0809

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-327 The parameter of META_NVRAM_Decompose_AudioParam_W0809

Parameter	IN/OUT	Description
param	IN/OUT	Output audio parameter data
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.7.49 META_NVRAM_TRIM_THERMO_Len

Definition:

META_RESULT __stdcall META_NVRAM_TRIM_THERMO_Len(int *len);

Description:

This function returns the size of wndrv_cal_setting_trim_thermo_struct table.

Return Value:

Table 6-328 The return value of META_NVRAM_TRIM_THERMO_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-329 The parameter of META_NVRAM_TRIM_THERMO_Len

Parameter	IN/OUT	Description
Len	OUT	Size of wndrv_cal_setting_trim_thermo_struct table

6.7.50 META_NVRAM_WiFi_Compose_TrimThermo

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_Compose_TrimThermo(const wndrv_cal_setting_trim_thermo_struct *trim_struct, char *buf, const int buf_len);

```
typedef struct // LID: NVRAM_EF_WNDRV_EXT_SETTING_TRIMVAL_THERMOVAL_LID
{
  char          cAbsTemp;
  unsigned char  ucThermoValue;
  unsigned char  ucXtalTrim;
}wndrv_cal_setting_trim_thermo_struct;
```

Description:

Compose WiFi MT5921 thermo data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-330 The return value of META_NVRAM_WiFi_Compose_TrimThermo

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-331 The parameter of META_NVRAM_WiFi_Compose_TrimThermo

Parameter	IN/OUT	Description
adtx	IN	trim_struct
buf	IN/OUT	Output buffer to be composed.
buf_len	IN	Buffer length

6.7.51 META_NVRAM_WiFi_Decompose_TrimThermo

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_Decompose_TrimThermo(wndrv_cal_setting_trim_thermo_struct *trim_struct, const char *buf, const int buf_len);

Description:

Decompose WiFi MT5921 thermo data. Usually, once the buffer of structure data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure wndrv_cal_setting_trim_thermo_struct, and doesn’t take care the byte alignment problem.

Return Value:

Table 6-332 The return value of META_NVRAM_WiFi_Decompose_TrimThermo

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-333 The parameter of META_NVRAM_WiFi_Decompose_TrimThermo

Parameter	IN/OUT	Description
adtx	IN/OUT	Output WiFi MT5921 thermo data.
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.7.52 META_NVRAM_PortSetting_Len

Definition:

META_RESULT __stdcall META_NVRAM_PortSetting_Len(int *len);

Description:

This function returns the size of port_setting_struct.

Return Value:

Table 6-334 The return value of META_NVRAM_PortSetting_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-335 The parameter of META_NVRAM_PortSetting_Len

Parameter	IN/OUT	Description
Len	OUT	Size of port_setting_struct.

6.7.53 META_NVRAM_Compose_PortSetting

Definition:

```
META_RESULT __stdcall META_NVRAM_Compose_PortSetting(const port_setting_struct *port_setting, char *buf, const int buf_len);
```

```
typedef struct
{
   unsigned short tst_port_ps;
   unsigned short      ps_port;
   unsigned int     tst_baudrate_ps;
   unsigned int     ps_baudrate;
   bool        High_Speed_SIM_Enabled;
   unsigned char       swdbg;
   unsigned char       uart_power_setting;
   unsigned char       cti_uart_port;
   unsigned int     cti_baudrate;
   unsigned char        tst_port_l1;
   unsigned int      tst_baudrate_l1;
   // Support tst output to memory card
   unsigned char       tst_output_mode;
   unsigned char  usb_logging_mode;
}                                                              port_setting_struct;
```

Description:

Compose UART Port Setting data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-336 The return value of META_NVRAM_Compose_PortSetting

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-337 The parameter of META_NVRAM_Compose_PortSetting

Parameter	IN/OUT	Description
port_setting	IN	UART Port Setting.
buf	IN/OUT	Output buffer to be composed.
buf_len	IN	Buffer length

6.7.54 META_NVRAM_Decompose_PortSetting

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_PortSetting(port_setting_struct *port_setting, const char *buf, const int buf_len);

Description:

Decompose UART Port Setting data. Usually, once the buffer of structure data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure port_setting_struct, and doesn't take care the byte alignment problem.

Return Value:

Table 6-338 The return value of META_NVRAM_Decompose_PortSetting

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-339 The parameter of META_NVRAM_Decompose_PortSetting

Parameter	IN/OUT	Description
port_setting	IN/OUT	Output UART Port Setting.
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.7.55 META_NVRAM_SetCallback

Definition:

typedef int (__stdcall *CB_META_NVRAM_GET_REMOTE_KEY_LENGTH)(unsigned int * const length, void *usrData);

typedef int (__stdcall *CB_META_NVRAM_GET_REMOTE_KEY)(char* const key, unsigned int key_length, void *usrData);

typedef int (__stdcall *CB_META_NVRAM_GET_REMOTE_DATABASE_LENGTH)(unsigned int * const length, void *usrData);

typedef int (__stdcall *CB_META_NVRAM_GET_REMOTE_DATABASE)(char* const database, unsigned int database_length, void *usrData);

```
META_RESULT  __stdcall META_NVRAM_SetCallback(
    CB_META_NVRAM_GET_REMOTE_KEY_LENGTH      getKeyLength,      void* getKeyLengthArgument,
    CB_META_NVRAM_GET_REMOTE_KEY             getKey,            void* getKeyArgument,
    CB_META_NVRAM_GET_REMOTE_DATABASE_LENGTH getDatabaseLength, void* getDatabaseLengthArgument,
    CB_META_NVRAM_GET_REMOTE_DATABASE        getDatabase,       void* getDatabaseArgument
    );
```

Description:

The API set the callback function for META mode NVRAM access authentication. The callback function is used to retrieve the remote key or remote database.
The user can overwrite the remote database loading or remote key loading by themselves by register valid callback function to get remote database.

CB_META_NVRAM_GET_REMOTE_KEY_LENGTH is used to get the remote key length so that META DLL will allocate the key buffer with length determined in this callback and later used in CB_META_NVRAM_GET_REMOTE_KEY.

CB_META_NVRAM_GET_REMOTE_KEY is used to get the remote key and put into the buffer that META DLL allocated.

CB_META_NVRAM_GET_REMOTE_DATABASE_LENGTH is used to get the remote database size so that META DLL will alloate the database buffer with the length determined in this callback and later used in ***CB_META_NVRAM_GET_REMOTE_DATABASE***.

CB_META_NVRAM_GET_REMOTE_DATABASE_LENGTH is used to get the remote database and put into the bufer that META DLL allocated.

Return Value:

Table 6-340 The return value of META_NVRAM_SetCallback

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-341 The parameter of META_NVRAM_SetCallback

Parameter	IN/OUT	Description
port_setting	IN/OUT	Output UART Port Setting.
getKeyLength	IN	callback function to determine the remote key length.
getKeyLengthArgument	IN	user argument for the getKeyLength callback
getKey	IN	callback function to get the remote key into buffer.
getKeyArgument	IN	user argument for the getKey callback
getDatabaseLength	IN	callback function to determine the remote database size.
getDatabaseLengthArgument	IN	user argument for getDatabaseLength
getDatabase	IN	callback function to get the remote database into buffer.
getDatabaseArgument	IN	user argument for getDatabase

Sample Code:

```
/**
 * Remote get key length user callback function implementation sample
 * user has to set the key length
 * META DLL will then allocate the key buffer by the length set by user
 */
int __stdcall remote_key_length(unsigned int * const length, void *usrData)
{
   AfxMessageBox("remote get key length");
   *length = 64;
   return 0;
}
```

```
/**
 * Remote get key user callback function implementation sample
 * the user has to copy the key content to the data buffer supplied by META DLL
 */
int __stdcall remote_key(char* const key, unsigned int key_length, void *usrData)
{
    AfxMessageBox("remote get key");
    for(int i=0;i<key_length;i++)
    {
        key[i] = i;
    }
    return 0;
}
/**
 * Remote get database size user callback function implementation sample
 * user has to set the database size
 * META DLL will then allocate the database buffer by the size set by user
 */
int __stdcall remote_db_length(unsigned int * const length, void *usrData)
{
    AfxMessageBox("remote get db length");
    try
    {
        ifstream ifs;
        ifs.open("Z:\\db", ios::binary|ios::in);
        ifs.seekg(0, std::ios::end);
        *length = ifs.tellg();
    }
    catch(...)
    {
        *length = 0;
    }
    return 0;
}
/**
 * Remote get database user callback function implementation sample
 * the user has to copy the database content to the data buffer supplied by META DLL
 */
int __stdcall remote_db(char* const database, unsigned int database_length, void *usrData)
{
    AfxMessageBox("remote get db");
    try
    {
        ifstream is;
        is.open ("Z:\\db", ios::binary );
        // read data as a block:
        is.read (database, database_length);
        is.close();
    }
    catch(...)
    {
        AfxMessageBox("remote get db failed");
    }
    return 0;
}
/**
 * Example1: If the user wants to load the database or key on remote server
 * The user has to set the callback function before doing NVRAM init
 */
static bool __stdcall remote_load_db_key(void)
{
  /*****************************************************
   * both key and database are loaded on remote server
   *****************************************************/
  unsigned long addr;
  if(META_SUCCESS != META_NVRAM_SetCallback(
        remote_key_length, NULL,
        remote_key,       NULL,
        remote_db_length,  NULL,
        remote_db,        NULL))
  {
    AfxMessageBox("set callback failed");
  }
  if(META_SUCCESS != META_NVRAM_Init_r(0, "Z:\\db", &addr))
  {
    AfxMessageBox("init NVRAM failed");
  }
  return true;
}
/**
 * Example2: If the user wants to load the database or key on remote server
 * The user has to set the callback function before doing NVRAM init
 */
static bool __stdcall remote_load_key(void)
{
    /*****************************************************
     * key is loaded on remote server
     ****************************************************/
    unsigned long addr;
    if(META_SUCCESS != META_NVRAM_SetCallback(
            remote_key_length, NULL,
            remote_key,       NULL,
            NULL,  NULL,
            NULL,        NULL))
    {
      AfxMessageBox("set callback failed");
    }
    if(META_SUCCESS != META_NVRAM_Init_r(0, "Z:\\db", &addr))
    {
      AfxMessageBox("init NVRAM failed");
    }
    return true;
  }
  /**
   * Example3: If the user wants to load the database or key on remote server
   * The user has to set the callback function before doing NVRAM init
   */
  static bool __stdcall remote_load_db(void)
  {
    /*****************************************************
     * key is loaded on remote server
     ****************************************************/
    unsigned long addr;
    if(META_SUCCESS != META_NVRAM_SetCallback(
            NULL, NULL,
            NULL,       NULL,
            remote_db_length,  NULL,
            remote_db,        NULL))
   {
      AfxMessageBox("set callback failed");
   }
   if(META_SUCCESS != META_NVRAM_Init_r(0, "Z:\\db", &addr))
   {
      AfxMessageBox("init NVRAM failed");
   }
   return true;
}
```

6.7.56 META_NVRAM_QueryRecField

Definition:

META_RESULT __stdcall META_NVRAM_QueryRecField(const char *LID, const char *field, unsigned int* fieldSize, unsigned int* fieldOffset);

Description:

The API query the size and offset of a member field in a NVRAM LID.

Return Value:

Table 6-342 The return value of META_NVRAM_QueryRecField

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-343 The parameter of META_NVRAM_QueryRecField

Parameter	IN/OUT	Description
LID	IN	The NVRAM LID name to be queried.
field	IN	The member field name to be queried.
fieldSize	OUT	The size of the mbmer field.
fieldOffset	OUT	The offset of the member field in the buffer. (unit: byte)

Sample Code:

```
    // init database…
    unsigned long nvram_idb;
    META_NVRAM_Init_r(meta_handle, "c:\\db_file", &nvram_idb);
    // query a member field ("agcPathLoss[0][0].max_arfcn" ) in "NVRAM_EF_L1_AGCPATHLOSS_LID",
dereferencing the member field directly in the field parameter
    unsigned int fieldSize;
    unsigned int fieldOffset;
    if(META_SUCCESS == META_NVRAM_QueryRecField("NVRAM_EF_L1_AGCPATHLOSS_LID",
"agcPathLoss[0][0].max_arfcn", &fieldSize, &fieldOffset))
    {
        printf("fieldSize(%d), fieldOffset(%d)", fieldSize, fieldOffset);
    }
```

Use Case: Get / Set member field data from / to the NVRAM buffer

```
    unsigned int fieldSize;
    unsigned int fieldOffset;
    unsigned char* fieldBuffer;
    unsigned char* nv_buffer;
    int nv_length;
    if(META_SUCCESS == META_NVRAM_GetRecLen("NVRAM_EF_L1_AGCPATHLOSS_LID", &nv_length))
    {
        nv_buffer = new unsigned char[nv_length];
        FT_NVRAM_WRITE_REQ write_req;
        FT_NVRAM_WRITE_CNF write_cnf;
        FT_NVRAM_READ_REQ read_req;
```

```
FT_NVRAM_READ_CNF read_cnf;
read_req.LID = "NVRAM_EF_L1_AGCPATHLOSS_LID";
read_req.RID = 1;
read_cnf.buf = nv_buffer;
read_cnf.len = nv_length;
write_req.LID = "NVRAM_EF_L1_AGCPATHLOSS_LID";
write_req.RID = 1;
write_req.buf = nv_buffer;
write_req.len = nv_length;
if(META_SUCCESS == META_NVRAM_Read_Ex_r(0, 3000, &read_req, &read_cnf))
{
  if(META_SUCCESS == META_NVRAM_QueryRecField("NVRAM_EF_L1_AGCPATHLOSS_LID",
"agcPathLoss[1][1].max_arfcn", &fieldSize, &fieldOffset))
  {
    fieldBuffer = new unsigned char[fieldSize];
    META_NVRAM_GetRecFieldValue(
      "NVRAM_EF_L1_AGCPATHLOSS_LID",
      "agcPathLoss[1][1].max_arfcn",
      (char*)nv_buffer,
      nv_length,
      fieldBuffer,
      fieldSize);
    short* s_p = (short*) fieldBuffer;
    (*s_p)++;
    META_NVRAM_SetRecFieldValue(
      "NVRAM_EF_L1_AGCPATHLOSS_LID",
      "agcPathLoss[1][1].max_arfcn",
      (char*)nv_buffer,
      nv_length,
                fieldBuffer,
                fieldSize);
            META_NVRAM_Write_Ex_r(0, 3000, &write_req, &write_cnf);
            delete [] fieldBuffer;
          }
        }
        delete [] nv_buffer;
    }
```

6.8 Audio related NVRAM buffer operations

6.8.1 META_NVRAM_CustAcousticVol_Len

Definition:

META_RESULT __stdcall META_NVRAM_CustAcousticVol_Len (int *len)

Description:

This function returns the size of custom acoustic volume gain data.

Return Value:

Table 6-344 The return value of META_NVRAM_CustAcousticVol_Len

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-345 The parameter of META_NVRAM_CustAcousticVol_Len

Parameter	IN/OUT	Description
Len	OUT	Size of custom acoustic volume gain data

6.8.2 META_NVRAM_Compose_CustAcousticVol

Definition:

```
META_RESULT __stdcall META_NVRAM_Compose_CustAcousticVol(const CustAcousticVol_T *cust_acoustic_vol, char *buf, const int buf_len)
```

```
// Custom Acoustic Volume
#define MAX_VOL_CATE        3
#define MAX_VOL_TYPE        7
#define MAX_VOL_LEVEL  7

typedef struct {
        unsigned char    volume_gain[MAX_VOL_CATE][MAX_VOL_TYPE][MAX_VOL_LEVEL];
        unsigned char    volume[MAX_VOL_CATE][MAX_VOL_TYPE];
} CustAcousticVol_T;
```

Description:

Compose custom volume gain table data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-346 The return value of META_NVRAM_Compose_CustAcousticVol

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-347 The parameter of META_NVRAM_Compose_CustAcousticVol

Parameter	IN/OUT	Description
cust_acoustic_vol->volume_gain	IN	Custom volume gain table. The 1st dimension MAX_VOL_CATE is mode category: 0 → Normal Mode. 1 → Headset Mode. 2 → Loudspeaker Mode.

Parameter	IN/OUT	Description
		The 2nd dimension MAX_VOL_TYPE is volume type: 0 → Call Tone. 1 → Key Tone. 2 → MIC. 3 → GMI. 4 → Speech. 5 → Side Tone. 6 → Melody. The 3rd dimension MAX_VOL_LEVEL is volume gain value for 7 levels: 0 → Level 1 volume gain value. 1 → Level 2 volume gain value. 2 → Level 3 volume gain value. 3 → Level 4 volume gain value. 4 → Level 5 volume gain value. 5 → Level 6 volume gain value. 6 → Level 7 volume gain value. The gain value is allowed from 0~255.
cust_acoustic_vol->volume	IN	Current volume gain index. The 1st dimension MAX_VOL_CATE is mode category: 0 → Normal Mode. 1 → Headset Mode. 2 → Loudspeaker Mode. The 2nd dimension MAX_VOL_TYPE is volume type: 0 → Call Tone. 1 → Key Tone. 2 → MIC. 3 → GMI. 4 → Speech. 5 → Side Tone. 6 → Melody. The volume level value is allowed from 0~6. (Level 1 ~ Level 7)
buf	IN/OUT	Output buffer to be composed.
buf_len	IN	Buffer length

6.8.3 META_NVRAM_Decompose_CustAcousticVol

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_CustAcousticVol(
CustAcousticVol_T *cust_acoustic_vol,
const char *buf, const int buf_len)

Description:

Decompose custom volume gain data. Usually, once the buffer of audio coefficient data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure CustAcousticVol_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-348 The return value of META_NVRAM_Decompose_CustAcousticVol

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-349 The parameter of META_NVRAM_Decompose_CustAcousticVol

Parameter	IN/OUT	Description
cust_acoustic_vol	IN/OUT	Output custom volume gain data
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.8.4 META_NVRAM_AudioBesLoudNess_Len

Definition:

META_RESULT __stdcall META_NVRAM_AudioBesLoudNess_Len(int *len);

Description:

Get the structure size of nvram_ef_audio_besloudness_struct.

Return Value:

Table 6-350 The return value of META_NVRAM_AudioBesLoudNess_Len

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-351 The parameter of META_NVRAM_AudioBesLoudNess_Len

Parameter	IN/OUT	Description
len	OUT	Size of buf

6.8.5 META_NVRAM_Compose_AudioBesLoudNess

Definition:

```
META_RESULT __stdcall META_NVRAM_Compose_AudioBesLoudNess(const l1audio_besloudness_T
*param, char *buf, const int buf_len);

typedef struct
{
   unsigned int hsf_coeffs[9][4];
   unsigned int bpf_coeffs[4][6][3];
   /// BesLoudness V3
   unsigned int audio_besloudness_DRC_Forget_Table[9][2];
   unsigned int audio_besloudness_WS_Gain_Max;
   unsigned int audio_besloudness_WS_Gain_Min;
   unsigned int audio_besloudness_Filter_First;
   char        audio_besloudness_Gain_Map_In[5];
   char        audio_besloudness_Gain_Map_Out[5];
} l1audio_besloudness_T;
```

Description:

Compose the param into NVRAM structure "nvram_ef_audio_besloudness_struct".

Return Value:

Table 6-352 The return value of META_NVRAM_Compose_AudioBesLoudNess

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-353 The parameter of META_NVRAM_Compose_AudioBesLoudNess

Parameter	IN/OUT	Description
param	IN	The super set of audio BesLoudness including V1 to V3.
buf	IN/OUT	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.6 META_NVRAM_Decompose_AudioBesLoudNess

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AudioBesLoudNess(l1audio_besloudness_T *param, const char *buf, const int buf_len);

```
typedef struct
{
   unsigned int hsf_coeffs[9][4];
   unsigned int bpf_coeffs[4][6][3];
   /// BesLoudness V3
   unsigned int audio_besloudness_DRC_Forget_Table[9][2];
   unsigned int audio_besloudness_WS_Gain_Max;
   unsigned int audio_besloudness_WS_Gain_Min;
   unsigned int audio_besloudness_Filter_First;
   char       audio_besloudness_Gain_Map_In[5];
   char       audio_besloudness_Gain_Map_Out[5];
} l1audio_besloudness_T;
```

Description:

Deompose the buf into the META_DLL PC size structure "l1audio_besloudness_T"

Return Value:

Table 6-354 The return value of META_NVRAM_Decompose_AudioBesLoudNess

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-355 The parameter of META_NVRAM_Decompose_AudioBesLoudNess

Parameter	IN/OUT	Description
param	IN/OUT	The super set of audio BesLoudness including V1 to V3.
buf	IN	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.7 META_NVRAM_Compose_AudioFIRParam_WB

Definition:

META_RESULT __stdcall META_NVRAM_Compose_AudioFIRParam_WB(const l1audio_wb_speech_fir_struct *param, char *buf, const int buf_len);

```
typedef struct
{
   short coeff[6][90];
}l1audio_wb_speech_fir_struct;
```

Description:

Compose the buf into the META_DLL PC size structure “l1audio_wb_speech_fir_struct”

Return Value:

Table 6-356 The return value of META_NVRAM_Compose_AudioFIRParam_WB

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-357 The parameter of META_NVRAM_Compose_AudioFIRParam_WB

Parameter	IN/OUT	Description
param	IN/OUT	The parameter for buffer composing
buf	IN	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.8 META_NVRAM_Decompose_AudioFIRParam_WB

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AudioFIRParam_WB(l1audio_wb_speech_fir_struct *param, const char *buf, const int buf_len);

```
typedef struct
{
   short coeff[6][90];
}l1audio_wb_speech_fir_struct;
```

Description:

Decompose the buf into the META_DLL PC size structure “l1audio_wb_speech_fir_struct”

Return Value:

Table 6-358 The return value of META_NVRAM_Decompose_AudioFIRParam_WB

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-359 The parameter of META_NVRAM_Decompose_AudioFIRParam_WB

Parameter	IN/OUT	Description
param	IN	The parameter for buffer decomposing
buf	IN/OUT	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.9 META_NVRAM_Compose_AudioSpeechParam_WB

Definition:

META_RESULT __stdcall META_NVRAM_Compose_AudioSpeechParam_WB(const l1audio_wb_speech_mode_struct *param, char *buf, const int buf_len);

```
typedef struct
{
   short param[8][16];
}l1audio_wb_speech_mode_struct;
```

Description:

Compose the buf into the META_DLL PC size structure "l1audio_wb_speech_mode_struct"

Return Value:

Table 6-360 The return value of META_NVRAM_Compose_AudioSpeechParam_WB

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-361 The parameter of META_NVRAM_Compose_AudioSpeechParam_WB

Parameter	IN/OUT	Description
param	IN/OUT	The parameter for buffer composing
buf	IN	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.10 META_NVRAM_Decompose_AudioSpeechParam_WB

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AudioSpeechParam_WB(l1audio_wb_speech_mode_struct *param, const char *buf, const int buf_len);

```
typedef struct
{
   short param[8][16];
}l1audio_wb_speech_mode_struct;
```

Description:

Decompose the buf into the META_DLL PC size structure "l1audio_wb_speech_mode_struct"

Return Value:

Table 6-362 The return value of META_NVRAM_Decompose_AudioSpeechParam_WB

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-363 The parameter of META_NVRAM_Decompose_AudioSpeechParam_WB

Parameter	IN/OUT	Description
param	IN	The parameter for buffer decomposing
buf	IN/OUT	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.11 META_NVRAM_Compose_AudioParam_EX2

Definition:

META_RESULT __stdcall META_NVRAM_Compose_AudioParam_EX2(const l1audio_param_EX2_T *param, char *buf, const int buf_len);

```
typedef struct
{
  short        speech_input_FIR_coeffs[6][45];
  short        speech_output_FIR_coeffs[6][45];
  unsigned short selected_FIR_output_index;
  unsigned short speech_common_para[12];
  unsigned short speech_mode_para[8][16];
  unsigned short speech_volume_para[3][7][4];
  unsigned short Media_Playback_Maximum_Swing;
  short        Melody_FIR_Coeff_Tbl[25];
  short        audio_compensation_coeff[3][45];
  L1_audio_abf_param_struct_T  abf_param;
} l1audio_param_EX2_T;
```

Description:

Compose the buf into the META_DLL PC size structure "l1audio_param_EX2_T"

Return Value:

Table 6-364 The return value of META_NVRAM_Compose_AudioParam_EX2

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-365 The parameter of META_NVRAM_Compose_AudioParam_EX2

Parameter	IN/OUT	Description
param	IN/OUT	The parameter for buffer composing
buf	IN	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.12 META_NVRAM_Decompose_AudioParam_EX2

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AudioParam_EX2(l1audio_param_EX2_T *param, const char *buf, const int buf_len);

```
typedef struct
{
  short       speech_input_FIR_coeffs[6][45];
  short       speech_output_FIR_coeffs[6][45];
  unsigned short selected_FIR_output_index;
  unsigned short speech_common_para[12];
  unsigned short speech_mode_para[8][16];
  unsigned short speech_volume_para[3][7][4];
  unsigned short Media_Playback_Maximum_Swing;
  short       Melody_FIR_Coeff_Tbl[25];
  short       audio_compensation_coeff[3][45];
```

```
    L1_audio_abf_param_struct_T  abf_param;
} l1audio_param_EX2_T;
```

Description:

Decompose the buf into the META_DLL PC size structure “l1audio_param_EX2_T”

Return Value:

Table 6-366 The return value of META_NVRAM_Decompose_AudioParam_EX2

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-367 The parameter of META_NVRAM_Decompose_AudioParam_EX2

Parameter	IN/OUT	Description
param	IN	The parameter for buffer decomposing
buf	IN/OUT	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.13 META_NVRAM_Compose_AC_SWFIR_Param

Definition:

META_RESULT __stdcall META_NVRAM_Compose_AC_SWFIR_Param(const l1audio_swfir_T *param, char *buf, const int buf_len);

```
typedef struct
{
    short audio_compensation_filter_sw_ver_coeffs[3][3][45];
} l1audio_swfir_T;
```

Description:

Compose the buf into the META_DLL PC size structure “l1audio_swfir_T”

Return Value:

Table 6-368 The return value of META_NVRAM_Compose_AC_SWFIR_Param

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-369 The parameter of META_NVRAM_Compose_AC_SWFIR_Param

Parameter	IN/OUT	Description
param	IN/OUT	The parameter for buffer composing
buf	IN	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.14 META_NVRAM_Decompose_AC_SWFIR_Param

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AudioParam_EX2(l1audio_param_EX2_T *param, const char *buf, const int buf_len);

```
typedef struct
{
   short audio_compensation_filter_sw_ver_coeffs[3][3][45];
} l1audio_swfir_T;
```

Description:

Decompose the buf into the META_DLL PC size structure “l1audio_swfir_T”

Return Value:

Table 6-370 The return value of META_NVRAM_Decompose_AC_SWFIR_Param

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-371 The parameter of META_NVRAM_Decompose_AC_SWFIR_Param

Parameter	IN/OUT	Description
param	IN	The parameter for buffer decomposing
buf	IN/OUT	The NVRAM buffer used to hold the nvram_ef_audio_besloudness_struct
buf_len	IN	The length of the buf

6.8.15 RF related NVRAM buffer operations

6.8.15.1 META_NVRAM_interRampData_Len

Definition:

META_RESULT __stdcall META_NVRAM_interRampData_Len(int *len)

Description:

This function returns the size of inter-ramp table.

Return Value:

Table 6-372 The return value of META_NVRAM_interRampData_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-373 The parameter of META_NVRAM_interRampData_Len

Parameter	IN/OUT	Description
Len	OUT	size of inter-ramp table

6.8.15.2 META_NVRAM_Compose_interRampData

Definition:

META_RESULT __stdcall META_NVRAM_Compose_interRampData(const l1cal_interRampData_T *tbl, char *buf, const int buf_len)

typedef struct {

```
        unsigned char       interRampData[16];
}l1cal_interRampData_T;
`
```

Description:

Compose inter-ramp Table. Usually, once the calibrated power level for each band are acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-374 The return value of META_NVRAM_Compose_interRampData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-375 The parameter of META_NVRAM_Compose_interRampData

Parameter	IN/OUT	Description
tbl	IN	Inter-ramp table
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.3 META_NVRAM_Decompose_interRampData

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_interRampData(l1cal_interRampData_T *tbl, const char *buf, const int buf_len)

Description:

Decompose inter-ramp Table. Usually, once the buffer of inter-ramp profile and transmission level data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1cal_interRampData_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-376 The return value of META_NVRAM_Decompose_interRampData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-377 The parameter of META_NVRAM_Decompose_interRampData

Parameter	IN/OUT	Description
tbl	IN/OUT	Inter-ramp table
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.4 META_NVRAM_crystalAfcData_Len

Definition:

META_RESULT __stdcall META_NVRAM_crystalAfcData_Len(int *len)

Description:

This function returns the size of crystal afc data.

Return Value:

Table 6-378 The return value of META_NVRAM_crystalAfcData_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-379 The parameter of META_NVRAM_crystalAfcData_Len

Parameter	IN/OUT	Description
len	OUT	Size of crystal afc data

6.8.15.5 META_NVRAM_Compose_crystalAfcData

Definition:

META_RESULT __stdcall META_NVRAM_Compose_crystalAfcData(const l1cal_crystalAfcData_T *xo_afc, char *buf, const int buf_len)

```
#define XO_SlopeArea_Num      8

typedef struct {
            int         min_freq;
            short       min_dac;
            int         inv_slope;
}XO_SLOPE_AREA_DATA;

typedef struct {
            XO_SLOPE_AREA_DATA      XO_SlopeAreaData[XO_SlopeArea_Num];
}l1cal_crystalAfcData_T;
```

Description:

Compose crystal afc data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-380 The return value of META_NVRAM_Compose_crystalAfcData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-381 The parameter of META_NVRAM_Compose_crystalAfcData

Parameter	IN/OUT	Description
xo_afc	IN	Crystal afc data
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.6 META_NVRAM_Decompose_crystalAfcData

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_crystalAfcData(l1cal_crystalAfcData_T *xo_afc, const char *buf, const int buf_len)

Description:

Decompose crystal afc data. Usually, once the buffer of crystal afc data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1cal_crystalAfcData_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-382 The return value of META_NVRAM_Decompose_crystalAfcData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-383 The parameter of META_NVRAM_Decompose_crystalAfcData

Parameter	IN/OUT	Description
xo_afc	IN/OUT	Crystal afc data
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.7 META_NVRAM_agcPathLoss_Len

Definition:

META_RESULT __stdcall META_NVRAM_agcPathLoss_Len(int *len)

Description:

This function returns the size of agcPathLoss.

Return Value:

Table 6-384 The return value of META_NVRAM_agcPathLoss_Len

Return value	Description
META_SUCCESS	Success

Return value	Description
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-385 The parameter of META_NVRAM_agcPathLoss_Len

Parameter	IN/OUT	Description
len	OUT	Size of agcPathLoss

6.8.15.8 META_NVRAM_Compose_agcPathLoss

Definition:

META_RESULT __stdcall META_NVRAM_Compose_agcPathLoss(const l1cal_agcPathLoss_T *loss, char *buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_Compose_agcPathLoss_r(const int meta_handle, const l1cal_agcPathLoss_T* loss, char* buf, const int buf_len)

```
typedef struct
{
        short     max_arfcn;       // The maximum ARFCN of the indicated sub-band
        char      gain_offset;     // The maximum available gain of transceiver of the indicated sub-band
} sAGCGAINOFFSET;

typede enum
{
        FrequencyBand400 = 0,           // GSM 450/480 band
        FrequencyBand850,               // GSM 850 band
        FrequencyBand900,               // GSM 900 band
        FrequencyBand1800,              // DCS 1800 band
        FrequencyBand1900,              // PCS 1900 band
        FrequencyBandCount              // count of supported bands
} FrequencyBand;
```

```
#define PLTABLE_SIZE 13                // element count of path loss table

typedef struct
{
        sAGCGAINOFFSET          agcPathLoss[FrequencyBandCount][PLTABLE_SIZE];
} l1cal_agcPathLoss_T;
```

Description:

Compose agcPathLoss. Usually, once the calibrated path loss data for each band are acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-386 The return value of META_NVRAM_Compose_agcPathLoss

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-387 The parameter of META_NVRAM_Compose_agcPathLoss

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
loss	IN	agcPathLoss
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.9 META_NVRAM_Decompose_agcPathLoss

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_agcPathLoss(l1cal_agcPathLoss_T *loss, const char *buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_Decompose_agcPathLoss_r(const int meta_handle, l1cal_agcPathLoss_T* loss, const char* buf, const int buf_len)

Description:

Decompose agcPathLoss. Usually, once the buffer of path loss data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1cal_agcPathLoss_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-388 The return value of META_NVRAM_Decompose_agcPathLoss

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-389 The parameter of META_NVRAM_Decompose_agcPathLoss

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
loss	IN/OUT	agcPathLoss
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.10 META_NVRAM_rampTable_Len

Definition:

META_RESULT __stdcall META_NVRAM_rampTable_Len(int *len)

Description:

This function returns the size of ramp table.

Return Value:

Table 6-390 The return value of META_NVRAM_rampTable_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-391 The parameter of META_NVRAM_rampTable_Len

Parameter	IN/OUT	Description
Len	OUT	Size of ramp table

6.8.15.11 META_NVRAM_Compose_rampTable

Definition:

META_RESULT __stdcall META_NVRAM_Compose_rampTable(const l1cal_rampTable_T *tbl, char *buf, const int buf_len)

```
typedef struct
{
        unsigned char           point[2][16];      // ramp up/down profile
} sRAMPAREADATA;

typedef  struct
{
        short                   max_arfcn;         // sub-band boundary of this PCL weighting area
        unsigned short          mid_level;         // PCL boundary level to apply high/low weighting
        unsigned short          hi_weight;         // scale factor of PCLs higher than mid_level
        unsigned short          low_weight;        // scale factor of PCLs lower than mid_level
} sARFCN_SECTION;

typedef struct
{
        int                     lowest_power;             // The lower apc power of the indicated band
        unsigned short          power[16];                // The mapping of power level to apc dac value
        sRAMPAREADATA ramp[ PROFILE_NUM ];      // ramp profile
        sARFCN_SECTION arfcn_weight[ ARFCN_SECTION_NUM ];       // profile of weighting power level
        unsigned short          battery_compensate[3][3];                  // [volt][temp]
        short                   tx_afc_offset;
} sRAMPDATA;


typedef struct
{
        sRAMPDATA               rampData;          // apc ramp profile of all bands
} l1cal_rampTable_T;
```

Description:

Compose ramp Table. Usually, once the calibrated power level for each band are acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-392 The return value of META_NVRAM_Compose_rampTable

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-393 The parameter of META_NVRAM_Compose_rampTable

Parameter	IN/OUT	Description
tbl	IN	Ramp table
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.12 META_NVRAM_Decompose_rampTable

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_rampTable(l1cal_rampTable_T *tbl, const char *buf, const int buf_len)

Description:

Decompose ramp Table. Usually, once the buffer of ramp profile and transmission level data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure 1cal_rampTable_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-394 The return value of META_NVRAM_Decompose_rampTable

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-395 The parameter of META_NVRAM_Decompose_rampTable

Parameter	IN/OUT	Description
tbl	IN/OUT	Ramp table
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.13 META_NVRAM_rampTable_Len_Ex

Definition:

META_RESULT __stdcall META_NVRAM_rampTable_Len_Ex (int *len)

Description:

This function returns the size of ramp table.

Return Value:

Table 6-396 The return value of META_NVRAM_rampTable_Len_Ex

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-397 The parameter of META_NVRAM_rampTable_Len_Ex

Parameter	IN/OUT	Description
Len	OUT	Size of ramp table

6.8.15.14 META_NVRAM_Compose_rampTable_Ex

Definition:

```
        META_RESULT __stdcall META_NVRAM_Compose_rampTable_Ex (const l1cal_rampTable_T_Ex *tbl, char
*buf, const int buf_len)

#define ARFCN_SECTION_NUM_Ex      64

typedef struct

{

            unsigned char                 point[2][16];        // ramp up/down profile

} sRAMPAREADATA;


typedef  struct

{

            short                          max_arfcn;          // sub-band boundary of this PCL weighting area

            unsigned short                 mid_level;          // PCL boundary level to apply high/low weighting

            unsigned short                 hi_weight;          // scale factor of PCLs higher than mid_level

            unsigned short                 low_weight;         // scale factor of PCLs lower than mid_level

} sARFCN_SECTION;


typedef struct

{

            int                            lowest_power;               // The lower apc power of the indicated band

            unsigned short                 power[16];                  // The mapping of power level to apc dac value

            sRAMPAREADATA ramp[ PROFILE_NUM ];     // ramp profile

            sARFCN_SECTION arfcn_weight[ ARFCN_SECTION_NUM_Ex ];// profile of weighting power level

            unsigned short    battery_compensate[3][3];                        // [volt][temp]
            short                          tx_afc_offset;
} sRAMPDATA _Ex;
```

```
typedef struct
{
        sRAMPDATA _Ex          rampData;               // apc ramp profile of all bands
} l1cal_rampTable_T_Ex;
```

Description:

Compose ramp Table. Usually, once the calibrated power level for each band are acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-398 The return value of META_NVRAM_Compose_rampTable_Ex

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-399 The parameter of META_NVRAM_Compose_rampTable_Ex

Parameter	IN/OUT	Description
tbl	IN	Ramp table
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.15 META_NVRAM_Decompose_rampTable_Ex

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_rampTable_Ex (l1cal_rampTable_T_Ex *tbl, const char *buf, const int buf_len)

Description:

Decompose ramp Table. Usually, once the buffer of ramp profile and transmission level data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into 1 1cal_rampTable_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-400 The return value of META_NVRAM_Decompose_rampTable_Ex

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-401 The parameter of META_NVRAM_Decompose_rampTable_Ex

Parameter	IN/OUT	Description
tbl	IN/OUT	Ramp table
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.16 META_NVRAM_rampTable_Len_Ex2

Definition:

META_RESULT __stdcall META_NVRAM_rampTable_Len_Ex2 (int *len)

Description:

This function returns the size of ramp table.

Return Value:

Table 6-402 The return value of META_NVRAM_rampTable_Len_Ex2

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-403 The parameter of META_NVRAM_rampTable_Len_Ex2

Parameter	IN/OUT	Description
Len	OUT	Size of ramp table

6.8.15.17 META_NVRAM_Compose_rampTable_Ex2

Definition:

```
        META_RESULT __stdcall META_NVRAM_Compose_rampTable_Ex2 (const l1cal_rampTable_T_Ex *tbl, char
*buf, const int buf_len)

typedef struct
{
            unsigned char               point[2][16];       // ramp up/down profile
} sRAMPAREADATA;


typedef  struct
{
            short                       max_arfcn;          // sub-band boundary of this PCL weighting area
            unsigned short              mid_level;          // PCL boundary level to apply high/low weighting
            unsigned short              hi_weight;          // scale factor of PCLs higher than mid_level
            unsigned short              low_weight;         // scale factor of PCLs lower than mid_level
} sARFCN_SECTION;


typedef struct
{
            int                         lowest_power;                // The lower apc power of the indicated band
            unsigned short              power[16];                   // The mapping of power level to apc dac value
            sRAMPAREADATA ramp[ PROFILE_NUM ];      // ramp profile
            sARFCN_SECTION arfcn_weight[ ARFCN_SECTION_NUM ];// profile of weighting power
            unsigned short             battery_compensate[3][3];                 // [volt][temp]
            short                       tx_afc_offset;
            unsigned char              vbias[16];


} sRAMPDATA _Ex2;
```

```
typedef struct
{
        sRAMPDATA _Ex   2           rampData;                       // apc ramp profile of all bands
} l1cal_rampTable_T_Ex2;
```

Description:

Compose ramp Table. Usually, once the calibrated power level for each band are acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-404 The return value of META_NVRAM_Compose_rampTable_Ex2

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-405 The parameter of META_NVRAM_Compose_rampTable_Ex2

Parameter	IN/OUT	Description
tbl	IN	Ramp table
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.18 META_NVRAM_Decompose_rampTable_Ex2

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_rampTable_Ex2 (l1cal_rampTable_T_Ex2 *tbl, const char *buf, const int buf_len)

Description:

Decompose ramp Table. Usually, once the buffer of ramp profile and transmission level data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure 1cal_rampTable_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-406 The return value of META_NVRAM_Decompose_rampTable_Ex2

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-407 The parameter of META_NVRAM_Decompose_rampTable_Ex2

Parameter	IN/OUT	Description
tbl	IN/OUT	Ramp table
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.19 META_NVRAM_Compose_MT6140tx_PaVbias

Definition:

META_RESULT __stdcall META_NVRAM_Compose_MT6140tx_PaVbias (const mt6140tx *pavbias, char *buf, const int buf_len)

```
typedef struct {
    pa_vbias GSM850_pa_vbias[8];
    pa_vbias GSM900_pa_vbias[8];
    pa_vbias                    DCS1800_pa_vbias[8];
    pa_vbias PCS1900_pa_vbias[8];
}mt6140tx_pa_vbias;

typedef struct{
    mt6140tx_pa_vbias data;
}mt6140tx;
```

Description:

Compose mt6140tx Table. Usually, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-408 The return value of META_NVRAM_Compose_MT6140tx_PaVbias

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-409 The parameter of META_NVRAM_Compose_MT6140tx_PaVbias

Parameter	IN/OUT	Description
pavbias	IN/OUT	Output mt6140_tx data
buf	OUT	Input buffer to decompose.
buf_len	OUT	Size of buf

6.8.15.20 META_NVRAM_Decompose_MT6140tx_PaVbias

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_MT6140tx_PaVbias(mt6140tx *pavbias, const char *buf, const int buf_len)

```
typedef struct {
    pa_vbias GSM850_pa_vbias[8];
    pa_vbias GSM900_pa_vbias[8];
    pa_vbias DCS1800_pa_vbias[8];
    pa_vbias PCS1900_pa_vbias[8];
}mt6140tx_pa_vbias;

typedef struct{
    mt6140tx_pa_vbias data;
}mt6140tx;
```

Description:

Decompose mt6140tx Table. Usually, once the buffer of mt6140tx t data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure mt6140tx, and doesn't take care the byte alignment problem.

Return Value:

Table 6-410 The return value of META_NVRAM_Decompose_MT6140tx_PaVbias

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-411 The parameter of META_NVRAM_Decompose_MT6140tx_PaVbias

Parameter	IN/OUT	Description
pavbias	IN/OUT	nvram_ef_btradio_rfmd3500_struct
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.21 META_NVRAM_BBTXParameters_Len

Definition:

META_RESULT __stdcall META_NVRAM_BBTXParameters_Len(int *len)

Description:

This function returns the size of sBBTXParameters table.

Return Value:

Table 6-412 The return value of META_NVRAM_BBTXParameters_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-413 The parameter of META_NVRAM_BBTXParameters_Len

Parameter	IN/OUT	Description
Len	OUT	Size of sBBTXParameters table

6.8.15.22 META_NVRAM_Compose_BBTXParameters

Definition:

META_RESULT __stdcall META_NVRAM_Compose_BBTXParameters(const BBTXParameters_T *bbtx, char *buf, const int buf_len);

```
typedef struct{
    unsigned char bbtx_common_mode_voltage;
    unsigned char bbtx_gain;
    unsigned char bbtx_calrcsel;
    unsigned char bbtx_trimI;
    unsigned char bbtx_trimQ;
     unsigned char bbtx_dccoarseI;
   unsigned char bbtx_dccoarseQ;
   unsigned char bbtx_offsetI;
   unsigned char bbtx_offsetQ;
   unsigned char bbtx_isCalibrated;
   int         apc_bat_low_voltage;
   int         apc_bat_high_voltage;
   int         apc_bat_low_temperature;
   int         apc_bat_high_temperature;
   unsigned char bbtx_common_mode_voltage_h;
   unsigned char bbtx_gain_h;
   unsigned char bbtx_calrcsel_h;
   unsigned char bbtx_trimI_h;
   unsigned char bbtx_trimQ_h;
    unsigned char bbtx_dccoarseI_h;
   unsigned char bbtx_dccoarseQ_h;
```

```
    unsigned char bbtx_offsetI_h;
    unsigned char bbtx_offsetQ_h;
    unsigned char bbtx_phsel;
    unsigned char bbtx_phsel_h;
    unsigned char bbrx_gsm850_gsm900_swap;
    unsigned char bbrx_dcs1800_pcs1900_swap;
}BBTXParameters_T;
```

Description:

Compose sBBTXParameters. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-414 The return value of META_NVRAM_Compose_BBTXParameters

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-415 The parameter of META_NVRAM_Compose_BBTXParameters

Parameter	IN/OUT	Description
bbtx	IN	BBTXParameters_T
Buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.23 META_NVRAM_Decompose_BBTXParameters

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_BBTXParameters(BBTXParameters_T *bbtx, const char *buf, const int buf_len);

```
typedef struct{
    unsigned char bbtx_common_mode_voltage;
    unsigned char bbtx_gain;
    unsigned char bbtx_calrcsel;
    unsigned char bbtx_trimI;
    unsigned char bbtx_trimQ;
    unsigned char bbtx_dccoarseI;
   unsigned char bbtx_dccoarseQ;
   unsigned char bbtx_offsetI;
   unsigned char bbtx_offsetQ;
   unsigned char bbtx_isCalibrated;
   int        apc_bat_low_voltage;
   int        apc_bat_high_voltage;
   int        apc_bat_low_temperature;
   int        apc_bat_high_temperature;
   unsigned char bbtx_common_mode_voltage_h;
   unsigned char bbtx_gain_h;
   unsigned char bbtx_calrcsel_h;
   unsigned char bbtx_trimI_h;
   unsigned char bbtx_trimQ_h;
    unsigned char bbtx_dccoarseI_h;
   unsigned char bbtx_dccoarseQ_h;
   unsigned char bbtx_offsetI_h;
   unsigned char bbtx_offsetQ_h;
   unsigned char bbtx_phsel;
   unsigned char bbtx_phsel_h;
   unsigned char bbrx_gsm850_gsm900_swap;
   unsigned char bbrx_dcs1800_pcs1900_swap;
}BBTXParameters_T;
```

Description:

Decompose sBBTXParameters. Usually, once the buffer of sBBTXParameters data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure BBTXParameters_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-416 The return value of META_NVRAM_Decompose_BBTXParameters

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-417 The parameter of META_NVRAM_Decompose_BBTXParameters

Parameter	IN/OUT	Description
bbtx	IN/OUT	Pointer to BBTXParameters_T
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.24 META_NVRAM_Compose_ad6546tx

Definition:

META_RESULT __stdcall META_NVRAM_Compose_ad6546tx(const ad6546tx *adtx, char *buf, const int buf_len);

```
typedef struct
{
   unsigned char REFDET_SLOPE_SKEW;
   unsigned char AM_FB_DAC;

}ad6546tx;
```

Description:

Compose ad6546tx RF chip data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-418 The return value of META_NVRAM_Compose_ad6546tx

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-419 The parameter of META_NVRAM_Compose_ad6546tx

Parameter	IN/OUT	Description
adtx	IN	ad6546tx
buf	IN/OUT	Output buffer to be composed.
buf_len	IN	Buffer length

6.8.15.25 META_NVRAM_Decompose_ad6546tx

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_ad6546tx(ad6546tx *adtx, const char *buf, const int buf_len);

Description:

Decompose ad6546tx RF chip data. Usually, once the buffer of strcture is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure ad6546tx, and doesn’t take care the byte alignment problem.

Return Value:

Table 6-420 The return value of META_NVRAM_Decompose_ad6546tx

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-421 The parameter of META_NVRAM_Decompose_ad6546tx

Parameter	IN/OUT	Description
adtx	IN/OUT	Output ad6546tx RF chip data
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.8.15.26 META_NVRAM_ClosedLoopTXPC_Len

Definition:

META_RESULT __stdcall META_NVRAM_ClosedLoopTXPC_Len(int *len);

Description:

This function returns the size of l1cal_txpc_T.

Return Value:

Table 6-422 The return value of META_NVRAM_ClosedLoopTXPC_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-423 The parameter of META_NVRAM_ClosedLoopTXPC_Len

Parameter	IN/OUT	Description
len	OUT	Size of l1cal_txpc_T

6.8.15.27 META_NVRAM_Compose_ClosedLoopTXPC

Definition:

```
META_RESULT __stdcall META_NVRAM_Compose_ClosedLoopTXPC(const l1cal_txpc_T *tbl, char *buf,
const int buf_len);
typedef struct
{
  unsigned short data[8];
} sTXPC_TEMPDATA;

typedef struct
{
  char          is_calibrated;
```

```
    sTXPC_ADCDATA  adc[FrequencyBandCount];
    short         temperature;
    sTXPC_TEMPDATA temp[FrequencyBandCount];
  } sTXPC_L1CAL;

  typedef sTXPC_L1CAL l1cal_txpc_T;
```

Description:

Compose function for l1cal_txpc_T for closed-loop compensation calibration data

Return Value:

Table 6-424 The return value of META_NVRAM_Compose_ClosedLoopTXPC

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-425 The parameter of META_NVRAM_Compose_ClosedLoopTXPC

Parameter	IN/OUT	Description
loss	IN	l1cal_txpc_T
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.28 META_NVRAM_Decompose_ClosedLoopTXPC

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_ClosedLoopTXPC(l1cal_txpc_T *tbl, const char *buf, const int buf_len);

Description:

Decompose function for l1cal_txpc_T for closed-loop compensation calibration data.

Return Value:

Table 6-426 The return value of META_NVRAM_Decompose_ClosedLoopTXPC

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-427 The parameter of META_NVRAM_Decompose_ClosedLoopTXPC

Parameter	IN/OUT	Description
loss	IN/OUT	L1cal_txpc_T
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.29 META_NVRAM_Compose_AvgW_RFSpecialCoef

Definition:

META_RESULT __stdcall META_NVRAM_Compose_AvgW_RFSpecialCoef(const RF_AvgW_Coef_T *rf_mod_coef, char *buf, const int buf_len);

```
typedef struct
{
   short w_re[19];
   short w_im[19];
}RF_AvgW_Coef_T;
```

Description:

Compose function for RF_AvgW_Coef_T for IRR W coefficient calibration data

Return Value:

Table 6-428 The return value of META_NVRAM_Compose_AvgW_RFSpecialCoef

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-429 The parameter of META_NVRAM_Compose_AvgW_RFSpecialCoef

Parameter	IN/OUT	Description
loss	IN	RF_AvgW_Coef_T
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.30 META_NVRAM_Decompose_AvgW_RFSpecialCoef

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_AvgW_RFSpecialCoef(RF_AvgW_Coef_T *rf_mod_coef, const char *buf, const int buf_len);

Description:

Decompose function for RF_AvgW_Coef_T for IRR W coefficient calibration data.

Return Value:

Table 6-430 The return value of META_NVRAM_Decompose_AvgW_RFSpecialCoef

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-431 The parameter of META_NVRAM_Decompose_AvgW_RFSpecialCoef

Parameter	IN/OUT	Description
loss	IN/OUT	RF_AvgW_Coef_T
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.31 META_NVRAM_lnaPathLoss_Len

Definition:

META_RESULT __stdcall META_NVRAM_lnaPathLoss_Len(int *len);

Description:

This function returns the size of lnaPathLoss.

Return Value:

Table 6-432 The return value of META_NVRAM_lnaPathLoss_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-433 The parameter of META_NVRAM_lnaPathLoss_Len

Parameter	IN/OUT	Description
len	OUT	Size of lnaPathLoss

6.8.15.32 META_NVRAM_Compose_lnaPathLoss

Definition:

META_RESULT __stdcall META_NVRAM_Compose_lnaPathLoss(const l1cal_lnaPathLoss_T *loss, char *buf, const int buf_len);

```
typede enum
{
        FrequencyBand400 = 0,           // GSM 450/480 band
        FrequencyBand850,               // GSM 850 band
        FrequencyBand900,               // GSM 900 band
        FrequencyBand1800,              // DCS 1800 band
        FrequencyBand1900,              // PCS 1900 band
        FrequencyBandCount              // count of supported bands
} FrequencyBand;

#define PLTABLE_SIZE 13         // element count of path loss table

typedef struct
{
  char gain_offset_middle;
  char gain_offset_low;
} sLNAGAINOFFSET;
typedef struct
{
```

sLNAGAINOFFSET lnaPathLoss[FrequencyBandCount][PLTABLE_SIZE];

}l1cal_lnaPathLoss_T;

Description:

Compose agcPathLoss. Usually, once the calibrated path loss data for each band are acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-434 The return value of META_NVRAM_Compose_lnaPathLoss

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-435 The parameter of META_NVRAM_Compose_lnaPathLoss

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
loss	IN	lnaPathLoss
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.33 META_NVRAM_Decompose_lnaPathLoss

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_lnaPathLoss(l1cal_lnaPathLoss_T *loss, const char *buf, const int buf_len);

META_RESULT __stdcall META_NVRAM_Decompose_lnaPathLoss_r(const int meta_handle, l1cal_lnaPathLoss_T* loss, const char* buf, const int buf_len);

Description:

Decompose lnaPathloss. Usually, once the buffer of path loss data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1cal_agcPathLoss_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-436 The return value of META_NVRAM_Decompose_lnaPathLoss

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-437 The parameter of META_NVRAM_Decompose_lnaPathLoss

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
loss	IN/OUT	lnaPathLoss
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.34 META_NVRAM_Compose_temperatureADC

Definition:

META_RESULT __stdcall META_NVRAM_Compose_temperatureADC(const l1cal_temperatureADC_T* dac, char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_Compose_temperatureADC_r(const int meta_handle, const l1cal_temperatureADC_T* dac, char* buf, const int buf_len)

```
typedef struct
{
   unsigned short data[8];
} sTEMPERATURE_ADC_L1CAL;

typedef sTEMPERATURE_ADC_L1CAL l1cal_temperatureADC_T;
```

Description:

Compose GGE temperature ADC date to raw date buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-438 The return value of META_NVRAM_Compose_temperatureADC

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-439 The parameter of META_NVRAM_Compose_temperatureADC

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
dac	IN	GGE temperature ADC data
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.35 META_NVRAM_Decompose_temperatureADC

Definition:

META_RESULT __stdcall META_NVRAM_Decompose_temperatureADC(l1cal_temperatureADC_T* dac, const char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_Decompose_temperatureADC_r(const int meta_handle, l1cal_temperatureADC_T* dac, const char* buf, const int buf_len)

Description:

Decompose the raw date buffer of GGE temperature ADC to structure l1cal_temperatureADC_T. This function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1cal_temperatureADC_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-440 The return value of META_NVRAM_Decompose_temperatureADC

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-441 The parameter of META_NVRAM_Decompose_temperatureADC

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
dac	IN/OUT	GGE temperature ADC data
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.36 META_NVRAM_Compose_EPSKtxPaOctLevData

Definition:

META_RESULT __stdcall META_NVRAM_Compose_EPSKtxPaOctLevData(const RF_EPSK_8PA_SPECIAL_Coef_T* epsk_specialCoef, char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_Compose_EPSKtxPaOctLevData_r(const int meta_handle, const RF_EPSK_8PA_SPECIAL_Coef_T* epsk_specialCoef, char* buf, const int buf_len)

```
#define  PA_OCT_16_LEVEL 16

typedef struct

{

   short       pcl_index;

   unsigned char  pa_vbias;

   unsigned short pa_gain;

} epsk_pa_vbias;


typedef struct

{

   epsk_pa_vbias GSM850_8pa_vbias[PA_OCT_16_LEVEL];
```

```
    epsk_pa_vbias GSM900_8pa_vbias[PA_OCT_16_LEVEL];
    epsk_pa_vbias DCS1800_8pa_vbias[PA_OCT_16_LEVEL];
    epsk_pa_vbias PCS1900_8pa_vbias[PA_OCT_16_LEVEL];
} EPSK_8PA_VBIAS;

typedef struct
{
    EPSK_8PA_VBIAS data;
} RF_EPSK_8PA_TX_Coef;

typedef struct
{
    RF_EPSK_8PA_TX_Coef  tx;
} RF_EPSK_8PA_SPECIAL_Coef_T;
```

Description:

Compose the EPSK PA level control date to raw date buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-442 The return value of META_NVRAM_Compose_EPSKtxPaOctLevData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-443 The parameter of META_NVRAM_Compose_EPSKtxPaOctLevData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

Parameter	IN/OUT	Description
epsk_specialCoef	IN	EPSK PA level control data
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.37 META_NVRAM_DeCompose_EPSKtxPaOctLevData

Definition:

META_RESULT __stdcall META_NVRAM_DeCompose_EPSKtxPaOctLevData(RF_EPSK_8PA_SPECIAL_Coef_T* epsk_specialCoef, const char* buf, const int buf_len);

META_RESULT __stdcall META_NVRAM_DeCompose_EPSKtxPaOctLevData_r(const int meta_handle, RF_EPSK_8PA_SPECIAL_Coef_T* epsk_specialCoef, const char* buf, const int buf_len);

Description:

Decompose the raw buffer of EPSK PA level control data to the structure RF_EPSK_8PA_SPECIAL_Coef_T. This function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure RF_EPSK_8PA_SPECIAL_Coef_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-444 The return value of META_NVRAM_DeCompose_EPSKtxPaOctLevData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-445 The parameter of META_NVRAM_DeCompose_EPSKtxPaOctLevData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
epsk_specialCoef	IN/OUT	EPSK PA level control data
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.38 META_NVRAM_3G_Compose_pathlossData

Definition:

META_RESULT __stdcall META_NVRAM_3G_Compose_pathlossData(const ul1cal_pathlossData_T* pathloss, char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_3G_Compose_pathlossData_r(const int meta_handle, const ul1cal_pathlossData_T* pathloss, char* buf, const int buf_len)

```
#define  CAL_UARFCN_SECTION      15

#define  CAL_TEMP_SECTION         8


typedef  struct

{

   unsigned short  max_uarfcn;

   char   path_loss_H;//loss;

   char   path_loss_M;//gain_diff_HM;

   char   path_loss_L;//gain_diff_HL;

   char   path_loss_LPM_offset;

}  U_sAGCGAINOFFSET;

typedef  struct

{

   U_sAGCGAINOFFSET  gain_of_uarfcn[CAL_UARFCN_SECTION];

}  U_sTEMPAGCOFFSET;


typedef struct

{

   U_sTEMPAGCOFFSET  pathlossData[CAL_TEMP_SECTION];

} ul1cal_pathlossData_T;
```

Description:

Compose 3G pathLoss Data to raw data buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-446 The return value of META_NVRAM_3G_Compose_pathlossData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-447 The parameter of META_NVRAM_3G_Compose_pathlossData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
pathloss	IN	3G pathLoss data
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.39 META_NVRAM_3G_Decompose_pathlossData

Definition:

META_RESULT __stdcall META_NVRAM_3G_Decompose_pathlossData(ul1cal_pathlossData_T* pathloss, const char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_3G_Decompose_pathlossData_r(const int meta_handle, ul1cal_pathlossData_T* pathloss, const char* buf, const int buf_len)

Description:

Decompose raw buffer of 3G pathLoss data to the structure ul1cal_pathlossData_T. This function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure ul1cal_pathlossData_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-448 The return value of META_NVRAM_3G_Decompose_pathlossData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-449 The parameter of META_NVRAM_3G_Decompose_pathlossData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
pathloss	IN/OUT	3G pathLoss data
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.40 META_NVRAM_3G_Compose_tempdacData

Definition:

META_RESULT __stdcall META_NVRAM_3G_Compose_tempdacData(const ul1cal_tempdacData_T* dac, char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_3G_Compose_tempdacData_r(const int meta_handle, const ul1cal_tempdacData_T* dac, char* buf, const int buf_len)

```
typedef struct
{
  unsigned short tempdacData[8];
} ul1cal_tempdacData_T;
```

Description:

Compose 3G temperature dac Data to raw data buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-450 The return value of META_NVRAM_3G_Compose_tempdacData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-451 The parameter of META_NVRAM_3G_Compose_tempdacData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
dac	IN	3G Temperature dac data
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.41 META_NVRAM_3G_Decompose_tempdacData

Definition:

META_RESULT __stdcall META_NVRAM_3G_Decompose_tempdacData(ul1cal_tempdacData_T* dac, const char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_3G_Decompose_tempdacData_r(const int meta_handle, ul1cal_tempdacData_T* dac, const char* buf, const int buf_len)

Description:

Decompose raw buffer ot 3G Temperature dac data to structure ul1cal_tempdacData_T. This function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure ul1cal_tempdacData_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-452 The return value of META_NVRAM_3G_Decompose_tempdacData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-453 The parameter of META_NVRAM_3G_Decompose_tempdacData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
dac	IN/OUT	3G Temperature dac data
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.42 META_NVRAM_3G_Compose_txPaOctLevData

Definition:

META_RESULT __stdcall META_NVRAM_3G_Compose_txPaOctLevData(const ul1cal_txPaOctLevData_T* paoctlevdata, char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_3G_Compose_txPaOctLevData_r(const int meta_handle, const ul1cal_txPaOctLevData_T* paoctlevdata, char* buf, const int buf_len)

```
typedef struct
{
   unsigned char   pa_mode;
   char          prf;
   unsigned char   dc2dc_lvl;
   unsigned char   vm1;
   unsigned char   vm2;
   unsigned short  vbias_dac;
   unsigned short  pa_gain;
} U_sPMULEVHANDLE;

typedef  struct
{
   unsigned char     octlev_num_section;
   unsigned int      pa_phase_compensation[3]; // 0: PA high mode, 1: PA mid mode
   U_sPMULEVHANDLE   pmu_level_handle[8];
} U_sPAOCTLVLSETTING;
```

```
typedef struct
{
   U_sPAOCTLVLSETTING txPaOctLevData;
} ul1cal_txPaOctLevData_T;
```

Description:

Compose the 3G PA level control data to raw data buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-454 The return value of META_NVRAM_3G_Compose_txPaOctLevData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-455 The parameter of META_NVRAM_3G_Compose_txPaOctLevData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
paoctlevdata	IN	3G PA level control data
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.43 META_NVRAM_3G_Decompose_txPaOctLevData

Definition:

META_RESULT __stdcall META_NVRAM_3G_Decompose_txPaOctLevData(ul1cal_txPaOctLevData_T* paoctlevdata, const char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_3G_Decompose_txPaOctLevData_r(const int meta_handle, ul1cal_txPaOctLevData_T* paoctlevdata, const char* buf, const int buf_len)

Description:

Decompose raw buffer of 3G PA level control data to the structure ul1cal_txPaOctLevData_T. This function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure ul1cal_txPaOctLevData_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-456 The return value of META_NVRAM_3G_Decompose_txPaOctLevData

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-457 The parameter of META_NVRAM_3G_Decompose_txPaOctLevData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
paoctlevdata	IN/OUT	3G PA level control data
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.15.44 META_NVRAM_3G_Compose_txdacData_B

Definition:

META_RESULT __stdcall META_NVRAM_3G_Compose_txdacData_B(const ul1cal_txdacData_T_B* txdac, char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_3G_Compose_txdacData_B_r(const int meta_handle, const ul1cal_txdacData_T_B* txdac, char* buf, const int buf_len)

```
#define CAL_UARFCN_SECTION      15

#define CAL_PWR_DETECTOR_SECTION 32


typedef struct

{

  unsigned short level_0;
```

```
    unsigned short level_1;
} U_sDC2DC;

typedef  struct
{
    unsigned short  max_uarfcn;
    short          pwr_offset;
    short          pwr_slope;
} U_sARFCN_SECTION;

typedef  struct
{
    U_sPADATA         pa_data;
    unsigned short    vga_dac[90];
    U_sARFCN_SECTION   vga_comp_by_subband[CAL_UARFCN_SECTION];
    short             vga_comp_by_temperature[8][2];  //[0]:slope, [1]:offset
} U_sTXPOWERDATA;

typedef  struct
{
    unsigned short start;
    unsigned short end;
} U_sHYSTERESISDATA;

typedef struct
{
    unsigned short max_uarfcn;
    short        pwr_offset_dB; /* unit: 1/32 dB, range: -8 ~ +7 dB */
    short         pwr_offset_txdac;
} U_sARFCN_SECTION_B;

typedef  struct
{
    unsigned char      pwr_dt_thr;
    unsigned char      pwr_dt_section;
    unsigned short     pwr_dt_dac[ CAL_PWR_DETECTOR_SECTION ];
    short              pwr_dt_value[ CAL_PWR_DETECTOR_SECTION ]; //bit0~4 is used for fractions
    U_sARFCN_SECTION_B  pwr_dt_comp_by_subband[ CAL_UARFCN_SECTION ];
    short              pwr_dt_comp_by_temperature[8][2];  //[0]:offset in dB (unit: 1/32 dB), [1]:offset in txdac
}  U_sPWTDTDATA_B;

typedef  struct
{
    U_sDC2DC            pa_dc2dc;
    U_sTXPOWERDATA_B     power_dac[3];   //0:PA high mode, 1:PA mid mode, 2:PA low mode
    U_sHYSTERESISDATA    tx_hysteresis[2];
    U_sPWTDTDATA_B       pwr_dt_data;
} U_sRAMPDATA_B;   // for MT6268B later

typedef struct
{
    U_sRAMPDATA_B txdacData;
} ul1cal_txdacData_T_B;
```

Description:

Compose the 3G tx dac data to raw data buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

It is extended function of META_NVRAM_3G_Compose_txdacData.

Return Value:

Table 6-458 The return value of META_NVRAM_3G_Compose_txdacData_B

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-459 The parameter of META_NVRAM_3G_Compose_txdacData_B

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
txdac	IN	3G TX Dac data
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.15.45 META_NVRAM_3G_Decompose_txdacData_B

Definition:

META_RESULT __stdcall META_NVRAM_3G_Decompose_txdacData_B(ul1cal_txdacData_T_B* txdac, const char* buf, const int buf_len)

META_RESULT __stdcall META_NVRAM_3G_Decompose_txdacData_B_r(const int meta_handle, ul1cal_txdacData_T_B* txdac, const char* buf, const int buf_len)

Description:

Decompose the raw buffer of 3G tx dac data to the structure ul1cal_txdacData_T_B. This function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure ul1cal_txdacData_T_B, and doesn't take care the byte alignment problem.

Return Value:

Table 6-460 The return value of META_NVRAM_3G_Decompose_txdacData_B

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-461 The parameter of META_NVRAM_3G_Decompose_txdacData_B

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
txdac	IN/OUT	3G TX Dac data
buf	IN	Buffer
buf_len	IN	Size of buf

6.8.16 BT related NVRAM buffer operations

6.8.16.1 META_NVRAM_BT_Compose_RFMD3500Radio

Definition:

META_RESULT __stdcall META_NVRAM_BT_Compose_RFMD3500Radio(const nvram_ef_btradio_rfmd3500_struct *radio, char *buf, const int buf_len);

```
typedef struct
{
   unsigned char BluetoothAddress[6];
   unsigned char MinEncryptionSize[1];
   unsigned char MaxEncryptionSize[1];
   unsigned char HCITransportLayerParameters[3];
   unsigned char FixedPIN[16];
   unsigned char FixedPINLength[1];
   unsigned char SleepEnableMask[1];
   unsigned char LowPowerClockParameter[8];
```

```
  unsigned char PowerControlConfiguration[13];
  unsigned char SleepControlParameters[12];
  unsigned char DebugControl[4];
  unsigned char LCandRMOverrideEnable[4];
  unsigned char RadioRegisterOverride[6];
  unsigned char CodecConfiguration[8];
  unsigned char CVSDGainVolumeSettings[6];
  unsigned char VoiceSettings[2];
  unsigned char UserBaudRate[3];

  unsigned char LowPowerDriftRate[1];
  unsigned char MaxTxPowerLevel[1];
  unsigned char AdaptiveFrequencyHoppingParameters[29];
  unsigned char BufferSize[4];
  unsigned char GpioMapping[16];
  unsigned char GpioPolarity[4];
} nvram_ef_btradio_rfmd3500_struct;
```

Description:

Compose nvram_ef_btradio_rfmd3500_struct Table. Usually, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-462 The return value of META_NVRAM_BT_Compose_RFMD3500Radio

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-463 The parameter of META_NVRAM_BT_Compose_RFMD3500Radio

Parameter	IN/OUT	Description
radio	IN	nvram_ef_btradio_rfmd3500_struct
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.16.2 META_NVRAM_BT_Decompose_RFMD3500Radio

Definition:

META_RESULT __stdcall META_NVRAM_BT_Decompose_RFMD3500Radio(nvram_ef_btradio_rfmd3500_struct *radio, const char *buf, const int buf_len);

```
typedef struct
{
  unsigned char BluetoothAddress[6];
  unsigned char MinEncryptionSize[1];
  unsigned char MaxEncryptionSize[1];
  unsigned char HCITransportLayerParameters[3];
  unsigned char FixedPIN[16];
  unsigned char FixedPINLength[1];
  unsigned char SleepEnableMask[1];
  unsigned char LowPowerClockParameter[8];
  unsigned char PowerControlConfiguration[13];
  unsigned char SleepControlParameters[12];
  unsigned char DebugControl[4];
  unsigned char LCandRMOverrideEnable[4];
  unsigned char RadioRegisterOverride[6];
  unsigned char CodecConfiguration[8];
  unsigned char CVSDGainVolumeSettings[6];
  unsigned char VoiceSettings[2];
  unsigned char UserBaudRate[3];
  unsigned char LowPowerDriftRate[1];
  unsigned char MaxTxPowerLevel[1];
  unsigned char AdaptiveFrequencyHoppingParameters[29];
  unsigned char BufferSize[4];
  unsigned char GpioMapping[16];
  unsigned char GpioPolarity[4];
} nvram_ef_btradio_rfmd3500_struct;
```

Description:

Decompose nvram_ef_btradio_rfmd3500_struct Table. Usually, once the buffer of nvram_ef_btradio_rfmd3500_struct data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure nvram_ef_btradio_rfmd3500_struct, and doesn't take care the byte alignment problem.

Return Value:

Table 6-464 The return value of META_NVRAM_BT_Decompose_RFMD3500Radio

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-465 The parameter of META_NVRAM_BT_Decompose_RFMD3500Radio

Parameter	IN/OUT	Description
radio	IN/OUT	nvram_ef_btradio_rfmd3500_struct
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.16.3 META_NVRAM_BT_Compose_MT6601Radio

Definition:

META_RESULT __stdcall META_NVRAM_BT_Compose_MT6601Radio(const nvram_ef_btradio_mt6601_struct *radio, char *buf, const int buf_len);

```
typedef struct
{
    unsigned char BDAddr[6];
    unsigned char ClassOfDevice[3];
    unsigned char LinkKeyType[1];
    unsigned char UnitKey[16];
    unsigned char Encryption[3];
    unsigned char PinCodeType[1];
    unsigned char Voice[2];
    unsigned char Codec[1];
    unsigned char Radio[30];
    unsigned char Sleep[6];
    unsigned char MainOscillatorInfo[5];
    unsigned char LPOInfo[4];
    unsigned char AFH[9];
    unsigned char PTA[49];
    unsigned char WDT[2];
    unsigned char Debug[1];
    unsigned char UART[2];
} nvram_ef_btradio_mt6601_struct;
```

Description:

Compose nvram_ef_btradio_mt6601_struct Table. Usually, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-466 The return value of META_NVRAM_BT_Compose_MT6601Radio

Return value	Description
META_SUCCESS	Success

Return value	Description
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-467 The parameter of META_NVRAM_BT_Compose_MT6601Radio

Parameter	IN/OUT	Description
radio	IN	nvram_ef_btradio_mt6601_struct
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.16.4 META_NVRAM_BT_Decompose_MT6601Radio

Definition:

```
META_RESULT __stdcall
META_NVRAM_BT_Decompose_MT6601Radio(nvram_ef_btradio_mt6601_struct *radio, const char *buf, const
int buf_len);

typedef struct

{

  unsigned char BDAddr[6];

  unsigned char ClassOfDevice[3];

  unsigned char LinkKeyType[1];

  unsigned char UnitKey[16];

  unsigned char Encryption[3];

  unsigned char PinCodeType[1];

  unsigned char Voice[2];

  unsigned char Codec[1];

  unsigned char Radio[30];

  unsigned char Sleep[6];

  unsigned char MainOscillatorInfo[5];

  unsigned char LPOInfo[4];

  unsigned char AFH[9];
```

```
    unsigned char PTA[49];
    unsigned char WDT[2];
    unsigned char Debug[1];
    unsigned char UART[2];
} nvram_ef_btradio_mt6601_struct;
```

Description:

Decompose nvram_ef_btradio_mt6601_struct Table. Usually, once the buffer of nvram_ef_btradio_mt6601_struct data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure nvram_ef_btradio_mt6601_struct, and doesn't take care the byte alignment problem.

Return Value:

Table 6-468 The return value of META_NVRAM_BT_Decompose_MT6601Radio

Return value	Description
META_SUCCESS	Success
META_BUFFER_LEN	The length of buffer is not enough
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-469 The parameter of META_NVRAM_BT_Decompose_MT6601Radio

Parameter	IN/OUT	Description
radio	IN/OUT	nvram_ef_btradio_mt6601_struct
buf	IN/OUT	Buffer
buf_len	IN	Size of buf

6.8.16.5 META_NVRAM_BT_Compose_MT6611Radio

Definition:

META_RESULT __stdcall META_NVRAM_BT_Compose_MT6611Radio(const nvram_ef_btradio_mt6611_struct *radio, char *buf, const int buf_len);

```
typedef struct
{
        unsigned char BDAddr[6];
        unsigned char CapId[1];
        unsigned char LinkKeyType[1];
        unsigned char UnitKey[16];
        unsigned char Encryption[3];
        unsigned char PinCodeType[1];
        unsigned char Voice[2];
        unsigned char Codec[1];
        unsigned char Radio[6];
        unsigned char Sleep[7];
        unsigned char Reserved[2];
}nvram_ef_btradio_mt6611_struct;
```

Description:

Compose mt6611 BT data to a buffer. This function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-470 The return value of META_NVRAM_BT_Compose_MT6611Radio

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-471 The parameter of META_NVRAM_BT_Compose_MT6611Radio

Parameter	IN/OUT	Description
radio	IN	nvram_ef_btradio_mt6611_struct
buf	IN/OUT	Output buffer to be composed.
buf_len	IN	Buffer length

6.8.16.6 META_NVRAM_BT_Compose_MediatekBtChip

Definition:

META_RESULT __stdcall META_NVRAM_BT_Compose_MediatekBtChip(const nvram_ef_btradio_mtk_bt_chip_struct *radio, char *buf, const int buf_len);

Description:

Compose MediaTek BT data. Usually, once the buffer of audio coefficient data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1audio_param_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-472 The return value of META_NVRAM_BT_Compose_MediatekBtChip

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-473 The parameter of META_NVRAM_BT_Compose_MediatekBtChip

Parameter	IN/OUT	Description
radio	IN/OUT	Output MediaTek BT data
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.8.16.7 META_NVRAM_BT_Decompose_MediatekBtChip

Definition:

META_RESULT __stdcall META_NVRAM_BT_Decompose_MediatekBtChip(nvram_ef_btradio_mtk_bt_chip_struct *radio, const char *buf, const int buf_len);

Description:

Decompose MediaTek BT data. Usually, once the buffer of audio coefficient data is acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure l1audio_param_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-474 The return value of META_NVRAM_BT_Decompose_MediatekBtChip

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-475 The parameter of META_NVRAM_BT_Decompose_MediatekBtChip

Parameter	IN/OUT	Description
radio	IN/OUT	Output MediatTek BT data
buf	IN	Input buffer to decompose.
buf_len	IN	Size of buf

6.9 Exported Functions for Audio Testing

This section mentions the exported functions that are related to audio testing.

6.9.1 META_Audio_Query_ID

Definition:

META_RESULT __stdcall META_Audio_Query_ID(const META_AUDIO_QUERY_ID_CNF cnf_cb, short *token, void *usrData)

```
// audio testing result
typedef enum {
        AUD_RES_OK = 0,                        // OK
        AUD_RES_FAIL,                                  // General Fail
        AUD_RES_BUSY,                          // system busy
        AUD_RES_DISC_FULL,                             // Memory full
        AUD_RES_OPEN_FILE_FAIL,                        // open file fail
        AUD_RES_END_OF_FILE,                   // play finish
        AUD_ERR_PEER_BUF_ERROR = 0xFD,                 // peer buf error
        AUD_ERR_FILEPATH_ERROR = 0xFE,                 // filepath error
        AUD_ERR_FILEPATH_TOO_LONG = 0xFF               // filepath too long
} AUDIO_RESULT;
```

```
// default system embeded audio id query
typedef struct {
            unsigned short          MinRingTone_ID;
            unsigned short          MaxRingTone_ID;
            unsigned short          MinMIDI_ID;
            unsigned short          MaxMIDI_ID;
            unsigned short          MinSound_ID;
            unsigned short          MaxSound_ID;
            AUDIO_RESULT            status;
} Audio_Query_ID_Cnf;
```

Description:

This function is used to query the default system-embeded audio id.

Callback:

typedef void (__stdcall *META_AUDIO_QUERY_ID_CNF)(const Audio_Query_ID_Cnf *cnf, const short token, void *usrData);

Return Value:

Table 6-476 The return value of META_Audio_Query_ID

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_NO_MEMORY	Cannot allocate memory.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-477 The parameter of META_Audio_Query_ID

Parameter	IN/OUT	Description
cnf_cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
Token	IN/OUT	Token used by user to uninstall the confirmation and indication callback function.

Parameter	IN/OUT	Description
UsrData	IN	Parameter used by user.

6.9.2 META_Audio_Play

Definition:

```
        META_RESULT  __stdcall META_Audio_Play(
                                    const Audio_Play_Req  *req,
                                    const META_AUDIO_PLAY_CNF cnf_cb,
                                    const META_AUDIO_PLAY_OVER_IND  ind_cb,
                                    short *token, void *usrData)


// play style enum
typedef enum {
        FT_L4AUD_AUDIO_PLAY_CRESCENDO = 0,    // Play sound for crescendo.
        FT_L4AUD_AUDIO_PLAY_INFINITE,         // Play sound for infinite.
        FT_L4AUD_AUDIO_PLAY_ONCE,                     // Play sound for once.
        FT_L4AUD_AUDIO_PLAY_DESCENDO                  // Play sound for descendo.
} AUDIO_PLAY_STYLE;


// play default system embeded audio by the given audio id
typedef struct {
        unsigned short          audio_id;           // default system embeded audio id
        AUDIO_PLAY_STYLE        play_style;         // play style
}Audio_Play_Req;
```

Description:

This function is used to play audio by the given system-embeded audio id.

Callback:

typedef void (__stdcall *META_AUDIO_PLAY_CNF)(const AUDIO_RESULT status, const short token, void *usrData);
typedef void (__stdcall *META_AUDIO_PLAY_OVER_IND)(const AUDIO_RESULT status, const short token, void *usrData);

Return Value:

Table 6-478 The return value of META_Audio_Play

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_NO_MEMORY	Cannot allocate memory.
META_INVALID_ARGUMENTS	Invalid arguments.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-479 The parameter of META_Audio_Play

Parameter	IN/OUT	Description
req	IN	Request parameter
cnf_cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
ind_cb	IN	Indication callback function called by META_DLL, when META_DLL receives a indication from target.
token	IN/OUT	Token used by user to uninstall the confirmation and indication callback function.
usrData	IN	Parameter used by user.

6.9.3 META_Audio_Play_ByName

Definition:

```
META_RESULT __stdcall META_Audio_Play_ByName(
        const Audio_Play_ByName_Req  *req,
        const META_AUDIO_PLAY_BYNAME_CNF  cnf_cb,
        const META_AUDIO_PLAY_OVER_IND  ind_cb,
        short *token, void *usrData)
```

// play audio from FAT by the given filepath

typedef struct {

```
        const char              *fat_filepath;      // filepath on target FAT file system
        AUDIO_PLAY_STYLE        play_style;         // play style
}Audio_Play_ByName_Req;
```

Description:

This function is used to play audio file on target FAT file system.

Callback:

typedef void (__stdcall *META_AUDIO_PLAY_BYNAME_CNF)(const AUDIO_RESULT status, const short token, void *usrData);
typedef void (__stdcall *META_AUDIO_PLAY_OVER_IND)(const AUDIO_RESULT status, const short token, void *usrData);

Return Value:

Table 6-480 The return value of META_Audio_Play_ByName

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_NO_MEMORY	Cannot allocate memory.
META_INVALID_ARGUMENTS	Invalid arguments.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-481 The parameter of META_Audio_Play_ByName

Parameter	IN/OUT	Description
req	IN	Request parameter
cnf_cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
ind_cb	IN	Indication callback function called by META_DLL, when META_DLL receives a indication from target.
token	IN/OUT	Token used by user to uninstall the confirmation and indication callback function.
usrData	IN	Parameter used by user.

6.9.4 META_Audio_Play_IMY_ByBuf

Definition:

META_RESULT __stdcall META_Audio_Play_IMY_ByBuf(

```
                    const Audio_Play_IMY_ByBuf_Req  *req,
                    const META_AUDIO_PLAY_IMY_BYBUF_CNF  cnf_cb,
                    const META_AUDIO_PLAY_OVER_IND  ind_cb,
                    short *token, void *usrData)
```

```
// play imelody by the buffer from PC side
typedef struct {
        const char          *imy_buf;           // buffer that stores iMelody content
        unsigned int        imy_buf_len;        // length of buffer
        unsigned char       imy_instrument;     // instrument id, 1 ~ 128
        AUDIO_PLAY_STYLE    play_style;         // play style
}Audio_Play_IMY_ByBuf_Req;
```

Description:

This function is used to play iMelody. You can load your iMelody file into memory, then you use this function to send the iMelody content to target to play.

Callback:

typedef void (__stdcall *META_AUDIO_PLAY_IMY_BYBUF_CNF)(const AUDIO_RESULT status, const short token, void *usrData);

typedef void (__stdcall *META_AUDIO_PLAY_OVER_IND)(const AUDIO_RESULT status, const short token, void *usrData);

Return Value:

Table 6-482 The return value of META_Audio_Play_IMY_ByBuf

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_NO_MEMORY	Cannot allocate memory.
META_INVALID_ARGUMENTS	Invalid arguments.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-483 The parameter of META_Audio_Play_IMY_ByBuf

Parameter	IN/OUT	Description
req	IN	Request parameter
cnf_cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
ind_cb	IN	Indication callback function called by META_DLL, when META_DLL receives a indication from target.
token	IN/OUT	Token used by user to uninstall the confirmation and indication callback function.
usrData	IN	Parameter used by user.

6.9.5 META_Audio_Stop

Definition:

META_RESULT __stdcall META_Audio_Stop(const META_AUDIO_STOP_CNF cnf_cb, short *token, void *usrData)

Description:

This function is used to stop audio playing. When you issue the stop command, the play indication callback will return ,too.

Callback:

typedef void (__stdcall *META_AUDIO_STOP_CNF)(const AUDIO_RESULT status, const short token, void *usrData);

Return Value:

Table 6-484 The return value of META_Audio_Stop

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_NO_MEMORY	Cannot allocate memory.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-485 The parameter of META_Audio_Stop

Parameter	IN/OUT	Description
cnf_cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the confirmation and indication callback function.
usrData	IN	Parameter used by user.

6.9.6 META_Audio_MEDIA_Play

Definition:

```
META_RESULT  __stdcall META_Audio_MEDIA_Play(
                const Audio_MEDIA_Play_Req  *req,
                const META_AUDIO_MEDIA_PLAY_CNF  cnf_cb,
                const META_AUDIO_MEDIA_PLAY_OVER_IND  ind_cb,
                short *token, void *usrData)
```

```
// play mp3 from FAT by the given filepath
typedef struct {
        const char              *fat_filepath;   // filepath on target FAT file system
        AUDIO_PLAY_STYLE        play_style;      // play style
}Audio_MEDIA_Play_Req;
```

Description:

This function is used to play MP3 file on target FAT file system.

Callback:

typedef void (__stdcall *META_AUDIO_MEDIA_PLAY_CNF)(const AUDIO_RESULT status, const short token, void *usrData);

typedef void (__stdcall *META_AUDIO_MEDIA_PLAY_OVER_IND)(const AUDIO_RESULT status, const short token, void *usrData);

Return Value:

Table 6-486 The return value of META_Audio_MEDIA_Play

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_NO_MEMORY	Cannot allocate memory.
META_INVALID_ARGUMENTS	Invalid arguments.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-487 The parameter of META_Audio_MEDIA_Play

Parameter	IN/OUT	Description
req	IN	Request parameter
cnf_cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
ind_cb	IN	Indication callback function called by META_DLL, when META_DLL receives a indication from target.
Token	IN/OUT	Token used by user to uninstall the confirmation and indication callback function.
usrData	IN	Parameter used by user.

6.9.7 META_Audio_MEDIA_Stop

Definition:

META_RESULT __stdcall META_Audio_MEDIA_Stop(const META_AUDIO_MEDIA_STOP_CNF cnf_cb, short *token, void *usrData)

Description:

This function is used to stop MP3 playing. When you issue the stop command, the play indication callback will return ,too.

Callback:

typedef void (__stdcall *META_AUDIO_MEDIA_STOP_CNF)(const AUDIO_RESULT status, const short token, void *usrData);

Return Value:

Table 6-488 The return value of META_Audio_MEDIA_Stop

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_NO_MEMORY	Cannot allocate memory.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-489 The parameter of META_Audio_MEDIA_Stop

Parameter	IN/OUT	Description
cnf_cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the confirmation and indication callback function.
usrData	IN	Parameter used by user.

6.9.8 META_Audio_Set_Echo_Loop

Definition:

META_RESULT __stdcall META_Audio_Set_Echo_Loop(unsigned int ms_timeout,const Audio_Set_Echo_Req *req);

```
// set Echo Loop
typedef struct {
        unsigned char           echoflag;                     // 1 means true, 0 means false
}Audio_Set_Echo_Req;
```

Description:

This function is used to set Echo Loop under meta mode.

Callback:

typedef void (__stdcall *META_AUDIO_SET_ECHO_CNF)(const AUDIO_RESULT status, const short token, void *usrData);

6.9.9 META_Audio_Set_Mode

Definition:

META_RESULT __stdcall META_Audio_Set_Mode(unsigned int ms_timeout,const Audio_Set_Mode_Req *req);

```
typedef struct {
        unsigned char           modeflag;                     // modeflag
}Audio_Set_Mode_Req;
```

Description:

This function is used to set Audio mode under meta mode.

Type	Short name	Long name	Parameter/comment
Integer	mode	Audio mode	normal0 handset1 loudspeaker2

Return Value:

Table 6-490 The return value of META_Audio_Set_Mode

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_NO_MEMORY	Cannot allocate memory.
META_INVALID_ARGUMENTS	Invalid arguments.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-491 The parameter of META_Audio_Set_Mode

Parameter	IN/OUT	Description
req	IN	Request parameter
ms_timeout	IN	Function timeout value. (in milliseconds)

6.9.10 META_Audio_Set_Gain

Definition:

```
        META_RESULT  __stdcall META_Audio_Set_Gain(unsigned int  ms_timeout,const Audio_Set_Gain_Req
*req);


typedef struct {
        unsigned char                type;
        unsigned char                gain;
}Audio_Set_Gain_Req;
```

Description:

This function is used to set Audio Gain under meta mode.

Type	Short name	Long name	Parameter/comment
Integer	type	Audio type	call tone0 keypad tone1 microphone2 <reserved>3 speech sound4 side tone5 MP3, Wave, melody, I-melody, midi6
Integer	Gain	Gain value	0~255

Return Value:

Table 6-492 The return value of META_Audio_Set_Gain

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_NO_MEMORY	Cannot allocate memory.
META_INVALID_ARGUMENTS	Invalid arguments.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-493 The parameter of META_Audio_Set_Gain

Parameter	IN/OUT	Description
req	IN	Request parameter
ms_timeout	IN	Function timeout value. (in milliseconds)

6.9.11 META_Audio_Set_Volume

Definition:

```
        META_RESULT  __stdcall META_Audio_Set_Volume(
                        const Audio_Set_Volume_Req  *req,
                        const META_AUDIO_SET_VOLUME_CNF  cnf_cb,
                        short *token, void *usrData)
```

```
// set volume
typedef struct {
        unsigned char      volume;         // play volume, 0 ~ 6
}Audio_Set_Volume_Req;
```

Description:

This function is used to adjust output volume. You can also adjust volume value while audio is playing.

Callback:

typedef void (__stdcall *META_AUDIO_SET_VOLUME_CNF)(const AUDIO_RESULT status, const short token, void *usrData);

6.9.12 META_Audio_Tone_Loop_Back_Rec

Definition:

META_Audio_Tone_Loop_Back_Rec(unsigned int ms_timeout, Audio_Tone_LoopBackRec_Req *req, Audio_Tone_LoopBackRec_Cnf *cnf);

```
typedef struct {
        unsigned short          fre;
        unsigned char           spkgain;
        unsigned char           micgain;
        unsigned short    ulgain;
        unsigned short    dlgain;
        unsigned short      amp;
}Audio_Tone_LoopBackRec_Req;


typedef struct {
        unsigned int                buffer[2000];
}Audio_Tone_LoopBackRec_Cnf;
```

Description:

This function reads the address of PMIC register

Callback:

Return Value:

Table 6-494 The return value of META_Audio_Tone_Loop_Back_Rec

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-495 The parameter of META_Audio_Tone_Loop_Back_Rec

Parameter	IN/OUT	Description
req	IN	Audio_Tone_LoopBackRec_Req,includes frequency,spkgain,micgain, downlinkgain and uplink gain and amplifier.
Cnf	IN	Audio_Tone_LoopBackRec_Cnf, contains 8000bytes, the cnf will receive 2000bytes fourth times.
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.13 META_Audio_Set_LoudSpk_FIR_Coeffs

Definition:

META_Audio_Set_LoudSpk_FIR_Coeffs(unsigned int ms_timeout,const Audio_Set_LoudSpk_FIR_Coeffs_Req *req);

```
typedef struct {
        short           in_fir_coeffs[45];
        short           out_fir_coeffs[45];
}Audio_Set_LoudSpk_FIR_Coeffs_Req;
```

Description:

This function will set Loud Speak FIR Coefficient while run time setting parameter

Callback:

Return Value:

Table 6-496 The return value of META_Audio_Set_LoudSpk_FIR_Coeffs

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-497 The parameter of META_Audio_Set_LoudSpk_FIR_Coeffs

Parameter	IN/OUT	Description
req	IN	Audio_Set_LoudSpk_FIR_Coeffs_Req.
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.14 META_Audio_Set_Speech_Common

Definition:

```
META_Audio_Set_Speech_Common(unsigned int  ms_timeout,const Audio_Set_Speech_Common_Req  *req);
```

```
typedef struct {
        unsigned short speech_common_para[12];
}Audio_Set_Speech_Common_Req;
```

Description:

This function will set speech common parameter while run time

Callback:

Return Value:

Table 6-498 The return value of META_Audio_Set_Speech_Common

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-499 The parameter of META_Audio_Set_Speech_Common

Parameter	IN/OUT	Description
req	IN	Audio_Set_Speech_Common_Req
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.15 META_Audio_Set_LoudSpk_Mode

Definition:

```
META_Audio_Set_LoudSpk_Mode(unsigned int  ms_timeout,const Audio_Set_LoudSpk_Mode_Req  *req);


typedef struct {
        unsigned short speech_loudspk_mode_para[8];
}Audio_Set_LoudSpk_Mode_Req;
```

Description:

This function will set Loud Speak parameter while run time

Callback:

Return Value:

Table 6-500 The return value of META_Audio_Set_LoudSpk_Mode

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-501 The parameter of META_Audio_Set_LoudSpk_Mode

Parameter	IN/OUT	Description
req	IN	Audio_Set_LoudSpk_Mode_Req
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.16 META_Audio_Set_Playback_Maximum_Swing

Definition:

META_Audio_Set_Playback_Maximum_Swing(unsigned int ms_timeout,const Audio_Set_Playback_Maximum_Swing_Req *req);

```
typedef struct {
        unsigned short Media_Playback_Maximum_Swing;
}Audio_Set_Playback_Maximum_Swing_Req;
```

Description:

This function will set Playback Maximum_Swing_Req while run time

Callback:

Return Value:

Table 6-502 The return value of META_Audio_Set_Playback_Maximum_Swing

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-503 The parameter of META_Audio_Set_Playback_Maximum_Swing

Parameter	IN/OUT	Description
req	IN	Audio_Set_Playback_Maximum_Swing_Req
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.17 META_Audio_Set_Melody_FIR_Output_Coeffs

Definition:

META_Audio_Set_Melody_FIR_Output_Coeffs(unsigned int ms_timeout,const Audio_Set_Melody_FIR_Output_Coeffs_Req *req);

```
typedef struct {
        short Melody_FIR_Output_Coeff_32k_Tbl1[25];
}Audio_Set_Melody_FIR_Output_Coeffs_Req;
```

Description:

This function will set Melody_FIR_Output_Coeffs while run time

Callback:

Return Value:

Table 6-504 The return value of META_Audio_Set_Melody_FIR_Output_Coeffs

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-505 The parameter of META_Audio_Set_Melody_FIR_Output_Coeffs

Parameter	IN/OUT	Description
req	IN	Audio_Set_Melody_FIR_Output_Coeffs_Req
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.18 META_Audio_Set_Speech_Common_And_Mode

Definition:

META_Audio_Set_Speech_Common_And_Mode(unsigned int ms_timeout,const Audio_Set_Speech_Common_And_Mode_Req *req);

```
typedef struct {
        unsigned short speech_common_para[12];
        unsigned short speech_loudspk_mode_para[8];
}Audio_Set_Speech_Common_And_Mode_Req;
```

Description:

This function will set Audio_Set_Speech_Common_And_Mode_Req while run time

Return Value:

Table 6-506 The return value of META_Audio_Set_Speech_Common_And_Mode

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-507 The parameter of META_Audio_Set_Speech_Common_And_Mode

Parameter	IN/OUT	Description
req	IN	Audio_Set_Speech_Common_And_Mode_Req
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.19 META_Audio_Play_Freq_Vol_Tone

Definition:

META_RESULT __stdcall META_Audio_Play_Freq_Vol_Tone(unsigned int ms_timeout, const Audio_Set_Freq_Vol_Tone_Req_T *req);

```
typedef struct
{
        unsigned char  m_ucVolume;
        unsigned short m_u2Freq;
}Audio_Set_Freq_Vol_Tone_Req_T;
```

Description:

This function play tone with setting frequency and volume.

Return Value:

Table 6-508 The return value of META_Audio_Play_Freq_Vol_Tone

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-509 The parameter of META_Audio_Play_Freq_Vol_Tone

Parameter	IN/OUT	Description
req	IN	Audio_Set_Freq_Vol_Tone_Req_T,includes frequency and volume.
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.20 META_Audio_Stop_Freq_Vol_Tone

Definition:

```
META_RESULT __stdcall META_Audio_Stop_Freq_Vol_Tone(unsigned int  ms_timeout);
```

Description:

This function stop playing tone.

Return Value:

Table 6-510 The return value of META_Audio_Stop_Freq_Vol_Tone

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-511 The parameter of META_Audio_Stop_Freq_Vol_Tone

Parameter	IN/OUT	Description
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.21 META_Audio_Tone_Loop_Back_Rec_2K

Definition:

META_RESULT __stdcall META_Audio_Tone_Loop_Back_Rec_2K(unsigned int ms_timeout, Audio_Tone_LoopBackRec_Req *req, Audio_Tone_LoopBackRec_Cnf_2K *cnf);

```
typedef struct {
        unsigned short    fre;
        unsigned char     spkgain;
        unsigned char     micgain;
        unsigned short    ulgain;
        unsigned short    dlgain;
        unsigned short    amp;
}Audio_Tone_LoopBackRec_Req;

typedef struct {
        unsigned int            buffer[500];
}Audio_Tone_LoopBackRec_Cnf_2K;
```

Description:

This function will ask target to do tone loop back recording in loud-speaker mode with a 2k-bytes buffer.

Callback:

Return Value:

Table 6-512 The return value of META_Audio_Tone_Loop_Back_Rec_2K

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-513 The parameter of META_Audio_Tone_Loop_Back_Rec_2K

Parameter	IN/OUT	Description
req	IN	Audio_Tone_LoopBackRec_Req,includes frequency,spkgain,micgain, downlinkgain and uplink gain and amplifier.
Cnf	IN	Audio_Tone_LoopBackRec_Cnf_2K, contains 2000bytes
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.22 META_Audio_Tone_Loop_Back_Rec_2K_Normal

Definition:

```
META_RESULT __stdcall META_Audio_Tone_Loop_Back_Rec_2K_Normal(unsigned int ms_timeout, Audio_Tone_LoopBackRec_Req *req, Audio_Tone_LoopBackRec_Cnf_2K *cnf);


typedef struct {
        unsigned short   fre;
        unsigned char    spkgain;
        unsigned char    micgain;
        unsigned short   ulgain;
        unsigned short   dlgain;
        unsigned short   amp;
}Audio_Tone_LoopBackRec_Req;


typedef struct {
        unsigned int                    buffer[500];
}Audio_Tone_LoopBackRec_Cnf_2K;
```

Description:

This function will ask target to do tone loop back recording in normal-speaker mode with a 2k-bytes buffer.

Callback:

Return Value:

Table 6-514 The return value of META_Audio_Tone_Loop_Back_Rec_2K_Normal

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-515 The parameter of META_Audio_Tone_Loop_Back_Rec_2K_Normal

Parameter	IN/OUT	Description
req	IN	Audio_Tone_LoopBackRec_Req,includes frequency,spkgain,micgain, downlinkgain and uplink gain and amplifier.
Cnf	IN	Audio_Tone_LoopBackRec_Cnf_2K, contains 2000bytes
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.23 META_Audio_Get_Audio_Profile_Settings

Definition:

META_RESULT __stdcall META_Audio_Get_Audio_Profile_Settings(unsigned int ms_timeout, Audio_Get_Profile_Settings_By_Mode_Req_T *req, Audio_Get_Profile_Settings_By_Mode_Cnf_T *cnf);

```
typedef struct
{
    unsigned char  m_ucMode;
}Audio_Get_Profile_Settings_By_Mode_Req_T;
```

```
typedef struct
{
    unsigned char mode;
   unsigned char melody[7];
   unsigned char sound[7];
   unsigned char keytone[7];
   unsigned char speech[7];
   unsigned char mic[7];
   unsigned char sidetone;
   unsigned char max_melody_volume_gain;
   unsigned char melody_volume_gain_step;
   unsigned char tv_out_volume_gain[MAX_VOL_LEVEL];  // 7 here

}Audio_Get_Profile_Settings_By_Mode_Cnf_T;
```

Description:

This function will query target's audio profile settings by mode. (Only support mode 0, 1, 2)

Callback:

Return Value:

Table 6-516 The return value of META_Audio_Get_Audio_Profile_Settings

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-517 The parameter of META_Audio_Get_Audio_Profile_Settings

Parameter	IN/OUT	Description
req	IN	Audio_Get_Profile_Settings_By_Mode_Req_T, includes mode.
Cnf	IN	Audio_Get_Profile_Settings_By_Mode_Cnf_T, includes mode, melody, sound, keytone, speech, microphone, sidetone, max_melody_volume_gain, melody_volume_gain_step, tv_out_volume_gain;

Parameter	IN/OUT	Description
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.24 META_Audio_Set_Audio_Profile_Settings

Definition:

META_RESULT __stdcall META_Audio_Set_Audio_Profile_Settings(unsigned int ms_timeout, Audio_Set_Profile_Settings_By_Mode_Req_T *req, Audio_Set_Profile_Settings_By_Mode_Cnf_T *cnf);

```
typedef struct
{
    unsigned char mode;
   unsigned char melody[7];
   unsigned char sound[7];
   unsigned char keytone[7];
   unsigned char speech[7];
   unsigned char mic[7];
   unsigned char sidetone;
   unsigned char max_melody_volume_gain;
   unsigned char melody_volume_gain_step;
   unsigned char tv_out_volume_gain[MAX_VOL_LEVEL];  // 7 here

}Audio_Set_Profile_Settings_By_Mode_Req_T;

typedef struct
{
   unsigned short m_u2FailReason;  // possible fail resons

}Audio_Set_Profile_Settings_By_Mode_Cnf_T;
```

Description:

This function will set target's audio profile settings by mode. (Only support mode 0, 1, 2)

Callback:

Return Value:

Table 6-518 The return value of META_Audio_Set_Audio_Profile_Settings

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-519 The parameter of META_Audio_Set_Audio_Profile_Settings

Parameter	IN/OUT	Description
req	IN	Audio_Set_Profile_Settings_By_Mode_Req_T, includes mode, melody, sound, keytone, speech, microphone, sidetone, max_melody_volume_gain, melody_volume_gain_step, tv_out_volume_gain.
Cnf	IN	Audio_Set_Profile_Settings_By_Mode_Cnf_T, includes m_u2FailReason for error code.
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.25 META_Audio_Get_Audio_Param_Settings_0809

Definition:

META_RESULT __stdcall META_Audio_Get_Audio_Param_Settings_0809(unsigned int ms_timeout, l1audio_param_W0809_T *cnf);

```
typedef struct
{
  unsigned char mode;
  unsigned char melody[7];
  unsigned char sound[7];
  unsigned char keytone[7];
  unsigned char speech[7];
  unsigned char mic[7];
```

```
    unsigned char sidetone;

    unsigned char max_melody_volume_gain;

    unsigned char melody_volume_gain_step;

    unsigned char tv_out_volume_gain[MAX_VOL_LEVEL];  // 7 here


}Audio_Set_Profile_Settings_By_Mode_Req_T;


typedef struct

{

    unsigned short m_u2FailReason;  // possible fail resons


}Audio_Set_Profile_Settings_By_Mode_Cnf_T;
```

Description:

This function will set target’s audio profile settings by mode. (Only support mode 0, 1, 2)

Callback:

Return Value:

Table 6-520 The return value of META_Audio_Get_Audio_Param_Settings_0809

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-521 The parameter of META_Audio_Get_Audio_Param_Settings_0809

Parameter	IN/OUT	Description
req	IN	Audio_Set_Profile_Settings_By_Mode_Req_T, includes mode, melody, sound, keytone, speech, microphone, sidetone, max_melody_volume_gain, melody_volume_gain_step, tv_out_volume_gain.
Cnf	IN	Audio_Set_Profile_Settings_By_Mode_Cnf_T, includes m_u2FailReason for error code.
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.26 META_Audio_Set_Output_Dev

Definition:

META_RESULT __stdcall META_Audio_Set_Output_Dev(unsigned int ms_timeout, unsigned char *output_dev_req);

META_RESULT __stdcall META_Audio_Set_Output_Dev_r(const int meta_handle, unsigned int ms_timeout, unsigned char *output_dev_req);

Description:

This function will set target's audio output device.
0: Handset, 1:Headset, 2: Handsfree.

Callback:

Return Value:

Table 6-522 The return value of META_Audio_Set_Output_Dev

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-523 The parameter of META_Audio_Set_Output_Dev

Parameter	IN/OUT	Description
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.
output_dev_req	IN	The requested audio output device.

6.9.27 META_Audio_Set_Output_Vol

Definition:

META_RESULT __stdcall META_Audio_Set_Output_Vol(unsigned int ms_timeout, unsigned char *output_vol);

META_RESULT __stdcall META_Audio_Set_Output_Vol_r(const int meta_handle, unsigned int ms_timeout, unsigned char *output_vol);

Description:

This function will set target's output volume of current output device.

Callback:

Return Value:

Table 6-524 The return value of META_Audio_Set_Output_Vol

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-525 The parameter of META_Audio_Set_Output_Vol

Parameter	IN/OUT	Description
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.
output_vol	IN	The requested audio output volume.

6.9.28 META_Audio_FreeMemory

Definition:

META_RESULT __stdcall META_Audio_FreeMemory(unsigned int ms_timeout);

META_RESULT __stdcall META_Audio_FreeMemory_r(const int meta_handle, unsigned int ms_timeout);

Description:

This function will free the allocated memory for audio playing on the target.

Callback:

Return Value:

Table 6-526 The return value of META_Audio_FreeMemory

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-527 The parameter of META_Audio_FreeMemory

Parameter	IN/OUT	Description
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.29 META_Audio_PlayCurMemContent

Definition:

META_RESULT __stdcall META_Audio_PlayCurMemContent(unsigned int ms_timeout);

META_RESULT __stdcall META_Audio_PlayCurMemContent_r(const int meta_handle, unsigned int ms_timeout);

Description:

This function will intruct the target to play the current content in the allocated memory on the target.

Callback:

Return Value:

Table 6-528 The return value of META_Audio_PlayCurMemContent

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-529 The parameter of META_Audio_PlayCurMemContent

Parameter	IN/OUT	Description
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.30 META_Audio_StopPlaying

Definition:

META_RESULT __stdcall META_Audio_StopPlaying(unsigned int ms_timeout);

META_RESULT __stdcall META_Audio_StopPlaying_r(const int meta_handle, unsigned int ms_timeout);

Description:

This function will instruct the target to stop playing the audio.

Callback:

Return Value:

Table 6-530 The return value of META_Audio_StopPlaying

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-531 The parameter of META_Audio_StopPlaying

Parameter	IN/OUT	Description
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.31 META_Audio_Play_Wave_File

Definition:

META_RESULT __stdcall META_Audio_Play_Wave_File(unsigned int ms_timeout, Audio_Play_Wave_File_REQ_T *req, int *pStopFlag, bool *bSaveAllOnTargetMem);

META_RESULT __stdcall META_Audio_Play_Wave_File_r(const int meta_handle, unsigned int ms_timeout, Audio_Play_Wave_File_REQ_T *req, int *pStopFlag, bool *bSaveAllOnTargetMem);

```
typedef struct
{

  bool bCheckHdr;
  unsigned int u4StartFilePos; // only valid when bCheckHdr = false;

  char *pFilePath;

  bool bIsStereo;
  char i1BitPerSample;
  unsigned short u2SampleFreq;
  bool  bForceVoice;     // always set true
  CALLBACK_META_AUDIO_PROGRESS  cb_progress;

  void  *cb_progress_arg;

}Audio_Play_Wave_File_REQ_T;
```

Description:

This function will stream a PCM wave file to target, and ask target to play it.

Callback:

Return Value:

Table 6-532 The return value of META_Audio_Play_Wave_File

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-533 The parameter of META_Audio_Play_Wave_File

Parameter	IN/OUT	Description
req	IN	Audio_Play_Wave_File_REQ_T, includes bCheckHdr, u4StartFilePos, pFilePath, bTsStereo, i1BitPerSample, u2SampleFreq, bForceVoice, cb_progress, and cb_progress_arg
pStopFlag	IN	Indication of stop playing.
bSaveAllOnTargetMem	IN	Indication of whether the content will all be saved on target's memory.
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.32 META_Audio_EX_SetACFIIRToTargetEx

Definition:

META_RESULT __stdcall META_Audio_EX_SetACFIIRToTargetEx(const unsigned int ms_timeout, const Audio_Ex_SetACFToTarget_REQ_EX_T *req);

META_RESULT __stdcall META_Audio_EX_SetACFIIRToTargetEx_r(const int meta_handle, const unsigned int ms_timeout, const Audio_Ex_SetACFToTarget_REQ_EX_T *req);

```
typedef struct
{
   /// the buffer for the compose function sink
   char        buffer[2000];
   /// the buffer length (must be retrieved by META_NVRAM_AudioBesLoudNess_Len)
   unsigned int bufferLength;
}Audio_Ex_SetACFToTarget_REQ_EX_T;
```

Description:

This function will set the runtime IIR coefficient to the target for audio compensation - loud speaker application.The buffer length must be retrieved by the "*META_NVRAM_AudioBesLoudNess_Len*" API. All of the buffer operation should be composed/decomposed by "*META_NVRAM_Compose_AudioBesLoudNess*" and "*META_NVRAM_Decompose_AudioBesLoudNess*" to/from the buffer.

Return Value:

Table 6-534 The return value of META_Audio_EX_SetACFIIRToTargetEx

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-535 The parameter of META_Audio_EX_SetACFIIRToTargetEx

Parameter	IN/OUT	Description
req	IN	The input structure contains 2 parts, buffer and buffer length. The buffer part is the runtime coefficient (at most 2000 bytes).
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.33 META_Audio_EX_SetACFilterCoefEx

Definition:

META_RESULT __stdcall META_Audio_EX_SetACFilterCoefEx(unsigned int ms_timeout, const Audio_Ex_SetACFToTarget_REQ_EX_T *p_req);

META_RESULT __stdcall META_Audio_EX_SetACFIIRToTargetEx_r(const int meta_handle, const unsigned int ms_timeout, const Audio_Ex_SetACFToTarget_REQ_EX_T *req);

```
typedef struct
{
   /// the buffer for the compose function sink
   char        buffer[2000];
   /// the buffer length (must be retrieved by META_NVRAM_AudioBesLoudNess_Len)
   unsigned int bufferLength;
}Audio_Ex_SetACFToTarget_REQ_EX_T;
```

Description:

This function will set the runtime audio compensation filter coefficient to the target for audio compensation.

Return Value:

Table 6-536 The return value of META_Audio_EX_SetACFilterCoefEx

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-537 The parameter of META_Audio_EX_SetACFilterCoefEx

Parameter	IN/OUT	Description
req	IN	The input structure contains 2 parts, buffer and buffer length. The buffer part is the runtime coefficient (at most 2000 bytes).
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.34 META_Audio_EX_StartRecording

Definition:

META_RESULT __stdcall META_Audio_EX_StartRecording(unsigned int ms_timeout, const Audio_Ex_RecordingParam_T *param);

META_RESULT __stdcall META_Audio_EX_StartRecording_r(const int meta_handle, unsigned int ms_timeout, const Audio_Ex_RecordingParam_T *param);

```
typedef struct
{
   /// format MEDIA_FORMAT_WAV_DVI_ADPCM (narrow-band), MEDIA_FORMAT_WAV_DVI_ADPCM_16K (wide-band)
   unsigned int   fmt;
   /// parameter (0: for MEDIA_FORMAT_WAV_DVI_ADPCM/MEDIA_FORMAT_WAV_DVI_ADPCM_16K)
   unsigned short param;
   /// requested time(ms)
   unsigned int   requested_time;
   /// [IN/OUT] file path of target (set all the buffer to NULL means the target will create file on its own)
   char           file_path[512];
}Audio_Ex_RecordingParam_T;
```

Description:

This function is used in Dual-mic. NR calibration flow for recording VM file to request for starting recording

Return Value:

Table 6-538 The return value of META_Audio_EX_StartRecording

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-539 The parameter of META_Audio_EX_StartRecording

Parameter	IN/OUT	Description
req	IN	Input request parameter
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.35 META_Audio_EX_StopRecording

Definition:

META_RESULT __stdcall META_Audio_EX_StopRecording(unsigned int ms_timeout, const Audio_Ex_StopRecording_T * req);

META_RESULT __stdcall META_Audio_EX_StopRecording_r(const int meta_handle, unsigned int ms_timeout, const Audio_Ex_StopRecording_T * req);

```
typedef struct
{
  /// file path of target
  char    target_path[512];
  /// file path of local
  char    local_path[512];
  /// get file from target or not
  int     get_file;
  /// delete target side file or not
  int     delete_file;
  /// progress callback
  CALLBACK_META_FAT_PROGRESS cb;
  /// stop flag
  int     stop_flag;
}Audio_Ex_StopRecording_T;
```

Description:

This function is used in Dual-mic. NR calibration flow for recording VM file to request for stop recording.

Return Value:

Table 6-540 The return value of META_Audio_EX_StopRecording

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-541 The parameter of META_Audio_EX_StopRecording

Parameter	IN/OUT	Description
req	IN	Input request parameter
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.9.36 META_Audio_EX_QueryRecording

Definition:

META_RESULT __stdcall META_Audio_EX_QueryRecording(unsigned int ms_timeout, Audio_Ex_QueryRecording_T *status);

META_RESULT __stdcall META_Audio_EX_QueryRecording_r(const int meta_handle, unsigned int ms_timeout, Audio_Ex_QueryRecording_T *status);

```
typedef struct
{
    /// requested time(ms)
    unsigned int requested_time;
    /// recorded time(ms)
    unsigned int offset;
}Audio_Ex_QueryRecording_T;
```

Description:

This function is used in Dual-mic. NR calibration flow for recording VM file to query the current recording progress.

Return Value:

Table 6-542 The return value of META_Audio_EX_QueryRecording

Return value	Description
META_SUCCESS	SUCCESS
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-543 The parameter of META_Audio_EX_QueryRecording

Parameter	IN/OUT	Description
req	IN	Input request parameter
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event.

6.10 Exported Functions for Base Band Testing

6.10.1 META_BB_RegRead

Definition:

META_RESULT __stdcall META_BB_RegRead(RegRead_Req req, META_ BB_READREG_CNF cb, short *token, void *usrData)

```
typedef struct
{
        unsigned int        addr;               // The address of register that is to be read.
} RegRead_Req;
typedef struct
{
        unsigned short      value;              // The read back value
        unsigned char       status;             // 0: success, others: read register fail.
} RegRead_Cnf;
```

Description:

This function reads the value of a register that is specified in the addr.

Callback:

typedef void (__stdcall *META_BB_READREG_CNF)(RegRead_Cnf result, short token, void *usrData);

Return Value:

Table 6-544 The return value of META_BB_RegRead

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-545 The parameter of META_BB_RegRead

Parameter	IN/OUT	Description
req	IN	Specified the register that is to be read
cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the confirmation callback function.
usrData	IN	Parameter used by user.

6.10.2 META_BB_RegWrite

Definition:

META_RESULT __stdcall META_BB_RegWrite(RegWrite_Req req, META_BB_WRITEREG_CNF cb, short *token, void *usrData)

```
typedef struct
{
        unsigned int            addr;    // The address of register that is to be written.
        unsigned short          value;   // The value that is to be written.
} RegWrite_Req;
typedef struct
{
        unsigned char     status;        // 0: success, others: write register fail.
} RegWrite_Cnf;
```

Description:

This function reads the value of a register that is specified in the addr.

Callback:

typedef void (__stdcall *META_BB_WRITEREG_CNF)(RegWrite_Cnf result, short token, void *usrData);

Return Value:

Table 6-546 The return value of META_BB_RegWrite

Return value	Description
META_SUCCESS	SUCCESS

Return value	Description
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-547 The parameter of META_BB_RegWrite

Parameter	IN/OUT	Description
req	IN	Specified the register that is to be written.
cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the confirmation callback function.
usrData	IN	Parameter used by user.

6.10.3 META_BB_ADCGetMeaSumData

Definition:

```
        META_RESULT __stdcall META_BB_ADCGetMeaSumData(
                                        ADCMeaData_Req req,
                                        META_ BB_ADCGETMEADATA_CNF cb,
                                        short *token, void *usrData)
typedef struct
{
        unsigned char           channel;        // ADC channel number.
        unsigned short          Meacount;       // Number of measure times.
} ADCMeaData_Req;
typedef struct
{
        unsigned int    value;          // ADC value, it a sum value of each measurement data.
        unsigned char   status;         // 0: success, others: get ADC measurement fail.
} ADCMeaData_Cnf;
```

Description:

This function reads the sum value of each measurement data.

Callback:

typedef void (__stdcall *META_BB_ADCGETMEADATA_CNF)(ADCMeaData_Cnf result, short token, void *usrData);

Return Value:

Table 6-548 The return value of META_BB_ADCGetMeaSumData

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-549 The parameter of META_BB_ADCGetMeaSumData

Parameter	IN/OUT	Description
req	IN	Specified the channel that is tested.
cb	IN	Confirmation callback function called by META_DLL, when META_DLL receives a confirmation from target.
token	IN/OUT	Token used by user to uninstall the confirmation callback function.
usrData	IN	Parameter used by user.

6.10.4 META_BB_ADCGetMeaSumData_Ex

Definition:

META_BB_ADCGetMeaSumData_Ex(const unsigned int ms_timeout, const ADCMeaData_Req *req, ADCMeaData_Cnf *cnf);

```
typedef struct
{
        unsigned char           channel;        // ADC channel number.
        unsigned short          Meacount;       // Number of measure times.
} ADCMeaData_Req;
typedef struct
{
        unsigned int    value;          // ADC value, it a sum value of each measurement data.
        unsigned char   status;         // 0: success, others: get ADC measurement fail.
```

} ADCMeaData_Cnf;

Description:

This function reads the sum value of each measurement data.

Callback:

N/A

Return Value:

Table 6-550 The return value of META_BB_ADCGetMeaSumData_Ex

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-551 The parameter of META_BB_ADCGetMeaSumData_Ex

Parameter	IN/OUT	Description
req	IN	Specified the channel that is tested.
Cnf	OUT	ADC measurement result

6.10.5 META_PMIC_RegRead

Definition:

```
META_PMIC_RegRead(unsigned int ms_timeout,const RegRead_Req *req, RegRead_Cnf *cnf)

typedef struct
{
        unsigned int      addr;            // The address of register that is to be read.
} RegRead_Req;


typedef struct
{
        unsigned short    value;           // The read back value
        unsigned char     status;          // 0: success, others: read register fail.
```

} RegRead_Cnf;

Description:

This function reads the address of PMIC register

Callback:

Return Value:

Table 6-552 The return value of META_PMIC_RegRead

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-553 The parameter of META_PMIC_RegRead

Parameter	IN/OUT	Description
req	IN	Specified the address of Register
Cnf	IN	Specified the status and value of Register
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event

6.10.6 META_PMIC_RegWrite

Definition:

```
META_PMIC_RegWrite(unsigned int ms_timeout,const RegWrite_Req *req, RegWrite_Cnf *cnf)

typedef struct

{

        unsigned int        addr;               // The address of register that is to be written.

        unsigned short      value;              // The value that is to be written.

} RegWrite_Req;


typedef struct

{
        unsigned char      status;              // 0: success, others: write register fail.
} RegWrite_Cnf;
```

Description:

This function writes the value of the address of PMIC register

Callback:

Return Value:

Table 6-554 The return value of META_PMIC_RegWrite

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Memory is not enough.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.

Parameter:

Table 6-555 The parameter of META_PMIC_RegWrite

Parameter	IN/OUT	Description
req	IN	Specified the address and value of Register
Cnf	IN	Specified the status after writing value of Register
ms_timeout	IN	The unit is millisecond, after ms_timeout, the dll will catch a timeout event

6.11 Exported Functions for Target FAT File System Operation

6.11.1 META_FAT_Open

Definition:

```
    META_RESULT  __stdcall META_FAT_Open(
                                const char * fat_filepath,
                                FAT_OPEN_MODE  mode,
                                int *fs_handle,
                                short *p_token)
typedef enum {
        FAT_OPEN_READ = 0,
        FAT_OPEN_WRITE
}FAT_OPEN_MODE;
```

Description:

Open file for read/write on target FAT file system.

Return Value:

Table 6-556 The return value of META_FAT_Open

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.

Parameter:

Table 6-557 The parameter of META_FAT_Open

Parameter	IN/OUT	Description
fat_filepath	IN	The filepath that you want to open on target FAT system. Ex: "c:\def_sound\sound1.mid" (case insensitive)
mode	IN	Mode of open file, please refer to the definition of FAT_OPEN_MODE enum.
fs_handle	IN/OUT	Pointer to the file handle that is returned from target side.
token	IN/OUT	Token value for this operation.

6.11.2 META_FAT_Close

Definition:

META_RESULT __stdcall META_FAT_Close(int *fs_handle, short *p_token)

Description:

Close file on target FAT file system by fs_handle.

Return Value:

Table 6-558 The return value of META_FAT_Close

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.

Parameter:

Table 6-559 The parameter of META_FAT_Close

Parameter	IN/OUT	Description
fs_handle	IN/OUT	Pointer to the file handle which is created by META_FAT_Open(). If file handle was closed successfully, it will be set to –1.
token	IN/OUT	Token value for this operation.

6.11.3 META_FAT_GetFileSize

Definition:

META_RESULT __stdcall META_FAT_GetFileSize(const int fs_handle, int *filesize, short *p_token)

Description:

Get file size on target FAT file system by fs_handle.

This API can only work with the file opened by FAT_OPEN_READ mode, for FAT_OPEN_WRITE mode there is no filesize.

Return Value:

Table 6-560 The return value of META_FAT_GetFileSize

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.

Parameter:

Table 6-561 The parameter of META_FAT_GetFileSize

Parameter	IN/OUT	Description
fs_handle	IN	File handle which is created by META_FAT_Open().
filesize	IN/OUT	File size returned from target.
token	IN/OUT	Token value for this operation.

6.11.4 META_FAT_Read

Definition:

```
META_RESULT  __stdcall META_FAT_Read(
                    const int fs_handle,
                    char *buf, const int buf_len,
                    CALLBACK_META_FAT_PROGRESS   cb_progress,
                    void  *cb_progress_arg,
                    short *p_token)
```

Description:

Read file from target FAT file system into a buffer.

Callback:

typedef int (__stdcall *CALLBACK_META_FAT_PROGRESS)(unsigned char percent, int sent_bytes, int total_bytes, const short token, void *usr_arg);

This callback function will be invoked during reading progress; you can use this callback function to get the finish percentage, sent_bytes and total_bytes.

Return Value:

Table 6-562 The return value of META_FAT_Read

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.
META_BUFFER_LEN	Read length of data exceeds buffer length.

Parameter:

Table 6-563 The parameter of META_FAT_Read

Parameter	IN/OUT	Description
fs_handle	IN	File handle which is created by META_FAT_Open().
buf	IN/OUT	Buffer to store read data.
buf_len	IN	Buffer length.
cb_progress	IN	Function pointer of progress callback.
cb_progress_arg	IN	User argument that will be used into callback function.
token	IN/OUT	Token value for this operation.

6.11.5 META_FAT_Write

Definition:

```
META_RESULT __stdcall META_FAT_Write(
                    const int fs_handle,
                    const char *buf, const int buf_len,
                    CALLBACK_META_FAT_PROGRESS   cb_progress,
                    void  *cb_progress_arg,
                    short *p_token)
```

Description:

Write data of buffer into the file on target FAT file system.

Callback:

typedef int (__stdcall *CALLBACK_META_FAT_PROGRESS)(unsigned char percent, int sent_bytes, int total_bytes, const short token, void *usr_arg);

This callback function will be invoked during reading progress; you can use this callback function to get the finish percentage, sent_bytes and total_bytes.

Return Value:

Table 6-564 The return value of META_FAT_Write

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.

Parameter:

Table 6-565 The parameter of META_FAT_Write

Parameter	IN/OUT	Description
fs_handle	IN	File handle which is created by META_FAT_Open().
buf	IN/OUT	Buffer to store write data.
buf_len	IN	Buffer length.
cb_progress	IN	Function pointer of progress callback.
cb_progress_arg	IN	User argument that will be used into callback function.
token	IN/OUT	Token value for this operation.

6.11.6 META_FAT_Read_To_File

Definition:

```
META_RESULT  __stdcall META_FAT_Read_To_File(
                    const int fs_handle,
                    const char *local_filepath,
                    CALLBACK_META_FAT_PROGRESS  cb_progress,
                    void  *cb_progress_arg,
```

short *p_token)

Description:

Read file from target FAT file system into the file on the local disk.

Callback:

typedef int (__stdcall *CALLBACK_META_FAT_PROGRESS)(unsigned char percent, int sent_bytes, int total_bytes, const short token, void *usr_arg);

This callback function will be invoked during reading progress, you can use this callback function to get the finish percentage, sent_bytes and total_bytes.

Return Value:

Table 6-566 The return value of META_FAT_Read_To_File

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.
META_BUFFER_LEN	Read length of data exceeds buffer length.

Parameter:

Table 6-567 The parameter of META_FAT_Read_To_File

Parameter	IN/OUT	Description
fs_handle	IN	File handle which is created by META_FAT_Open().
local_filepath	IN	Local filepath to store the content of read data.
cb_progress	IN	Function pointer of progress callback.
cb_progress_arg	IN	User argument that will be used into callback function.
token	IN/OUT	Token value for this operation.

6.11.7 META_FAT_Write_By_File

Definition:

META_RESULT __stdcall META_FAT_Write_By_File(

const int fs_handle,

const char *local_filepath,

CALLBACK_META_FAT_PROGRESS cb_progress,

void *cb_progress_arg,

short *p_token)

Description:

Write the content of local file into the file on target FAT file system.

Callback:

typedef int (__stdcall *CALLBACK_META_FAT_PROGRESS)(unsigned char percent, int sent_bytes, int total_bytes, const short token, void *usr_arg);

This callback function will be invoked during reading progress; you can use this callback function to get the finish percentage, sent_bytes and total_bytes.

Return Value:

Table 6-568 The return value of META_FAT_Write_By_File

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.
META_FILE_BAD	The local file can't open for read, or file length is zero.

Parameter:

Table 6-569 The parameter of META_FAT_Write_By_File

Parameter	IN/OUT	Description
fs_handle	IN	File handle which is created by META_FAT_Open().
local_filepath	IN	Local filepath to read the content of written data.
cb_progress	IN	Function pointer of progress callback.
cb_progress_arg	IN	User argument that will be used into callback function.
token	IN/OUT	Token value for this operation.

6.11.8 META_FAT_Delete

Definition:

META_RESULT __stdcall META_FAT_Delete(const char *fa t_filepath, short *p_token)

Description:

Delete a remote file on target FAT file system by the given absolute FAT file path.

Return Value:

Table 6-570 The return value of META_FAT_Delete

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.

Parameter:

Table 6-571 The parameter of META_FAT_Delete

Parameter	IN/OUT	Description
fat_filepath	IN	The absolute FAT file path that you want to delete.
p_token	IN/OUT	Token value for this operation.

6.11.9 META_FAT_Move

Definition:

META_RESULT __stdcall META_FAT_Move(const char *fat_filepath, const char *new_fat_filepath, short *p_token)

Description:

Delete a remote file on target FAT file system by the given absolute FAT file path. Notice that this function is not supported in ULC project and wiil return error code (META_FAT_ACTION_NOT_SUPPORT

Return Value:

Table 6-572 The return value of META_FAT_Move

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.
META_FAT_ACTION_NOT_SUPPORT	This function is not supported.

Parameter:

Table 6-573 The parameter of META_FAT_Move

Parameter	IN/OUT	Description
fat_filepath	IN	The absolute FAT file path that you want to delete.
new_fat_filepath	IN	The new FAT file path you where want to move to. If the given new_fat_filepath doesn't contain the path, only present the filename, the original file will be rename as new filename under the original directory.
p_token	IN/OUT	Token value for this operation.

6.11.10 META_FAT_Find_Start

Definition:

```
        META_RESULT  __stdcall META_FAT_Find_Start(
                          const char *fat_base_dir,
                           const char *fat_find_pattern,
                           FAT_FIND_MODE  find_mode,
                           int *p_find_handle,
                           short *p_token)
```

```
typedef enum {
        FAT_FIND_FILE = 0,
        FAT_FIND_FILE_RECURSIVE,
        FAT_FIND_DIR_RECURSIVE
} FAT_FIND_MODE;
```

Description:

This function is used to search files or directories on target FAT file system. If there is any satisfied item found, this function will allocate a find_handle that is a found list to store all the found items, don't forget to call **META_FAT_Find_Close** to release the find_handle at the last.

Return Value:

Table 6-574 The return value of META_FAT_Find_Start

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.
META_FAT_NOT_FOUND	No matched item found by the given search pattern.

Parameter:

Table 6-575 The parameter of META_FAT_Find_Start

Parameter	IN/OUT	Description
fat_base_dir	IN	The search directory. Be sure that it must contain the drive letter, such as "C:\Temp".
fat_find_pattern	IN	Search pattern, it could contain the wildcard character. For example: "*.mid".
find_mode	IN	Search mode: FAT_FIND_FILE: Search files in the given directory. FAT_FIND_FILE_RECURSIVE: Recursively search files from the given directory. FAT_FIND_DIR_RECURSIVE: Recursively search directory from the given directory, fat_find_pattern takes no effect in this mode.
p_find_handle	OUT	If any target file or directory is found, it will return the handle of found list. You can use this handle to traverse the found list.
p_token	IN/OUT	Token value for this operation.

6.11.11 META_FAT_Find_Head

Definition:

META_RESULT __stdcall META_FAT_Find_Head(int find_handle)

Description:

This function is move the handle to the head of found list.

Return Value:

Table 6-576 The return value of META_FAT_Find_Head

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Failed to move handle to the head.
META_INVALID_ARGUMENTS	Invalid arguments.

Parameter:

Table 6-577 The parameter of META_FAT_Find_Head

Parameter	IN/OUT	Description
find_handle	IN	The handle of the found list.

6.11.12 META_FAT_Find_Prev

Definition:

META_RESULT __stdcall META_FAT_Find_Prev(int find_handle)

Description:

This function is move the handle to the previous found item.

Return Value:

Table 6-578 The return value of META_FAT_Find_Prev

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Failed to move handle to the previous found item.
META_INVALID_ARGUMENTS	Invalid arguments.

Parameter: The parameter of META_FAT_Find_Prev

Parameter	IN/OUT	Description
find_handle	IN	The handle of the found list.

6.11.13 META_FAT_Find_Next

Definition:

META_RESULT __stdcall META_FAT_Find_Next(int find_handle)

Description:

This function is move the handle to the next found item.

Return Value:

Table 6-579 The return value of META_FAT_Find_Next

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Failed to move handle to the next found item.
META_INVALID_ARGUMENTS	Invalid arguments.

Parameter:

Table 6-580 The parameter of META_FAT_Find_Next

Parameter	IN/OUT	Description
find_handle	IN	The handle of the found list.

6.11.14 META_FAT_Find_GetFileInfo

Definition:

```
META_RESULT __stdcall META_FAT_Find_GetFileInfo(
                int find_handle,
                char *p_filepath,
                const int filepath_len,
                int *p_filesize)
```

Description:

This function is to retrieve the filepath (the filename with path) and filesize of current found item.

Return Value:

Table 6-581 The return value of META_FAT_Find_GetFileInfo

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	Failed to get fileinfo.
META_INVALID_ARGUMENTS	Invalid arguments.

Parameter:

Table 6-582 The parameter of META_FAT_Find_GetFileInfo

Parameter	IN/OUT	Description
find_handle	IN	The handle of the found list.
p_filepath	OUT	The pointer to the buffer that you want to store the filepath.
filepath_len	IN	The length of buffer that you want to store the filepath. The length includes NULL terminated character.
p_filesize	OUT	The filesize of the current found item.

6.11.15 META_FAT_Find_Close

Definition:

META_RESULT __stdcall META_FAT_Find_Close(int *p_find_handle)

Description:

This function is to release the resource of find_handle.

Return Value:

Table 6-583 The return value of META_FAT_Find_Close

Return value	Description
META_SUCCESS	SUCCESS
META_INVALID_ARGUMENTS	Invalid arguments.

Parameter:

Table 6-584 The parameter of META_FAT_Find_Close

Parameter	IN/OUT	Description
p_find_handle	IN	The pointer to the handle of found list.

6.11.16 META_FAT_GetDiskInfo

Definition:

```
        META_RESULT  __stdcall META_FAT_GetDiskInfo(
                        const char DriveLetter,
                        FAT_DiskInfo_T  *p_DiskInfo,
                        short *p_token)

typedef enum {
        FAT12 = 0,
        FAT16,
        FAT32
}FAT_TYPE;

typedef struct {
        FAT_TYPE        Type;                   // FAT system type
        unsigned int    SectorsPerCluster;      // How many sectors per cluster
        unsigned int    TotalSize;              // Total size of this drive (in bytes
        unsigned int    FreeSpace;              // Current free space of this drive (in bytes)
}FAT_DiskInfo_T;
```

Description:

Query target FAT driver disk information.

Return Value:

Table 6-585 The return value of META_FAT_GetDiskInfo

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-586 The parameter of META_FAT_GetDiskInfo

Parameter	IN/OUT	Description
DriveLetter	IN	The disk drive letter. For example: 'C' or 'D' or 'E' …etc.

Parameter	IN/OUT	Description
p_DiskInfo	IN/OUT	Return disk information.
p_token	IN/OUT	The token number.

6.11.17 META_FAT_CheckEnoughSpace

Definition:

```
        META_RESULT  __stdcall META_FAT_CheckEnoughSpace(
                                  FAT_FILE_INFO_REQ_T *req)
```

```
typedef struct {
        char      m_cDriveLetter;                    // Target FAT disk drive letter such as: 'C'
        char     *m_pcfilepath;                      //  File path of the file we intend to write into target FAT
} FAT_FILE_INFO_REQ_T;
```

Description:

Query if target FAT disk has enough disk space for writing a new file.

Return Value:

Table 6-587 The return value of META_FAT_CheckEnoughSpace

Return value	Description
META_SUCCESS	Success. There is enough disk space for writing.
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-588 The parameter of META_FAT_CheckEnoughSpace

Parameter	IN/OUT	Description
req	IN	FAT file information request

6.11.18 META_FAT_GetDriveType

Definition:

META_RESULT __stdcall META_FAT_GetDriveType(unsigned int ms_timeout, const char DriveLetter, int *p_DriveType);

META_RESULT __stdcall META_FAT_GetDriveType_r(const int meta_handle, unsigned int ms_timeout, const char DriveLetter, int *p_DriveType);

/*

NOR_DRIVE = 1,

NAND_DRIVE=2,

CARD_DRIVE = 3,

EXTERNAL_DRIVE = 4

*/

Description:

Query the type of target's drive. Notice that this function is not supported in ULC project and will return error code (META_FAT_ACTION_NOT_SUPPORT).

Return Value:

Table 6-589 The return value of META_FAT_GetDriveType

Return value	Description
META_SUCCESS	Success. There is enough disk space for writing.
META_FAT_ACTION_NOT_SUPPORT	This function is not supported.
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-590 The parameter of META_FAT_GetDriveType

Parameter	IN/OUT	Description
ms_timeout	IN	The time we will wait for target's response
DriveLetter	IN	The drive we want to know
*p_DriveType	IN/OUT	Drive type (NOR, NAND, CARD, EXTERNAL)

6.11.19 META_FAT_Read_To_File_Ex

Definition:

META_RESULT __stdcall META_FAT_Read_To_File_Ex(const int fs_handle, const char *filepath, CALLBACK_META_FAT_PROGRESS cb_progress, void *cb_progress_arg, short *p_token, int *p_stopflag) ;

META_RESULT __stdcall META_FAT_Read_To_File_Ex_r(const int meta_handle, const int fs_handle, const char *filepath, CALLBACK_META_FAT_PROGRESS cb_progress, void *cb_progress_arg, short *p_token, int *p_stopflag);

Description:

Read file from target FAT file system into the file on the local disk with a stop flag to support user stop the operation at ease.

Callback:

typedef int (__stdcall *CALLBACK_META_FAT_PROGRESS)(unsigned char percent, int sent_bytes, int total_bytes, const short token, void *usr_arg);

This callback function will be invoked during reading progress, you can use this callback function to get the finish percentage, sent_bytes and total_bytes.

Return Value:

Table 6-591 The return value of META_FAT_Read_To_File_Ex

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.
META_BUFFER_LEN	Read length of data exceeds buffer length.

Parameter:

Table 6-592 The parameter of META_FAT_Read_To_File_Ex

Parameter	IN/OUT	Description
fs_handle	IN	File handle which is created by META_FAT_Open().
local_filepath	IN	Local filepath to store the content of read data.
cb_progress	IN	Function pointer of progress callback.
cb_progress_arg	IN	User argument that will be used into callback function.
token	IN/OUT	Token value for this operation.
p_stopflag	IN	Pointer of the stop flag to notify the operation should be stopped.

6.11.20 META_FAT_Write_By_File_Ex

Definition:

META_RESULT __stdcall META_FAT_Write_By_File_Ex(const int fs_handle, const char *filepath, CALLBACK_META_FAT_PROGRESS cb_progress, void *cb_progress_arg, short *p_token, int *p_stopfalg);

META_RESULT __stdcall META_FAT_Write_By_File_Ex_r(const int meta_handle, const int fs_handle, const char *filepath, CALLBACK_META_FAT_PROGRESS cb_progress, void *cb_progress_arg, short *p_token, int *p_stopflag);

Description:

Write the content of local file into the file on target FAT file system with a stop flag to stop the operation.

Callback:

typedef int (__stdcall *CALLBACK_META_FAT_PROGRESS)(unsigned char percent, int sent_bytes, int total_bytes, const short token, void *usr_arg);

This callback function will be invoked during reading progress; you can use this callback function to get the finish percentage, sent_bytes and total_bytes.

Return Value:

Table 6-593 The return value of META_FAT_Write_By_File_Ex

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.
META_FILE_BAD	The local file can't open for read, or file length is zero.

Parameter:

Table 6-594 The parameter of META_FAT_Write_By_File_Ex

Parameter	IN/OUT	Description
fs_handle	IN	File handle which is created by META_FAT_Open().

Parameter	IN/OUT	Description
local_filepath	IN	Local filepath to read the content of written data.
cb_progress	IN	Function pointer of progress callback.
cb_progress_arg	IN	User argument that will be used into callback function.
token	IN/OUT	Token value for this operation.
p_stopflag	IN	Pointer of the stop flag to notify the operation should be stopped.

6.11.21 META_FAT_RemoveDir

Definition:

META_RESULT __stdcall META_FAT_RemoveDir (const char *fat_dirpath);

META_RESULT __stdcall META_FAT_RemoveDir_r (const int meta_handle, const char *fat_dirpath);

Description:

Delete a remote directory on target FAT file system by the given absolute FAT file path.

Return Value:

Table 6-595 The return value of META_FAT_RemoveDir

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.
META_COMM_FAIL	Failure. This means the communication between PC and target are failed.
META_BUSY	FAT api is busy; please try again later.
META_TIMEOUT	Wait for target confirmation timeout.
META_INVALID_ARGUMENTS	Invalid arguments.
META_NO_MEMORY	Cannot allocate memory.
META_INTERNAL_DB_ERR	Internal database error.
META_INVALID_HANDLE	Invalid given meta handle.

Parameter:

Table 6-596 The parameter of META_FAT_RemoveDir

Parameter	IN/OUT	Description
fat_filepath	IN	The absolute FAT file path that you want to delete.
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

6.11.22 META_Check_ULC_support

Definition:

META_RESULT __stdcall META_RESULT __stdcall META_Check_ULC_support(unsigned int ms_timeout);

META_RESULT __stdcall META_RESULT __stdcall META_Check_ULC_support_r(const int meta_handle, unsigned int ms_timeout);

Description:

Check to see whether this is a ULC project or not.

Return Value:

Table 6-597 The return value of META_Check_ULC_support

Return value	Description
META_SUCCESS	SUCCESS
META_FAILED	The status field of target confirmation is error.

Parameter:

Table 6-598 The parameter of META_Check_ULC_support

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

6.12 Exported Functions for BlueTooth Operation

6.12.1 META_BTPowerOn

Definition:

META_BTPowerOn(unsigned int ms_timeout)

Description:

In previous version, the FT task in target will automatically let BlueTooth initialized, now users have to call META_BTPowerOn to make Bluetooth initialized, otherwise, BT module in target could not accept any command.

CallBack:

NA

Return Value:

Table 6-599 The return value of META_BTPowerOn

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-600 The parameter of META_BTPowerOn

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

6.12.2 META_BT_SendHCICommand

Definition:

META_BT_SendHCICommand(unsigned int ms_timeout, BT_HCI_COMMAND *req, META_BT_HCI_CNF cb, void *cb_arg, unsigned char Cmpltcode)

```
typedef struct {
        unsigned short    m_opcode;
```

```
            unsigned char       m_len;
            unsigned char       m_cmd[256];
} BT_HCI_COMMAND;


typedef struct {
            unsigned char   m_event;
            char                    m_status;
            unsigned short  m_handle;
            unsigned char       m_len;
            unsigned char       m_parms[256];
} BT_HCI_EVENT;
```

Description:

Send Bluetooth HCI command

CallBack:

typedef void (__stdcall *META_BT_HCI_CNF)(const BT_HCI_EVENT *cnf, const short token, void *usrData);

Return Value:

Table 6-601 The return value of META_BT_SendHCICommand

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-602 The parameter of META_BT_SendHCICommand

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).
req	IN	Bluetooth HCI command
cb_arg	IN	Interncal callback argument
Cmpltcode	IN	While Send HCI command, the last event you receive.
cb	IN	META_BT_HCI_CNF callback function

6.12.3 META_BT_CancelHCICommand

Definition:

META_BT_CancelHCICommand(unsigned int ms_timeout)

Description:

While Send Bluetooth HCI command, the command is on processing, you could submit META_BT_CancelHCICommand to cancel the command.

Return Value:

Table 6-603 The return value of META_BT_CancelHCICommand

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-604 The parameter of META_BT_CancelHCICommand

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).

6.12.4 META_BT_SendHCIData

Definition:

META_BT_SendHCIData(unsigned int ms_timeout, BT_HCI_BUFFER *snd, META_BT_HCI_TXDATA_CNF cb_tx, void *cb_arg)

```
typedef struct {
   unsigned short          m_con_hdl;
   unsigned short          m_len;
   unsigned char           m_buffer[BT_PACKET_LEN];
} BT_HCI_BUFFER;
typedef struct {
    unsigned short        m_len;
    unsigned char         m_data[BT_PACKET_LEN];
} BT_HCI_PACKET;
```

Description:

Send Bluetooth HCI Data

CallBack:

typedef void (__stdcall *META_BT_HCI_TXDATA_CNF)(const BT_HCI_PACKET *cnf, const short token, void *usrData);

Return Value:

Table 6-605 The return value of META_BT_SendHCIData

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-606 The parameter of META_BT_SendHCIData

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).
snd	IN	Bluetooth HCI Data
cb_arg	IN	Interncal callback argument
cb_tx	IN	META_BT_HCI_TXDATA_CNF callback function

6.12.5 META_BT_RegisterAutoCallback

Definition:

META_BT_RegisterAutoCallback(META_BT_AUTO_HCI_CNF cb_auto)

Description:

Register AUTO Callback function, this type of AUTO is reveice event which is triggered by peer devices

CallBack:

typedef void (__stdcall *META_BT_AUTO_HCI_CNF)(const BT_HCI_EVENT *cnf, const short token, void *usrData);

Return Value:

Table 6-607 The return value of META_BT_RegisterAutoCallback

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

6.12.6 META_BT_RemoveAutoCallback

Definition:

META_RESULT __stdcall META_BT_RemoveAutoCallback();

Description:

Remove AUTO Callback function, this type of AUTO is reveice event which is triggered by peer devices

Return Value:

Table 6-608 The return value of META_BT_RemoveAutoCallback

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

6.12.7 META_BT_ReceiveHCIData

Definition:

META_BT_ReceiveHCIData(META_BT_HCI_RXDATA_CNF cb_rx)

Description:

Register META_BT_HCI_RXDATA_CNF Callback function, while BT sender device send data to receiver, The receiver receives event which is processing by META_BT_HCI_RXDATA_CNF.

CallBack:

typedef void (__stdcall *META_BT_HCI_RXDATA_CNF)(const BT_HCI_BUFFER *cnf, const short token, void *usrData);

Return Value:

Table 6-609 The return value of META_BT_ReceiveHCIData

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

6.12.8 META_BT_RemoveReceiveHCIDataCallback

Definition:

META_RESULT __stdcall META_BT_RemoveReceiveHCIDataCallback();

Description:

Remove HCI Data Callback function.

Return Value:

Table 6-610 The return value of META_BT_RemoveReceiveHCIDataCallback

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

6.12.9 META_BT_TxPureTest

Definition:

META_BT_TxPureTest(unsigned int ms_timeout, BT_HCI_TX_PURE_TEST *snd, META_BT_HCI_TXTEST_CNF cb_tx, void *cb_arg)

```
typedef struct {
  unsigned short      m_con_hdl;
  unsigned short      m_len;
  unsigned short      m_total_pks;
} BT_HCI_TX_PURE_TEST;
typedef struct {
  unsigned int        m_used_time;
  unsigned short      m_len;
```

} BT_HCI_TX_PURE_TEST_STAT;

Description:

In order to do META throughput test for TX

CallBack:

typedef void (__stdcall *META_BT_HCI_TXTEST_CNF)(const BT_HCI_TX_PURE_TEST_STAT *cnf, const short token, void *usrData);

Return Value:

Table 6-611 The return value of META_BT_TxPureTest

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-612 The parameter of META_BT_TxPureTest

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).
Snd	IN	Specified the packet length and total packets to be sent directly by target FT task
Cb_tx	IN	The call back calculate the used_time and length, this will be calculated by used_time/length
cb_arg	IN	Internal callback argument

6.12.10 META_BT_RxTestStart

Definition:

META_BT_RxTestStart (unsigned int ms_timeout, META_BT_HCI_RXTEST_CNF cb_rx)

typedef struct {

  unsigned int   m_used_time;

  unsigned short   m_len;

} BT_HCI_RX_PURE_TEST_STAT;

Description:

In order to do META throughput test for RX.

CallBack:

typedef void (__stdcall *META_BT_HCI_RXTEST_CNF)(const BT_HCI_RX_PURE_TEST_STAT *cnf, const short token, void *usrData);

Return Value:

Table 6-613 The return value of META_BT_RxTestStart

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-614 The parameter of META_BT_RxTestStart

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).
Cb_rx	IN	The call back calculate the used_time and length, this will be calculated by used_time/length

6.12.11 META_BT_RxTestEnd

Definition:

META_BT_RxTestEnd(unsigned int ms_timeout)

Description:

End to calculate META throughput test for RX.

CallBack:

NA

Return Value:

Table 6-615 The return value of META_BT_RxTestEnd

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-616 The parameter of META_BT_RxTestEnd

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).

6.12.12 META_BT_TxPureTest_V2

Definition:

META_RESULT __stdcall META_BT_TxPureTest_V2(unsigned int ms_timeout, BT_HCI_TX_PURE_TEST *snd, META_BT_HCI_TXTEST_V2_CNF cb_txtest, void *cb_arg);

```
typedef struct {
    unsigned short        m_con_hdl;
    unsigned short        m_len;
    unsigned short        m_total_pks;
}                                                        BT_HCI_TX_PURE_TEST;

typedef struct {
    unsigned int          m_u4UsedTime;
    unsigned short        m_u2PktSentNum;

}                                                        BT_HCI_TX_PURE_TEST_STAT_V2;
```

Description:

A revised API In order to do META BT throughput test for TX. (so obsolete META_BT_TxPureTes)

CallBack:

typedef void (__stdcall *META_BT_HCI_TXTEST_V2_CNF)(const BT_HCI_TX_PURE_TEST_STAT_V2 *cnf, const short token, void *usrData);

Return Value:

Table 6-617 The return value of META_BT_TxPureTest_V2

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-618 The parameter of META_BT_TxPureTest_V2

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).
Snd	IN	Specified the packet length and total packets to be sent directly by target FT task
Cb_tx	IN	The call back calculate the used_time and packet number, this will be calculated by (packet number * packet size)/used_time.
cb_arg	IN	Internal callback argument

6.12.13 META_BT_RxTestStart_V2

Definition:

META_RESULT __stdcall META_BT_RxTestStart_V2(unsigned int ms_timeout, META_BT_HCI_RXTEST_CNF cb_rx);

```
typedef struct {
    unsigned int        m_used_time;
    unsigned short      m_len;
} BT_HCI_RX_PURE_TEST_STAT;
```

Description:

In order to do META throughput test for RX.

CallBack:

typedef void (__stdcall *META_BT_HCI_RXTEST_CNF)(const BT_HCI_RX_PURE_TEST_STAT *cnf, const short token, void *usrData);

Return Value:

Table 6-619 The return value of META_BT_RxTestStart_V2

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-620 The parameter of META_BT_RxTestStart_V2

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).
Cb_rx	IN	The call back calculate the used_time and length, this will be calculated by used_time/length

6.12.14 META_BT_EnableNvramOnlineUpdate

Definition:

META_RESULT __stdcall META_BT_EnableNvramOnlineUpdate(unsigned int ms_timeout);

Description:

Enable online update NVRAM data to BT stack, i.e., ask Target update BT stack with the latest NVRAM data via calling BT_PowerOn/Off when we update the data to NVRAM..

Return Value:

Table 6-621 The return value of META_BT_EnableNvramOnlineUpdate

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-622 The parameter of META_BT_EnableNvramOnlineUpdate

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).

6.12.15 META_BT_DisableNvramOnlineUpdate

Definition:

META_RESULT __stdcall META_BT_DisableNvramOnlineUpdate(unsigned int ms_timeout);

Description:

Disable online update NVRAM data to BT stack, i.e., ask Target not to update BT stack with the latest NVRAM data via calling BT_PowerOn/Off when we update the data to NVRAM. This will save several seconds if you don't want to apply new settings in BT stack right away.

Return Value:

Table 6-623 The return value of META_BT_DisableNvramOnlineUpdate

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-624 The parameter of META_BT_DisableNvramOnlineUpdate

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).

6.12.16 META_BT_EnablePcmClockSyncSignal

Definition:

META_RESULT __stdcall META_BT_EnablePcmClockSyncSignal(unsigned int ms_timeout);

Description:

Enable PCM clock sync. signal from AFE (Audio Front End) for BT calibration.

Return Value:

Table 6-625 The return value of META_BT_EnablePcmClockSyncSignal

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-626 The parameter of META_BT_EnablePcmClockSyncSignal

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).

6.12.17 META_BT_DisablePcmClockSyncSignal

Definition:

META_RESULT __stdcall META_BT_DisablePcmClockSyncSignal(unsigned int ms_timeout);
META_RESULT __stdcall META_BT_DisablePcmClockSyncSignal_r(const int meta_handle, unsigned int ms_timeout);

Description:

Disable PCM clock sync. signal from AFE (Audio Front End) for BT calibration.

Return Value:

Table 6-627 The return value of META_BT_DisablePcmClockSyncSignal

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-628 The parameter of META_BT_DisablePcmClockSyncSignal

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).

6.12.18 META_BT_POWERON_EX

Definition:

META_RESULT __stdcall META_BT_POWERON_EX(const unsigned int ms_timeout, const unsigned char u1WaitFlag);
META_RESULT __stdcall META_BT_POWERON_EX_r(const int meta_handle, const unsigned int ms_timeout, const unsigned char u1WaitFlag);

Description:

Command BT module to power on with wait flag. u1WaitFlag = 1: means the API will not return until the BT module really power on (Usually takes 2~3 secs).

Return Value:

Table 6-629 The return value of META_BT_POWERON_EX

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-630 The parameter of META_BT_POWERON_EX

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).
u1WaitFlag	IN	Setting the field to be 1 means the API will not return until the BT module really power on.

6.12.19 META_BT_POWEROFF_EX

Definition:

META_RESULT __stdcall META_BT_POWEROFF_EX(const unsigned int ms_timeout, const unsigned char u1WaitFlag);

META_RESULT __stdcall META_BT_POWEROFF_EX_r(const int meta_handle, const unsigned int ms_timeout, const unsigned char u1WaitFlag);

Description:

Command BT module to power off with wait flag. u1WaitFlag = 1: means the API will not return until the BT module really power off (Usually takes 2~3 secs).

Return Value:

Table 6-631 The return value of META_BT_POWEROFF_EX

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-632 The parameter of META_BT_POWEROFF_EX

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds).
u1WaitFlag	IN	Setting the field to be 1 means the API will not return until the BT module really power off.

6.12.20 META_QueryIfBTPowerOn

Definition:

META_QueryIfBTPowerOn(unsigned int ms_timeout);

Description:

Query if BT Power on

CallBack:

NA

Return Value:

Table 6-633 The return value of META_QueryIfBTPowerOn

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-634 The parameter of META_QueryIfBTPowerOn

Parameter	IN/OUT	Description
ms_timeout	IN	Function timeout value. (in milliseconds)

6.13 WiFi Operation

6.13.1 META_WiFi_QueryIfWiFiSupport

META_RESULT __stdcall META_WiFi_QueryIfWiFiSupport (unsigned int ms_timeout)

META_RESULT __stdcall META_WiFi_QueryIfWiFiSupport _r(const int meta_handle, unsigned int ms_timeout)

Description:

Query if target support WiFi.

Return Value:

Table 6-635 The return value of META_WiFi_QueryIfWiFiSupport

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-636 The parameter of META_WiFi_QueryIfWiFiSupport

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.13.2 META_WiFi_GetWiFiID

```
META_RESULT __stdcall META_WiFi_GetWiFiID(unsigned int ms_timeout, WiFiMod_ID *cnf)

META_RESULT __stdcall META_WiFi_GetWiFiID_r(const int meta_handle, unsigned int ms_timeout,
                                             WiFiMod_ID *cnf)

typedef struct {
        unsigned int        id;
} WiFiMod_ID;
```

Description:

Get WiFi module ID from target.

Return Value:

Table 6-637 The return value of META_WiFi_GetWiFiID

Return value	Description
META_SUCCESS	Success

Return value	Description
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-638 The parameter of META_WiFi_GetWiFiID

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	IN/OUT	WiFi module ID from target

6.13.3 META_WiFi_QueryMacAddress

META_RESULT __stdcall META_WiFi_QueryMacAddress (unsigned int ms_timeout, unsigned char* mac_addr)

META_RESULT __stdcall META_WiFi_QueryMacAddress_r(const int meta_handle, unsigned int ms_timeout, unsigned char* mac_addr)

Description:

Query target WiFi MAC address.

Return Value:

Table 6-639 The return value of META_WiFi_QueryMacAddress

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-640 The parameter of META_WiFi_QueryMacAddress

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
mac_addr	IN/OUT	WiFi MAC address

6.13.4 META_WiFi_SetSSID

META_RESULT __stdcall META_WiFi_SetSSID(unsigned int ms_timeout, char* p_SSID, bool bSetRegister)

META_RESULT __stdcall META_WiFi_SetSSID_r(const int meta_handle, unsigned int ms_timeout,
char* p_SSID, bool bSetRegister)

Description:

Set SSID to target WiFi module.

Return Value:

Table 6-641 The return value of META_WiFi_SetSSID

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-642 The parameter of META_WiFi_SetSSID

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
p_SSID	IN	SSID string
bSetRegister	IN	Set register flag

6.13.5 META_WiFi_SetDriverTestMode

META_RESULT __stdcall META_WiFi_SetDriverTestMode(unsigned int ms_timeout)

META_RESULT __stdcall META_WiFi_SetDriverTestMode_r(const int meta_handle, unsigned int ms_timeout)

Description:

Commands target to set WiFi driver to test mode for both RX and TX test.

Return Value:

Table 6-643 The return value of META_WiFi_SetDriverTestMode

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-644 The parameter of META_WiFi_SetDriverTestMode

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.13.6 META_WiFi_SetDriverNormalMode

META_RESULT __stdcall META_WiFi_ SetDriverNormalMode (unsigned int ms_timeout)

META_RESULT __stdcall META_WiFi_ SetDriverNormalMode _r(const int meta_handle, unsigned int ms_timeout)

Description:

Commands target to set WiFi driver to normal mode.

Return Value:

Table 6-645 The return value of META_WiFi_SetDriverNormalMode

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-646 The parameter of META_WiFi_SetDriverNormalMode

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.13.7 META_WiFi_Stop

META_RESULT __stdcall META_WiFi_Stop (unsigned int ms_timeout)

META_RESULT __stdcall META_WiFi_Stop_r (const int meta_handle, unsigned int ms_timeout)

Description:

Commands WiFi module to stop all WiFi testing, these testing include continuous packet TX, continuous packet RX, TX output power, TX carrier suppression and local frequency measure testing.

Return Value:

Table 6-647 The return value of META_WiFi_Stop

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-648 The parameter of META_WiFi_Stop

Parameter	IN/OUT	Description
Meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.13.8 META_WiFi_OutputPower

Definition:

META_RESULT __stdcall META_WiFi_OutputPower (unsigned int ms_timeout, eWiFiTxRate tx_rate)

META_RESULT __stdcall META_WiFi_OutputPower_r (const int meta_handle, unsigned int ms_timeout, eWiFiTxRate tx_rate)

```
typedef enum
{
  WiFiTxRate1M=0,                        // 1M
  WiFiTxRate2M,                         // 2M
  WiFiTxRate5_5M,                        // 5.5M
  WiFiTxRate11M,                         // 11M
  WiFiTxRate6M,                          // 6M
  WiFiTxRate9M,                          // 9M
  WiFiTxRate12M,                         // 12M
  WiFiTxRate18M,                         // 18M
  WiFiTxRate24M,                         // 24M
  WiFiTxRate36M,                         // 36M
  WiFiTxRate48M,                         // 48M
  WiFiTxRate54M,                         // 54M
  WiFiTxRateCount                        // count of WiFi TX rate
} eWiFiTxRate;
```

Description:

Commands WiFi module with output power for spectral mask and power measurement test.

Return Value:

Table 6-649 The return value of META_WiFi_OutputPower

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-650 The parameter of META_WiFi_OutputPower

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
tx_rate	IN	WiFi module TX rate

6.13.9 META_WiFi_LocalFrequencyMeasure

Definition:

META_RESULT __stdcall META_WiFi_LocalFrequencyMeasure(unsigned int ms_timeout, const WiFi_TestTx_S *req);

META_RESULT __stdcall META_WiFi_LocalFrequencyMeasure_r(const int meta_handle, unsigned int ms_timeout, const WiFi_TestTx_S *req);

```
typedef struct {
    unsigned int            ch_freq;/* Frq, units are kHz */
    WiFi_TestRate_E         tx_rate;
    unsigned char           txAnt; /* 0 for Antenna 0; 1 for Antenna 1 */
    unsigned short          tx_gain_dac;
} WiFi_TestTx_S;
```

Description:

Commands WiFi module to do local frequency test.

Return Value:

Table 6-651 The return value of META_WiFi_LocalFrequencyMeasure

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-652 The parameter of META_WiFi_LocalFrequencyMeasure

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	WiFi_TestTx_S

6.13.10 META_WiFi_CarrierSuppressionMeasure

META_RESULT __stdcall META_WiFi_CarrierSuppressionMeasure(unsigned int ms_timeout, const WiFi_TestTx_S *req);

META_RESULT __stdcall META_WiFi_CarrierSuppressionMeasure_r(const int meta_handle, unsigned int ms_timeout, const WiFi_TestTx_S *req);

```
typedef struct {
    unsigned int            ch_freq;/* Frq, units are kHz */
    WiFi_TestRate_E         tx_rate;
    unsigned char           txAnt; /* 0 for Antenna 0; 1 for Antenna 1 */
    unsigned short          tx_gain_dac;
} WiFi_TestTx_S;
```

Description:

Commands WiFi module to do carrier suppression measure test.

Return Value:

Table 6-653 The return value of META_WiFi_CarrierSuppressionMeasure

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-654 The parameter of META_WiFi_CarrierSuppressionMeasure

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	WiFi_TestTx_S

6.13.11 META_WiFi_ContPktTx

Definition:

META_RESULT __stdcall META_WiFi_ContPktTx(unsigned int ms_timeout, const WiFi_TestPktTx_S *req);

META_RESULT __stdcall META_WiFi_ContPktTx_r(const int meta_handle, unsigned int ms_timeout, const WiFi_TestPktTx_S *req);

```
typedef struct {
        unsigned int            ch_freq;            /* Frq, units are kHz */
        WiFi_TestRate_E         tx_rate;
        unsigned short          tx_gain_dac;
        unsigned int            pktCount;
        unsigned int            pktInterval;        /* interval between each Tx Packet */
        unsigned int            pktLength;          /* 24~1500 */
        WiFi_TestPktTxPattern_E pattern;            /* content of the Tx Packet */
        unsigned char           txAnt;              /* 0 for Antenna 0; 1 for Antenna 1 */
        unsigned char           is_short_preamble;  /* 0 for long preamble and 1 for short preamble */
        unsigned char           mac_header[ 24 ];   /* Frame Ctrl, Duration = 2bytes + 2bytes */
                                                    /* Address 1 = 6 bytes */
                                                    /* Address 2 = 6 bytes */
                                                    /* Address 3 = 6 bytes */
                                                    /* Sequence Ctrl = 2 bytes */
} WiFi_TestPktTx_S;
```

Description:

Commands WiFi module to continuous TX mode.

Return Value:

Table 6-655 The return value of META_WiFi_ContPktTx

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-656 The parameter of META_WiFi_ContPktTx

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	WiFi_TestPktTx_S, definition part has detail explanation

6.13.12 META_WiFi_QueryTxStatus

Definition:

META_RESULT __stdcall META_WiFi_QueryTxStatus(unsigned int ms_timeout, WiFi_TxStatus_S *cnf);

META_RESULT __stdcall META_WiFi_QueryTxStatus_r(const int meta_handle, unsigned int ms_timeout, WiFi_TxStatus_S *cnf);

```
typedef struct {
        unsigned int            pkt_sent_count;  /* total num sent */
        unsigned int            pkt_sent_acked;  /* acked num */
} WiFi_TxStatus_S;
```

Description:

Query how many packets sent by WiFi module.

Return Value:

Table 6-657 The return value of META_WiFi_QueryTxStatus

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-658 The parameter of META_WiFi_QueryTxStatus

Parameter	IN/OUT	Description
Meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	IN/OUT	Pointer to WiFi_TxStatus_S by WiFi module

6.13.13 META_WiFi_SetPowerManagementMode

Definition:

META_RESULT __stdcall META_WiFi_SetPowerManagementMode(unsigned int ms_timeout, const WiFi_PowerManagementMode_E mode)

META_RESULT __stdcall META_WiFi_SetPowerManagementMode_r(const int meta_handle, unsigned int ms_timeout, const WiFi_PowerManagementMode_E mode);

```
typedef enum {
        WIFI_POWER_MODE_NORMAL,
        WIFI_POWER_MODE_IDLE,
        WIFI_POWER_MODE_SLEEP
} WiFi_PowerManagementMode_E;
```

Description:

Commands WiFi module to switch power management operation.

Return Value:

Table 6-659 The return value of META_WiFi_SetPowerManagementMode

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-660 The parameter of META_WiFi_SetPowerManagementMode

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
mode	IN	Power management type

6.13.14 META_WiFi_ContPktRx

Definition:

META_RESULT __stdcall META_WiFi_ContPktRx(unsigned int ms_timeout, const WiFi_TestPktRx_S *req);

META_RESULT __stdcall META_WiFi_ContPktRx_r(const int meta_handle, unsigned int ms_timeout, const WiFi_TestPktRx_S *req);

```
typedef struct {
    unsigned int                          ch_freq;  /* Frq, units are kHz */
    WiFi_TestPktRxMode_E           mode;
    WiFi_RxAntSel_E                       rxAnt;
} WiFi_TestPktRx_S;
```

Description:

Command WiFi module to set continuous Packet RX mode.

Return Value:

Table 6-661 The return value of META_WiFi_ContPktRx

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-662 The parameter of META_WiFi_ContPktRx

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Type WiFi_TestPktRx_S, mode of continuous RX, Antenna(0: antenna A, 1: antenna B)

6.13.15 META_WiFi_QueryRxStatus

Definition:

META_RESULT __stdcall META_WiFi_QueryRxStatus(unsigned int ms_timeout, WiFi_RxStatus_S *cnf)

META_RESULT __stdcall META_WiFi_QueryRxStatus_r(const int meta_handle, unsigned int ms_timeout, WiFi_RxStatus_S *cnf);

```
typedef struct {
        unsigned int        int_rx_ok_num;    /* number of packets that Rx ok from interrupt */
        unsigned int        int_crc_err_num;  /* number of packets that CRC error from interrupt */
        unsigned int        pau_rx_pkt_count; /* number of packets that Rx ok from PAU */
        unsigned int        pau_crc_err_count; /* number of packets that CRC error from PAU */
        unsigned int        pau_cca_count;    /* CCA rising edge count */
        unsigned int        pau_rx_fifo_full_count; /* number of lost packets due to FiFo full */
        unsigned int        int_long_preamble_num;
        unsigned int        int_short_preamble_num;
        unsigned int        int_rate_ok_num[ WIFI_TEST_RATE_COUNT ];
        unsigned int        int_rate_crc_err_num[ WIFI_TEST_RATE_COUNT ];
        int                         int_rssi_max;
        int                         int_rssi_min;
        int                         int_rssi_mean;
        int                         int_rssi_variance;
} WiFi_RxStatus_S;
```

Description:

Command WiFi module to query RX status.

Return Value:

Table 6-663 The return value of META_WiFi_QueryRxStatus

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-664 The parameter of META_WiFi_QueryRxStatus

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	IN/OUT	WiFi_RxStatus_S, see definition part.

6.13.16 META_WiFi_SetChannel

Definition:

```
METAA_RESULT __stdcall META_WiFi_SetChannel ( int ms_timeout, int channel_freq)
METAA_RESULT __stdcall META_WiFi_SetChannel_r (const int meta_handle, int ms_timeout,
                                                int channel_freq)
```

Description:

Command WiFi module to set RF channel by frequency.

Return Value:

Table 6-665 The return value of META_WiFi_SetChannel

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-666 The parameter of META_WiFi_SetChannel

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle()
ms_timeout	IN	Timeout value, unit = minisecond
channel_freq	IN	Channel frequency in kHz (2412~2484)

6.13.17 META_WiFi_QueryChannelList

Definition:

```
METAA_RESULT __stdcall META_WiFi_QueryChannelList (int ms_timeout,
                                                    unsigned int *p_channel_num,
                                                    unsigned char *p_channel_id)
METAA_RESULT __stdcall META_WiFi_QueryChannelList _r (const int meta_handle,
                                                      int ms_timeout,
                                                      unsigned *p_channel_num,
                                                      unsigned char *p_channel_id)
```

Description:

Command WiFi module to query available RF channel list that can be used by ID.

Return Value:

Table 6-667 The return value of META_WiFi_QueryChannelList

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-668 The parameter of META_WiFi_QueryChannelList

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle()
ms_timeout	IN	Timeout value, unit = minisecond
p_channel_num	IN/OUT	Pointer to channel number
p_channel_freq	IN/OUT	Pointer to channel in use by frequency in kHz (2412~2484)

6.13.18 META_WiFi_SetRegDomain

Definition:

```
METAA_RESULT __stdcall META_WiFi_SetRegDomain ( int ms_timeout, unsigned char *p_reg_domain)
METAA_RESULT __stdcall META_WiFi_SetRegDomain _r (const int meta_handle, int ms_timeout,
                                                    unsigned char *p_reg_domain)
```

Description:

Commands WiFi module to set TX filter to meet standard of North America or Japan.

Return Value:

Table 6-669 The return value of META_WiFi_SetRegDomain

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-670 The parameter of META_WiFi_SetRegDomain

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle()
ms_timeout	IN	Timeout value, unit = minisecond
p_reg_domain	IN	Register domain (“US”: North America, “JP”: Japan)

6.13.19 META_WiFi_ReadMacReg

Definition:

METAAPP_RESULT __stdcall META_WiFi_ReadMacReg (int ms_timeout, int index, unsigned int *p_value)

METAAPP_RESULT __stdcall META_WiFi_ReadMacReg_r (const int meta_handle, int ms_timeout, unsigned int index, unsigned int *p_value)

Description:

Commands WiFi module to read data from MAC register.

Return Value:

Table 6-671 The return value of META_WiFi_ReadMacReg

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-672 The parameter of META_WiFi_ReadMacReg

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle()
ms_timeout	IN	Timeout value, unit = minisecond
index	IN	Index of MAC register
p_value	IN/OUT	Pointer of value read from MAC register

6.13.20 META_WiFi_WriteMacReg

Definition:

```
METAAPP_RESULT __stdcall META_WiFi_WriteMacReg( int ms_timeout,
                                                unsigned char index,
                                                unsigned char value)
METAAPP_RESULT __stdcall META_WiFi_WriteMacReg_r (const int meta_handle,
                                                  int ms_timeout,
                                                  unsigned char index,
                                                  unsigned char value)
```

Description:

Commands WiFi module to write data to MAC register.

Return Value:

Table 6-673 The return value of META_WiFi_WriteMacReg

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-674The parameter of META_WiFi_WriteMacReg

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle()
ms_timeout	IN	Timeout value, unit = minisecond
index	IN	Index of baseband register
value	IN	value write to baseband register (size: 1byte)

6.13.21 META_WiFi_ReadBBReg

Definition:

METAAPP_RESULT __stdcall META_WiFi_ReadBBReg (int ms_timeout, int index, unsigned char *p_value)

METAAPP_RESULT __stdcall META_WiFi_ReadBBReg_r (const int meta_handle, int ms_timeout,
unsigned char index, unsigned char *p_value)

Description:

Commands WiFi module to read 1 byte data from baseband register.

Return Value:

Table 6-675 The return value of META_WiFi_ReadBBReg

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-676 The parameter of META_WiFi_ReadBBReg

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle()
ms_timeout	IN	Timeout value, unit = minisecond
index	IN	Index of baseband register
p_value	IN/OUT	Pointer of value read from baseband register (size: 1byte)

6.13.22 META_WiFi_WriteBBReg

Definition:

METAAPP_RESULT __stdcall META_WiFi_WriteBBReg (int ms_timeout,
unsigned char index,
unsigned char value)

METAAPP_RESULT __stdcall META_WiFi_WriteBBReg_r (const int meta_handle,
int ms_timeout,

unsigned char index,

unsigned char value)

Description:

Commands WiFi module to write 1 byte data to baseband register.

Return Value:

Table 6-677 The return value of META_WiFi_WriteBBReg

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-678 The parameter of META_WiFi_WriteBBReg

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle()
ms_timeout	IN	Timeout value, unit = minisecond
index	IN	Index of baseband register
value	IN	value write to baseband register (size: 1byte)

6.13.23 META_WiFi_ContPktTx_Ex

Definition:

```
META_WiFi_ContPktTx_Ex(unsigned int  ms_timeout, const WiFi_TestPktTx_Ex_S *req);


typedef struct {
        unsigned int            ch_freq;            /* Frq, units are kHz */
        WiFi_TestRate_E         tx_rate;
        unsigned short          tx_gain_dac;
        unsigned int            pktCount;
        unsigned int            pktInterval;        /* interval between each Tx Packet */
        unsigned int            pktLength;                  /* 24~1500 */
        WiFi_TestPktTxPattern_E   pattern;                /* content of the Tx Packet */
        unsigned char                 txAnt;                          /*  0  for  Antenna  0;  1  for
Antenna 1 */
        unsigned int                  txFlags;
        unsigned int                  targetAlc;
        unsigned char                 is_short_preamble;/* 0 for long preamble and 1 for short preamble
*/
        unsigned char                 mac_header[ 24 ];        /* Frame Ctrl, Duration = 2bytes + 2bytes
*/
                                                               /* Address 1 = 6 bytes */
                                                               /* Address 2 = 6 bytes */
                                                               /* Address 3 = 6 bytes */
                                                               /* Sequence Ctrl = 2 bytes */
} WiFi_TestPktTx_Ex_S;
```

Description:

For support Alc, ContPktTx has new structure, to add additional two fields: txAnt,txFlags.

Return Value:

Table 6-679 The return value of META_WiFi_ContPktTx_Ex

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-680 The parameter of META_WiFi_ContPktTx_Ex

Parameter	IN/OUT	Description
ms_timeout	IN	Testing command.
req	IN	WiFi_TestPktTx_Ex to support Alc new structure additional two fields: txAnt,txFlags.

6.13.24 META_WiFi_SetTxALC2400M

Definition:

META_WiFi_SetTxALC2400M(unsigned int ms_timeout, const WiFi_TxALC_2400M_S *txalc);

```
typedef struct
{
    unsigned char alcSlop1Divider;
    unsigned char alcSlop1Dividend;
    unsigned char alcSlop2Divider;
    unsigned char alcSlop2Dividend;
} WiFi_TxALC_2400M_S;
```

Description:

For support Tx Alc slope, META_WiFi_SetTxALC2400M tun time setting.

Return Value:

Table 6-681 The return value of META_WiFi_SetTxALC2400M

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-682 The parameter of META_WiFi_SetTxALC2400M

Parameter	IN/OUT	Description
ms_timeout	IN	Testing command.
txalc	IN	WiFi_TxALC_2400M_S

6.13.25 META_WiFi_QueryTxStatus_Ex

Definition:

```
META_WiFi_QueryTxStatus_Ex(unsigned int  ms_timeout, WiFi_TxStatus_Ex_S *cnf);


typedef struct {
            unsigned int            pkt_sent_count;  /* total num sent */
            unsigned int            pkt_sent_acked;  /* acked num */
            unsigned short          avgAlc;
            unsigned char           cckGainControl;
   unsigned char             ofdmGainControl;
} WiFi_TxStatus_Ex_S;
```

Description:

For support Alc, QueryTxStatus has new structure, to add additional two fields: avgAlc, cckGainControl.

Return Value:

Table 6-683 The return value of META_WiFi_QueryTxStatus_Ex

Return value	Description
META_SUCCESS	Success
META_FAILED	Memory is not enough.
META_COMM_FAIL	Communication between PC and target are failed.

Parameter:

Table 6-684 The parameter of META_WiFi_QueryTxStatus_Ex

Parameter	IN/OUT	Description
ms_timeout	IN	Testing command.
cnf	IN	WiFi_TxStatus_Ex_S to support Alc new structure, additional two fields: avgAlc, cckGainControl.

6.13.26 META_NVRAM_WiFi_Compose_MacAddress

Definition:

```
META_RESULT __stdcall META_NVRAM_WiFi_Compose_MacAddress (
                        const wifi_permanent_mac_addresss_T * mac_addr,
```

char *buf, const int buf_len)

```
typedef struct
{
    kal_uint8       mac_addr[6];
} wifi_permanent_mac_addresss_T;
```

Description:

Compose WiFi MAC address. Usually, once the WiFi MAC address data is acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-685 The return value of META_NVRAM_WiFi_Compose_MacAddress

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-686 The parameter of META_NVRAM_WiFi_Compose_MacAddress

Parameter	IN/OUT	Description
mac_addr	IN	MAC address
buf	IN	Buffer
buf_len	IN	Size of buf

6.13.27 META_NVRAM_WiFi_Decompose_MacAddress

Definition:

```
    META_RESULT __stdcall META_NVRAM_WiFi_Decompose_MacAddress (
                                wifi_permanent_mac_addresss_T * mac_addr,
                                const char *buf, const int buf_len)
```

```
typedef struct
{
    kal_uint8        mac_addr[6];
} wifi_permanent_mac_addresss_T;
```

Description:

Decompose WiFi MAC address. Usually, once the buffer of WiFi MAC address data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure wifi_permanent_mac_addresss_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-687 The return value of META_NVRAM_WiFi_Decompose_MacAddress

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-688 The parameter of META_NVRAM_WiFi_Decompose_MacAddress

Parameter	IN/OUT	Description
p_mac_addr	IN/OUT	Pointer of MAC address
buf	IN	Buffer
buf_len	IN	Size of buf

6.13.28 META_NVRAM_WiFi_TxPower2400M_Len

Definition:

```
META_RESULT  __stdcall META_NVRAM_WiFi_TxPower2400M_Len(int *len)
```

Description:

This function returns the size of TxPower2400M.

Return Value:

Table 6-689 The return value of META_NVRAM_WiFi_TxPower2400M_Len

Return value	Description
META_SUCCESS	Success

Return value	Description
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-690 The parameter of META_NVRAM_WiFi_TxPower2400M_Len

Parameter	IN/OUT	Description
Len	OUT	Size of TxPower2400M

6.13.29 META_NVRAM_WiFi_Compose_TxPower2400M

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_Compose_TxPower2400M(const WiFi_TxPower_2400M_S *txpwr, char *buf, const int buf_len)

```
typedef struct {
        unsigned char     CCKTxPWR[14];
        unsigned char     OFDMTxPWR[14];
} WiFi_TxPower_2400M_S;
```

Description:

Compose WiFi TX power. Usually, once the calibrated WiFi TX power data is acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-691 The return value of META_NVRAM_WiFi_Compose_TxPower2400M

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-692 The parameter of META_NVRAM_WiFi_Compose_TxPower2400M

Parameter	IN/OUT	Description
tx_power	IN	WiFi_TxPower_2400M_S

Parameter	IN/OUT	Description
Buf	IN	Buffer
buf_len	IN	Size of buf

6.13.30 META_NVRAM_WiFi_Decompose_TxPower2400M

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_Decompose_TxPower2400M(WiFi_TxPower_2400M_S *txpwr, const char *buf, const int buf_len)

```
typedef struct {
        unsigned char     CCKTxPWR[14];
        unsigned char     OFDMTxPWR[14];
} WiFi_TxPower_2400M_S;
```

Description:

Decompose WiFi TX power. Usually, once the buffer of WiFi TX power data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure wifi_tx_power_table_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-693 The return value of META_NVRAM_WiFi_Decompose_TxPower2400M

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-694 The parameter of META_NVRAM_WiFi_Decompose_TxPower2400M

Parameter	IN/OUT	Description
p_tx_power	IN/OUT	Pointer to WiFi TX power TxPower2400M.
buf	IN	Buffer

Parameter	IN/OUT	Description
buf_len	IN	Size of buf

6.13.31 META_NVRAM_WiFi_TxPower5000M_Len

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_TxPower5000M_Len(int *len)

Description:

This function returns the size of TxPower5000M.

Return Value:

Table 6-695 The return value of META_NVRAM_WiFi_TxPower5000M_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can’t find structure info from InternalDB.

Parameter:

Table 6-696 The parameter of META_NVRAM_WiFi_TxPower5000M_Len

Parameter	IN/OUT	Description
Len	OUT	Size of TxPower5000M

6.13.32 META_NVRAM_WiFi_Compose_TxPower5000M

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_Compose_TxPower5000M(const WiFi_TxPower_5000M_S *txpwr, char *buf, const int buf_len)

```
typedef struct {
        unsigned char     TxPWR[34];
} WiFi_TxPower_5000M_S;
```

Description:

Compose WiFi TX power. Usually, once the calibrated WiFi TX power data is acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-697 The return value of META_NVRAM_WiFi_Compose_TxPower5000M

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-698 The parameter of META_NVRAM_WiFi_Compose_TxPower5000M

Parameter	IN/OUT	Description
tx_power	IN	WiFi_TxPower_5000M_S
Buf	IN	Buffer
buf_len	IN	Size of buf

6.13.33 META_NVRAM_WiFi_Decompose_TxPower5000M

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_Decompose_TxPower5000M(WiFi_TxPower_5000M_S *txpwr, const char *buf, const int buf_len)

```
typedef struct {
        unsigned char     TxPWR[34];
} WiFi_TxPower_5000M_S;
```

Description:

Decompose WiFi TX power. Usually, once the buffer of WiFi TX power data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure wifi_tx_power_table_T, and doesn't take care the byte alignment problem.

Return Value:

Table 6-699 The return value of META_NVRAM_WiFi_Decompose_TxPower5000M

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-700 The parameter of META_NVRAM_WiFi_Decompose_TxPower5000M

Parameter	IN/OUT	Description
p_tx_power	IN/OUT	Pointer to WiFi TX power WiFi_TxPower_5000M_S.
buf	IN	Buffer
buf_len	IN	Size of buf

6.13.34 META_NVRAM_WiFi_Compose_DacDcOffset

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_Compose_DacDcOffset(const WiFi_DAC_DC_Offset_S *dac, char *buf, const int buf_len)

```
typedef struct {
        unsigned char    i_ch_offset;
        unsigned char    q_ch_offset;
} WiFi_DAC_DC_Offset_S;
```

Description:

Compose DacDcOffset. The i_ch_offset and q_ch_offset will be composed.

Return Value:

Table 6-701 The return value of META_NVRAM_WiFi_Compose_DacDcOffset

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-702 The parameter of META_NVRAM_WiFi_Compose_DacDcOffset

Parameter	IN/OUT	Description
dac	IN/OUT	WiFi_DAC_DC_Offset_S
buf	IN	Buffer
buf_len	IN	Size of buf

6.13.35 META_NVRAM_WiFi_Decompose_DacDcOffset

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_Decompose_DacDcOffset(WiFi_DAC_DC_Offset_S *dac, const char *buf, const int buf_len)

```
typedef struct {
        unsigned char     i_ch_offset;
        unsigned char     q_ch_offset;
} WiFi_DAC_DC_Offset_S;
```

Description:

Decompose DacDcOffset. The i_ch_offset and q_ch_offset will be decomposed.

Return Value:

Table 6-703 The return value of META_NVRAM_WiFi_Decompose_DacDcOffset

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-704 The parameter of META_NVRAM_WiFi_Decompose_DacDcOffset

Parameter	IN/OUT	Description
dac	IN/OUT	WiFi_DAC_DC_Offset_S
buf	IN	Buffer
buf_len	IN	Size of buf

6.13.36 META_NVRAM_WiFi_Compose_ALC_2400M

Definition:

```
META_NVRAM_WiFi_Compose_ALC_2400M(const WiFi_ALC_2400M_S  *alc, char *buf, const int buf_len)

typedef struct {

        unsigned char txAlcCCK[14];

        unsigned char txOutputPowerDBCCK[14];

        unsigned char txAlcOFDM [8][14];

        unsigned char txOutputPowerDBOFDM[8][14];

} WiFi_ALC_2400M_S;
```

Description:

Compose WiFi ALC. Usually, once the calibrated WiFi ALC data is acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-705 The return value of META_NVRAM_WiFi_Compose_ALC_2400M

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-706 The parameter of META_NVRAM_WiFi_Compose_ALC_2400M

Parameter	IN/OUT	Description
alc	IN	WiFi_ALC_2400M_S
Buf	IN	Buffer
buf_len	IN	Size of buf

6.13.37 META_NVRAM_WiFi_Decompose_ALC_2400M

Definition:

META_NVRAM_WiFi_Decompose_ALC_2400M(WiFi_ALC_2400M_S *alc, const char *buf, const int buf_len);

```
typedef struct {
        unsigned char txAlcCCK[14];
        unsigned char txOutputPowerDBCCK[14];
        unsigned char txAlcOFDM [8][14];
        unsigned char txOutputPowerDBOFDM[8][14];
} WiFi_ALC_2400M_S;
```

Description:

Decompose WiFi ALC. Usually, once the buffer of WiFi ALC data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure struct_nvram_wifi_alc_2400m, and doesn't take care the byte alignment problem.

Return Value:

Table 6-707 The return value of META_NVRAM_WiFi_Decompose_ALC_2400M

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-708 The parameter of META_NVRAM_WiFi_Decompose_ALC_2400M

Parameter	IN/OUT	Description
alc	IN/OUT	Pointer to WiFi WiFi_ALC_2400M_S
buf	IN	Buffer
buf_len	IN	Size of buf

6.13.38 META_NVRAM_WiFi_ALC_2400M_Len

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_ALC_2400M_Len(int *len);

Description:

This function returns the size of WiFi_ALC_2400M table.

Return Value:

Table 6-709 The return value of META_NVRAM_WiFi_ALC_2400M_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-710 The parameter of META_NVRAM_WiFi_ALC_2400M_Len

Parameter	IN/OUT	Description
Len	OUT	Size of WiFi_ALC_2400M table

6.13.39 META_NVRAM_WiFi_Compose_ TxALC2400M

Definition:

META_NVRAM_WiFi_Compose_ TxALC2400M(const WiFi_TxALC_2400M_S *alc, char *buf, const int buf_len)

```
typedef struct
{
    unsigned char alcSlop1Divider;
    unsigned char alcSlop1Dividend;
    unsigned char alcSlop2Divider;
    unsigned char alcSlop2Dividend;
} WiFi_TxALC_2400M_S;
```

Description:

Compose WiFi ALC Slope. Usually, once the calibrated WiFi ALC data is acquired, this function is called before updating the corresponding data of NVRAM record, because this function take the responsibility of byte alignment issues while convert the structure data to raw data buffer, which need to be updated to NVRAM.

Return Value:

Table 6-711 The return value of META_NVRAM_WiFi_Compose_ TxALC2400M

Return value	Description
META_SUCCESS	Success

Return value	Description
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-712 The parameter of META_NVRAM_WiFi_Compose_ TxALC2400M

Parameter	IN/OUT	Description
alc	IN	WiFi_TxALC_2400M_S
Buf	IN	Buffer
buf_len	IN	Size of buf

6.13.40 META_NVRAM_WiFi_Decompose_ TxALC2400M

Definition:

META_NVRAM_WiFi_Decompose_ TxALC2400M (WiFi_TxALC_2400M_S *alc, const char *buf, const int buf_len);

```
typedef struct
{
    unsigned char alcSlop1Divider;
    unsigned char alcSlop1Dividend;
    unsigned char alcSlop2Divider;
    unsigned char alcSlop2Dividend;
} WiFi_TxALC_2400M_S;
```

Description:

Decompose WiFi ALC. Usually, once the buffer of WiFi ALC data are acquired from target (NVRAM) via META-DLL, this function should be called and it help programmer to mapping these raw data to fill into the proper field of the structure struct_nvram_wifi_alc_2400m, and doesn't take care the byte alignment problem.

Return Value:

Table 6-713 The return value of META_NVRAM_WiFi_Decompose_ TxALC2400M

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-714 The parameter of META_NVRAM_WiFi_Decompose_ TxALC2400M

Parameter	IN/OUT	Description
alc	IN/OUT	Pointer to WiFi_TxALC_2400M_S
buf	IN	Buffer
buf_len	IN	Size of buf

6.13.41 META_NVRAM_WiFi_TxALC2400M_Len

Definition:

META_RESULT __stdcall META_NVRAM_WiFi_ TxALC2400M_Len(int *len);

Description:

This function returns the size of WiFi_TXALC_2400M table.

Return Value:

Table 6-715 The return value of META_NVRAM_WiFi_TxALC2400M_Len

Return value	Description
META_SUCCESS	Success
META_INTERNAL_DB_ERR	Can't find structure info from InternalDB.

Parameter:

Table 6-716 The parameter of META_NVRAM_WiFi_TxALC2400M_Len

Parameter	IN/OUT	Description
Len	OUT	Size of WiFi_TXALC_2400M table

6.14 FM Radio Operation

6.14.1 META_FM_GetChipId

Definition:

META_RESULT __stdcall META_FM_GetChipId(unsigned int ms_timeout, FM_CHIP_ID_CNF_T *cnf);

```
META_RESULT __stdcall META_FM_GetChipId_r(const int meta_handle, unsigned int ms_timeout, FM_CHIP_ID_CNF_T *cnf);

#define FM_CHIP_ID_MT6189AN        0
#define FM_CHIP_ID_MT6189BN_CN     1
#define FM_CHIP_ID_MT6188A         3
#define FM_CHIP_ID_MT6188C         4
#define FM_CHIP_ID_MT6188D         5

typedef struct{
        unsigned char m_ucChipId;
}FM_CHIP_ID_CNF_T;
```

Description:

Query the FM chip ID.

Return Value:

Table 6-717 The return value of META_FM_GetChipId

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-718 The parameter of META_FM_GetChipId

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	IN/OUT	The FM chip ID

6.14.2 META_FM_PowerOn

Definition:

META_RESULT __stdcall META_FM_PowerOn(unsigned int ms_timeout);

META_RESULT __stdcall META_FM_PowerOn_r(const int meta_handle, unsigned int ms_timeout);

Description:

Turn on the FM Radio module.

Return Value:

Table 6-719 The return value of META_FM_PowerOn

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-720 The parameter of META_FM_PowerOn

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.14.3 META_FM_PowerOff

Definition:

META_RESULT __stdcall META_FM_PowerOff(unsigned int ms_timeout);

META_RESULT __stdcall META_FM_PowerOff_r(const int meta_handle, unsigned int ms_timeout);

Description:

Turn off the FM Radio module.

Return Value:

Table 6-721 The return value of META_FM_PowerOff

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-722 The parameter of META_FM_PowerOff

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.14.4 META_FM_SetFreq

Definition:

META_RESULT __stdcall META_FM_SetFreq(unsigned int ms_timeout, FM_FREQ_REQ_T *req);

META_RESULT __stdcall META_FM_SetFreq_r(const int meta_handle, unsigned int ms_timeout, FM_FREQ_REQ_T *req);

```
typedef struct{
        short m_i2CurFreq;              // freq range is [875, 1080]
}FM_FREQ_REQ_T;
```

Description:

Set the radio frequency.

Return Value:

Table 6-723 The return value of META_FM_SetFreq

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-724 The parameter of META_FM_SetFreq

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Frequency value. Range is [875-1080]

6.14.5 META_FM_GetRSSI

Definition:

META_RESULT __stdcall META_FM_GetRSSI(unsigned int ms_timeout, FM_FREQ_REQ_T *req, FM_RSSI_CNF_T *cnf);

META_RESULT __stdcall META_FM_GetRSSI_r(const int meta_handle, unsigned int ms_timeout, FM_FREQ_REQ_T *req, FM_RSSI_CNF_T *cnf);

```
typedef struct{
    short m_i2CurFreq;          // freq range is [875, 1080]
}FM_FREQ_REQ_T;


typedef struct{
   unsigned char m_ucSignalLevel;
}FM_RSSI_CNF_T;
```

Description:

Get the RSSI of the specified frequency.

Return Value:

Table 6-725 The return value of META_FM_GetRSSI

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-726 The parameter of META_FM_GetRSSI

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Frequency value. Range is [875-1080]
cnf	IN/OUT	Signal strength value.

6.14.6 META_FM_GetIfCnt

Definition:

META_RESULT __stdcall META_FM_GetIfCnt(unsigned int ms_timeout, FM_FREQ_REQ_T *req, FM_IF_CNT_CNF_T *cnf);

META_RESULT __stdcall META_FM_GetIfCnt_r(const int meta_handle, unsigned int ms_timeout, FM_FREQ_REQ_T *req, FM_IF_CNT_CNF_T *cnf);

```
typedef struct{
    short m_i2CurFreq;          // freq range is [875, 1080]
}FM_FREQ_REQ_T;


typedef struct{
    unsigned short m_u2IfCnt;
}FM_IF_CNT_CNF_T;
```

Description:

Get the IF counter value of the specified frequency.

Return Value:

Table 6-727 The return value of META_FM_GetIfCnt

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-728 The parameter of META_FM_GetIfCnt

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Frequency value. Range is [875-1080]
cnf	IN/OUT	IF counter value.

6.14.7 META_FM_SearchNextFreq

Definition:

META_RESULT __stdcall META_FM_SearchNextFreq(unsigned int ms_timeout, FM_FREQ_RANGE_REQ_T *req, FM_VAILD_FREQ_CNF_T *cnf);

META_RESULT __stdcall META_FM_SearchNextFreq_r(const int meta_handle, unsigned int ms_timeout, FM_FREQ_RANGE_REQ_T *req, FM_VAILD_FREQ_CNF_T *cnf);

```
typedef struct {             // freq range is [875, 1080]
    short m_i2StartFreq;     // note: when we try to search next: start freq should <= stop freq
    short m_i2StopFreq;      // note: when we try to search prev: start freq should >= stop freq
}FM_FREQ_RANGE_REQ_T;


typedef struct{
     unsigned char m_ucExit;        // 0: don't exist, 1: exist
     short       m_i2ValidFreq;     // -1: settings error, 0: invalid freq, others: 875-1080 valid
}FM_VAILD_FREQ_CNF_T;
```

Description:

Set a frequency range and then try to search the next frequency where we can listen to some radio programs from the start frequency to the stop frequency. Note that the start frequency should smaller than the stop frequency. Otherwise, it will be an invalid setting.

Return Value:

Table 6-729 The return value of META_FM_SearchNextFreq

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-730 The parameter of META_FM_SearchNextFreq

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Frequency range.
cnf	IN/OUT	get one/no frequency.

6.14.8 META_FM_SearchPrevFreq

Definition:

META_RESULT __stdcall META_FM_SearchPrevFreq(unsigned int ms_timeout, FM_FREQ_RANGE_REQ_T *req, FM_VAILD_FREQ_CNF_T *cnf);

META_RESULT __stdcall META_FM_SearchPrevFreq_r(const int meta_handle, unsigned int ms_timeout, FM_FREQ_RANGE_REQ_T *req, FM_VAILD_FREQ_CNF_T *cnf);

```
typedef struct {            // freq range is [875, 1080]
    short m_i2StartFreq;    // note: when we try to search next: start freq should <= stop freq
   short m_i2StopFreq;     // note: when we try to search prev: start freq should >= stop freq
}FM_FREQ_RANGE_REQ_T;


typedef struct{
     unsigned char m_ucExit;       // 0: don't exist, 1: exist
     short      m_i2ValidFreq;     // -1: settings error, 0: invalid freq, others: 875-1080 valid
}FM_VAILD_FREQ_CNF_T;
```

Description:

Set a frequency range and then try to search the previous frequency where we can listen to some radio programs from the start frequency to the stop frequency. Note that the start frequency should bigger than the stop frequency. Otherwise, it will be an invalid setting.

Return Value:

Table 6-731 The return value of META_FM_SearchPrevFreq

Return value	Description
META_SUCCESS	Success

Return value	Description
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-732 The parameter of META_FM_SearchPrevFreq

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Frequency range.
cnf	IN/OUT	get one/no frequency.

6.14.9 META_FM_SetMonoOrStereo_Blend

Definition:

META_RESULT __stdcall META_FM_SetMonoOrStereo_Blend(unsigned int ms_timeout, FM_MONO_STEREO_BLEND_REQ_T *req);

META_RESULT __stdcall META_FM_SetMonoOrStereo_Blend_r(const int meta_handle, unsigned int ms_timeout, FM_MONO_STEREO_BLEND_REQ_T *req);

```
typedef struct{
        unsigned short m_u2MonoOrStereo;     // 0: mono, 1: stereo
        unsigned short m_u2SblendOnOrOff;    // 0: sblend off, 1: sblend on
        unsigned int   m_u4ItemValue;        // 0: disable, 1: enable

}FM_MONO_STEREO_BLEND_REQ_T;
```

Description:

Set FM radio mono/stereo (sblend on/off).

Return Value:

Table 6-733 The return value of META_FM_SetMonoOrStereo_Blend

Return value	Description
META_SUCCESS	Success

Return value	Description
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-734 The parameter of META_FM_SetMonoOrStereo_Blend

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Mono/stereo settings

6.14.10 META_FM_SetRssiThreold

Definition:

META_RESULT __stdcall META_FM_SetRssiThreold(unsigned int ms_timeout, FM_RSSI_THRESHOLD_REQ_T *req);

META_RESULT __stdcall META_FM_SetRssiThreold_r(const int meta_handle, unsigned int ms_timeout, FM_RSSI_THRESHOLD_REQ_T *req);

```
typedef struct{
    unsigned int m_u4RssiThreshold;
}FM_RSSI_THRESHOLD_REQ_T;
```

Description:

Set the threshold of RSSI.

Return Value:

Table 6-735 The return value of META_FM_SetRssiThreold

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-736 The parameter of META_FM_SetRssiThreold

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The threshold value of RSSI settings

6.14.11 META_FM_SetIfCntDelta

Definition:

META_RESULT __stdcall META_FM_SetIfCntDelta(unsigned int ms_timeout, FM_IF_CNT_DELTA_REQ_T *req);

META_RESULT __stdcall META_FM_SetIfCntDelta_r(const int meta_handle, unsigned int ms_timeout, FM_IF_CNT_DELTA_REQ_T *req);

```
typedef struct{
    unsigned int m_u4IfCntDelta;
}FM_IF_CNT_DELTA_REQ_T;
```

Description:

Set the IF counter delta.

Return Value:

Table 6-737 The return value of META_FM_SetIfCntDelta

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-738 The parameter of META_FM_SetIfCntDelta

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The value of IF counter delta.

6.14.12 META_FM_ReadByte

Definition:

META_RESULT __stdcall META_FM_ReadByte(unsigned int ms_timeout, FM_READ_BYTE_ADDR_REQ_T *req, FM_READ_BYTE_CNF_T *cnf);

META_RESULT __stdcall META_FM_ReadByte_r(const int meta_handle, unsigned int ms_timeout, FM_READ_BYTE_ADDR_REQ_T *req, FM_READ_BYTE_CNF_T *cnf);

```
typedef struct{
        unsigned char m_ucAddr;
}FM_READ_BYTE_ADDR_REQ_T;

typedef struct{
       unsigned short m_u2ReadByte;
}FM_READ_BYTE_CNF_T;
```

Description:

Get the stored value in the specified register.

Return Value:

Table 6-739 The return value of META_FM_ReadByte

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-740 The parameter of META_FM_ReadByte

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The address of the register
cnf	IN/OUT	The value stored in the specified register.

6.14.13 META_FM_WriteByte

Definition:

META_RESULT __stdcall META_FM_WriteByte(unsigned int ms_timeout, FM_WRITE_BYTE_REQ_T *req);

META_RESULT __stdcall META_FM_WriteByte_r(const int meta_handle, unsigned int ms_timeout, FM_WRITE_BYTE_REQ_T *req);

```
typedef struct{
        unsigned char m_ucAddr;
        unsigned short m_u2WriteByte;
}FM_WRITE_BYTE_REQ_T;
```

Description:

Write the specified value in the specified register.

Return Value:

Table 6-741 The return value of META_FM_WriteByte

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-742 The parameter of META_FM_WriteByte

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The value we want to store and the address of the register

6.14.14 META_FM_SetSoftMute

Definition:

META_RESULT __stdcall META_FM_SetSoftMute(unsigned int ms_timeout, FM_SOFT_MUTE_ONOFF_REQ_T *req);

META_RESULT __stdcall META_FM_SetSoftMute_r(const int meta_handle, unsigned int ms_timeout, FM_SOFT_MUTE_ONOFF_REQ_T *req);

```
typedef struct{
    unsigned char m_bOnOff;       // 0: off, 1: on
}FM_SOFT_MUTE_ONOFF_REQ_T;
```

Description:

Set soft mute on/off.

Return Value:

Table 6-743 The return value of META_FM_SetSoftMute

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-744 The parameter of META_FM_SetSoftMute

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Soft mute on/off.

6.14.15 META_FM_SelectSoftMuteStage

Definition:

META_RESULT __stdcall META_FM_SelectSoftMuteStage(unsigned int ms_timeout, FM_STAGE_REQ_T *req);

META_RESULT __stdcall META_FM_SelectSoftMuteStage_r(const int meta_handle, unsigned int ms_timeout, FM_STAGE_REQ_T *req);

```
typedef struct{
    unsigned char m_ucStage;   // 1~3
}FM_STAGE_REQ_T;
```

Description:

Set soft mute stage.

Return Value:

Table 6-745 The return value of META_FM_SelectSoftMuteStage

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-746 The parameter of META_FM_SelectSoftMuteStage

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Soft mute stage.

6.14.16 META_FM_SelectSBlendStage

Definition:

META_RESULT __stdcall META_FM_SelectSBlendStage(unsigned int ms_timeout, FM_STAGE_REQ_T *req);

META_RESULT __stdcall META_FM_SelectSBlendStage_r(const int meta_handle, unsigned int ms_timeout, FM_STAGE_REQ_T *req);

```
typedef struct{
        unsigned char m_ucStage;  // 1~3
}FM_STAGE_REQ_T;
```

Description:

Set sblend stage.

Return Value:

Table 6-747 The return value of META_FM_SelectSBlendStage

Return value	Description
META_SUCCESS	Success

Return value	Description
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-748 The parameter of META_FM_SelectSBlendStage

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	sblend stage.

6.14.17 META_FM_GetHighOrLowSide

Definition:

META_RESULT __stdcall META_FM_GetHighOrLowSide(unsigned int ms_timeout, FM_FREQ_REQ_T *req, FM_HL_Side_CNF_T *cnf);

META_RESULT __stdcall META_FM_GetHighOrLowSide_r(const int meta_handle, unsigned int ms_timeout, FM_FREQ_REQ_T *req, FM_HL_Side_CNF_T *cnf);

```
typedef struct{
            short m_i2CurFreq;  // freq range is [875, 1080]
}FM_FREQ_REQ_T;


typedef struct{
           unsigned char  m_ucHighOrLow;
}FM_HL_Side_CNF_T;
```

Description:

Get the high/low side of the specified frequency.

Return Value:

Table 6-749 The return value of META_FM_GetHighOrLowSide

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-750 The parameter of META_FM_GetHighOrLowSide

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The specified frequency
cnf	IN/OUT	High/low side of the specified frequency

6.14.18 META_FM_GetStereoOrMono

Definition:

META_RESULT __stdcall META_FM_GetStereoOrMono(unsigned int ms_timeout, FM_Stereo_Mono_CNF_T *cnf);

META_RESULT __stdcall META_FM_GetStereoOrMono_r(const int meta_handle, unsigned int ms_timeout, FM_Stereo_Mono_CNF_T *cnf);

```
typedef struct{
        unsigned char m_ucStereoOrMono;
}FM_Stereo_Mono_CNF_T;
```

Description:

Get the Mono/Stereo state of the FM radio.

Return Value:

Table 6-751 The return value of META_FM_GetStereoOrMono

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-752 The parameter of META_FM_GetStereoOrMono

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

Parameter	IN/OUT	Description
cnf	IN/OUT	Mono/Stereo state of the FM Radio.

6.14.19 META_FM_GetAntennaType

Definition:

META_RESULT __stdcall META_FM_GetAntennaType(unsigned int ms_timeout, char* type);

META_RESULT __stdcall META_FM_GetAntennaType_r(const int meta_handle, int ms_timeout, char* type);

Description:

Get the antenna type from the target

Return Value:

Table 6-753 The return value of META_FM_GetAntennaType

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-754 The parameter of META_FM_GetAntennaType

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
type	OUT	Antenna type of the target (short/long)

6.14.20 META_FM_SetAntennaType

Definition:

META_RESULT __stdcall META_FM_SetAntennaType(unsigned int ms_timeout, char type);

META_RESULT __stdcall META_FM_SetAntennaType_r(const int meta_handle, int ms_timeout, char type);

Description:

Set the antenna type from the target

Return Value:

Table 6-755 The return value of META_FM_SetAntennaType

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-756 The parameter of META_FM_SetAntennaType

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
type	IN	Antenna type of the target (short/long)

6.14.21 META_FM_QueryCapArray

Definition:

META_RESULT __stdcall META_FM_QueryCapArray(unsigned int ms_timeout, float* cap_id);

META_RESULT __stdcall META_FM_QueryCapArray_r(const int meta_handle, int ms_timeout, float* cap_id);

Description:

Set the antenna type from the target

Return Value:

Table 6-757 The return value of META_FM_QueryCapArray

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-758 The parameter of META_FM_QueryCapArray

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.15 TDMB Operation

6.15.1 META_TDMB_TurnOn

Definition:

META_RESULT __stdcall META_TDMB_TurnOn(unsigned int ms_timeout);

META_RESULT __stdcall META_TDMB_TurnOn_r(const int meta_handle, unsigned int ms_timeout);

Description:

Turn on the TDMB module.

Return Value:

Table 6-759 The return value of META_TDMB_TurnOn

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-760 The parameter of META_TDMB_TurnOn

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.15.2 META_TDMB_SetBand

Definition:

META_RESULT __stdcall META_TDMB_SetBand(unsigned int ms_timeout, TDMB_SET_BAND_REQ_T *req);

META_RESULT __stdcall META_TDMB_SetBand_r(const int meta_handle, unsigned int ms_timeout, TDMB_SET_BAND_REQ_T *req);

```
typedef enum {

  META_TDMB_KOREA_BAND=1,
  META_TDMB_BAND_III,
  META_TDMB_L_BAND,
  META_TDMB_CANADA_BAND,
  META_TDMB_CHINESE_BAND,
  META_TDMB_BAND_II,
  META_TDMB_BAND_IF,
  META_TDMB_UNDEF_BAND
} META_TDMB_BAND_enum;

typedef struct{
     META_TDMB_BAND_enum  m_rBand;
}TDMB_SET_BAND_REQ_T;
```

Description:

Set the Band for the TDMB module.

Return Value:

Table 6-761 The return value of META_TDMB_SetBand

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-762 The parameter of META_TDMB_SetBand

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The band value

6.15.3 META_TDMB_AutoScan_GetFreq

Definition:

META_RESULT __stdcall META_TDMB_AutoScan_GetFreq(unsigned int ms_timeout, TDMB_AUTO_SCAN_CNF_T *cnf);

META_RESULT __stdcall META_TDMB_AutoScan_GetFreq_r(const int meta_handle, unsigned int ms_timeout, TDMB_AUTO_SCAN_CNF_T *cnf);

```
typedef struct{
        unsigned char m_ucFreqNum;
        unsigned int m_u4Freq[10];
}TDMB_AUTO_SCAN_CNF_T;
```

Description:

Autoscan to get the frequency information.

Return Value:

Table 6-763 The return value of META_TDMB_AutoScan_GetFreq

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-764 The parameter of META_TDMB_AutoScan_GetFreq

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	IN/OUT	The number of frequency, and the value of each frequency

6.15.4 META_TDMB_SetFreq

Definition:

META_RESULT __stdcall META_TDMB_SetFreq(unsigned int ms_timeout, TDMB_SET_FREQ_REQ_T *req, TDMB_SET_FREQ_CNF_T *cnf);

META_RESULT __stdcall META_TDMB_SetFreq_r(const int meta_handle, unsigned int ms_timeout, TDMB_SET_FREQ_REQ_T *req, TDMB_SET_FREQ_CNF_T *cnf);

```
typedef struct{
    unsigned int  m_u4Freq;
}TDMB_SET_FREQ_REQ_T;

typedef struct
{
        char m_cResult;  // 0: success,
                            // 1: the band not exist  ==> META_TDMB_ERR_BAND_NOT_EXIST
                           // 2: frequency not exist ==> META_TDMB_ERR_FREQ_NOT_EXIST
        unsigned char     m_ucEnsembleNum;
      TDMB_ENSEMBLEDB_T  m_rEnsembleDB[10];
        TDMB_ENSEMBLEDB_T  m_rCurEnsembleDB;
}TDMB_SET_FREQ_CNF_T;
```

Description:

Set the frequency, and then get the ensemble information.

Return Value:

Table 6-765 The return value of META_TDMB_SetFreq

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-766 The parameter of META_TDMB_SetFreq

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The frequency for TDMB module
cnf	IN/OUT	The ensemble information

6.15.5 META_TDMB_AutoScan_GetEnsemble

Definition:

META_RESULT __stdcall META_TDMB_AutoScan_GetEnsemble(unsigned int ms_timeout, TDMB_GET_ENSM_INFO_BY_AUTO_SCAN_CNF_T *cnf);

META_RESULT __stdcall META_TDMB_AutoScan_GetEnsemble_r(const int meta_handle, unsigned int ms_timeout, TDMB_GET_ENSM_INFO_BY_AUTO_SCAN_CNF_T *cnf);

```
typedef struct{
  unsigned char     m_ucEnsembleNum;
  TDMB_ENSEMBLEDB_T  m_rEnsembleDB[10];
}TDMB_GET_ENSM_INFO_BY_AUTO_SCAN_CNF_T;
```

Description:

Set the frequency, and then get the ensemble information.

Return Value:

Table 6-767 The return value of META_TDMB_AutoScan_GetEnsemble

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-768 The parameter of META_TDMB_AutoScan_GetEnsemble

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	IN/OUT	The ensemble information

6.15.6 META_TDMB_GetSignal

Definition:

META_RESULT __stdcall META_TDMB_GetSignal(unsigned int ms_timeout, TDMB_GET_SIGNAL_CNF_T *cnf);

META_RESULT __stdcall META_TDMB_GetSignal_r(const int meta_handle, unsigned int ms_timeout, TDMB_GET_SIGNAL_CNF_T *cnf);

```
typedef struct{
        unsigned short m_u2Snr;
        unsigned short m_u2PostBer; // not provided so far, so return 0
        unsigned short m_u2PreBer;
        unsigned short m_u2RSSI;
}TDMB_GET_SIGNAL_CNF_T;
```

Description:

After selecting a service, the signal information can be retrieved.

Return Value:

Table 6-769 The return value of META_TDMB_GetSignal

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-770 The parameter of META_TDMB_GetSignal

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	IN/OUT	The signal information

6.15.7 META_TDMB_SelService

Definition:

META_RESULT __stdcall META_TDMB_SelService(unsigned int ms_timeout, TDMB_SEL_SERVICE_REQ_T *req, const META_TDMB_SEL_SERV_CNF cnf_cb);

META_RESULT __stdcall META_TDMB_SelService_r(const int meta_handle, unsigned int ms_timeout, TDMB_SEL_SERVICE_REQ_T *req, const META_TDMB_SEL_SERV_CNF cnf_cb);

```
typedef struct{
        unsigned int m_u4ServiceId;
        unsigned int m_u4SubChnId;
        char     *m_pcfilepath; // store the TS stream data to this file
}TDMB_SEL_SERVICE_REQ_T;

typedef void (__stdcall *META_TDMB_SEL_SERV_CNF)(const TDMB_SEL_SERV_ERROR_RESULT status);
```

Description:

Select a service, and then the parsed TS stream data will be stored in target's file system (NAND flash or SD card) or MED memory(around 3Mbytes). After calling META_TDMB_SetIdle( ), the content of file/MED memory will be transmitted to the PC side and be auto-deleted/auto-released. So, remember to call META_TDMB_SetIdle( ) after select a service successfully.

Return Value:

Table 6-771 The return value of META_TDMB_SelService

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-772 The parameter of META_TDMB_SelService

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Service information
cnf_cb	IN/OUT	Function pointer of possible error callback.

6.15.8 META_TDMB_SetIdle

Definition:

META_RESULT __stdcall META_TDMB_SetIdle(unsigned int ms_timeout, CALLBACK_META_FAT_PROGRESS cb_progress, void *cb_progress_arg);

META_RESULT __stdcall META_TDMB_SetIdle_r(const int meta_handle, unsigned int ms_timeout, CALLBACK_META_FAT_PROGRESS cb_progress, void *cb_progress_arg);

typedef int (__stdcall *CALLBACK_META_FAT_PROGRESS)(unsigned char percent, int sent_bytes, int total_bytes, const short token, void *usr_arg);

Description:

Stop TDMB module from parsing the TS stream data, and get the stored file in target's FAT/ stored content in target's memory. At last, delete the file/release the memory from the target side. Note that, this function must be called after a META_TDMB_SelService() was called before.

Return Value:

Table 6-773 The return value of META_TDMB_SetIdle

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-774 The parameter of META_TDMB_SetIdle

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cb_progress	IN/OUT	Function pointer of progress callback.
cb_progress_arg	IN/OUT	User argument that will be used into callback function.

6.15.9 META_TDMB_TurnOff

Definition:

META_RESULT __stdcall META_TDMB_TurnOff(unsigned int ms_timeout);

META_RESULT __stdcall META_TDMB_TurnOff_r(const int meta_handle, unsigned int ms_timeout);

Description:

Turn off the TDMB module.

Return Value:

Table 6-775 The return value of META_TDMB_TurnOff

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-776 The parameter of META_TDMB_TurnOff

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.15.10 META_TDMB_GetEnsm

Definition:

META_RESULT __stdcall META_TDMB_GetEnsm(unsigned int ms_timeout, TDMB_GET_ENSM_CNF_T *cnf);

META_RESULT __stdcall META_TDMB_GetEnsm_r(const int meta_handle, unsigned int ms_timeout, TDMB_GET_ENSM_CNF_T *cnf);

```
typedef struct {
        unsigned char     m_ucEnsembleNum;
       TDMB_ENSEMBLEDB_T  m_rEnsembleDB[10];
        TDMB_ENSEMBLEDB_T  m_rCurEnsembleDB;
}TDMB_GET_ENSM_CNF_T;
```

Description:

Retrieve the Ensemble information.

Return Value:

Table 6-777 The return value of META_TDMB_GetEnsm

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-778 The parameter of META_TDMB_GetEnsm

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	IN/OUT	The ensemble information.

6.15.11 META_TDMB_SelServiceOnly

Definition:

META_RESULT __stdcall META_TDMB_SelServiceOnly(unsigned int ms_timeout, TDMB_SEL_SERVICE_ONLY_REQ_T *req);

META_RESULT __stdcall META_TDMB_SelServiceOnly_r(const int meta_handle, unsigned int ms_timeout, TDMB_SEL_SERVICE_ONLY_REQ_T *);

```
typedef struct {
        unsigned int m_u4ServiceId;
        unsigned int m_u4SubChnId;
}TDMB_SEL_SERVICE_ONLY_REQ_T;
```

Description:

Select the service without storing the parsed TS stream data in the target FAT. If you just want to retrieve the signal information of some service, this function will be helpful.

Return Value:

Table 6-779 The return value of META_TDMB_SelServiceOnly

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-780 The parameter of META_TDMB_SelServiceOnly

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	Service settings.

6.15.12 META_TDMB_StopAutoScan

Definition:

META_RESULT __stdcall META_TDMB_StopAutoScan(unsigned int ms_timeout);

META_RESULT __stdcall META_TDMB_StopAutoScan_r(const int meta_handle, unsigned int ms_timeout);

Description:

Ask target to stop the operation of auto scan.

Return Value:

Table 6-781 The return value of META_TDMB_StopAutoScan

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-782 The parameter of META_TDMB_StopAutoScan

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.16 Exported functions for Backup and Restore Calibration Data

INI file format:

```
[Backup and Restore Calibration Data Basic Settings]
Backup NVRAM folder path=Z:\NVRAM\NVD_DATA\
Restore NVRAM folder path=Z:\NVRAM\NVD_DATA\

[Target NVRAM Backup and Restore List]
; XXXLID = xxx_File_Prefix(Note: the length of file prefix = 4)

[Target Backup and Restore List]
;xxx full path of old target= yyy full path of new target

[Upload PC files to Target List]

; xxx full path of PC side = yyy full path of target

[Backup/Restore File Prefix-MORE]

; not supported until  META_DLL_v5.0920.0

; the file prefix length must = 4

; sample

;file_prefix1=abcd

;file_prefix2=efgh


[Backup/Restore File Prefix-DELETE]

; not supported until  META_DLL_v5.0920.0

; the file prefix length must = 4

; can not define the same file prefix in SEC: Backup/Restore File Prefix-MORE

; sample

;file_prefix1=ijkl

;file_prefix2=mnop
```

Internal NVRAM file-prefix superset:

Table 6-783 Internal NVRAM file-prefix superset

LID	File-Prefix	File-Prefix
WIFI		
NVRAM_EF_WNDRV_TX_POWER_2400M_LID	WIFI	
NVRAM_EF_WNDRV_TX_POWER_5000M_LID	WIFI	
NVRAM_EF_WNDRV_DAC_DC_OFFSET_LID	WIFI	
NVRAM_EF_WNDRV_TX_ALC_POWER_LID	WIFI	
NVRAM_EF_WNDRV_ALC_SLOPE_LID	WIFI	
NVRAM_EF_WNDRV_MAC_ADDRESS_LID	WIFI	
BT		
NVRAM_EF_BTRADIO_MT6601_LID	MP27	
NVRAM_EF_BTRADIO_MT6611_LID	MP28	
RF		
NVRAM_EF_L1_AGCPATHLOSS_LID	MT05	
NVRAM_EF_L1_RAMPTABLE_GSM850_LID	MT06	
NVRAM_EF_L1_RAMPTABLE_GSM900_LID	MT07	
NVRAM_EF_L1_RAMPTABLE_DCS1800_LID	MT08	
NVRAM_EF_L1_RAMPTABLE_PCS1900_LID	MT09	
NVRAM_EF_L1_EPSK_RAMPTABLE_GSM850_LID	MT0A	
NVRAM_EF_L1_EPSK_RAMPTABLE_GSM900_LID	MT0B	
NVRAM_EF_L1_EPSK_RAMPTABLE_DCS1800_LID	MT0C	

LID	File-Prefix	File-Prefix
NVRAM_EF_L1_EPSK_RAMPTABLE_PCS1900_LID	MT0D	
NVRAM_EF_L1_INTERSLOT_RAMP_GSM850_LID	MT0L	
NVRAM_EF_L1_INTERSLOT_RAMP_GSM900_LID	MT0M	
NVRAM_EF_L1_INTERSLOT_RAMP_DCS1800_LID	MT0N	
NVRAM_EF_L1_INTERSLOT_RAMP_PCS1900_LID	MT0O	
NVRAM_EF_L1_EPSK_INTERSLOT_RAMP_GSM850_LID	MT0E	
NVRAM_EF_L1_EPSK_INTERSLOT_RAMP_GSM900_LID	MT0F	
NVRAM_EF_L1_EPSK_INTERSLOT_RAMP_DCS1800_LID	MT0G	
NVRAM_EF_L1_EPSK_INTERSLOT_RAMP_PCS1900_LID	MT0H	
NVRAM_EF_L1_AFCDATA_LID	MT0I	
NVRAM_EF_L1_TXIQ_LID	MT0J	
NVRAM_EF_L1_RFSPECIALCOEF_LID	MT0K	
NVRAM_EF_L1_CRYSTAL_AFCDATA_LID	MT0P	
NVRAM_EF_L1_CRYSTAL_CAPDATA_LID	MT0Q	
BB		
NVRAM_EF_ADC_LID	MP00	MP0W
NVRAM_EF_BARCODE_NUM_LID	MP09	MP0X

6.16.1 META_BackupCalibrationData

Definition:

META_RESULT __stdcall META_BackupCalibrationData(const MISC_BACKUP_REQ_T *req, int *p_backupstop);

```
META_RESULT __stdcall META_BackupCalibrationData_r(const int meta_handle, const MISC_BACKUP_REQ_T *req, int *p_backupstop);

typedef struct
{
        char                *m_pIniFilePath;
        char                *m_pBackupFolderPath;
        CALLBACK_MISC_PROGRESS  cb_progress;
        void                *cb_progress_arg;
}MISC_BACKUP_REQ_T;


typedef void (__stdcall *CALLBACK_MISC_PROGRESS)(unsigned char m_u1TotalNum, unsigned char m_u1BackupNum, void *usr_arg);
```

Description:

Base on the INI file to backup target 's calibration data and to download other target's file to PC side's backup folder.

Note: the function can only be used on W0829~later MAUI and 08A load.

Note: only sections [Target Backup and Restore List], [Upload PC files to Target List], [Backup/Restore File Prefix-MORE] and [Backup/Restore File Prefix-DELETE] will be processed.

Flow

1. **Check Target NVRAM Attribute Completeness**
2. **Check INI setting**
3. **Collect Target NVRAM Backup List**
4. **Check NVRAM folder has all files we need to take care**
   - (*CheckAllTargetNVRAMFolderHasAllFilesWeWant*)
5. **Backup Calibration Data**
   - (*META_BackupCalibrationData_EX)*

There are 2 steps can be customized in backup phase

- Check the file list of NVRAM folder with the calibration data item set (*CheckAllTargetNVRAMFolderHasAllFilesWeWant*)
- Backup file from the target (*META_BackupCalibrationData_EX)*

Return Value:

Table 6-784 The return value of META_BackupCalibrationData

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-785 The parameter of META_BackupCalibrationData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	req-> m_pIniFilePath: the INI file path req-> m_pBackupFolderPath: the folder path to store the backup files. req->cb_progress: the callback function to display the backup progress. req-> cb_progress_arg: an argument can be passed to the callback function.

6.16.2 META_BasicBackupCalibrationData

Definition:

META_RESULT __stdcall META_BasicBackupCalibrationData(const MISC_BACKUP_REQ_T *req, int *p_backupstop);

META_RESULT __stdcall META_BasicBackupCalibrationData_r(const int meta_handle, const MISC_BACKUP_REQ_T *req, int *p_backupstop);

```
typedef struct
{
        char            *m_pIniFilePath;
        char            *m_pBackupFolderPath;
        CALLBACK_MISC_PROGRESS  cb_progress;
        void            *cb_progress_arg;
```

}MISC_BACKUP_REQ_T;

typedef void (__stdcall *CALLBACK_MISC_PROGRESS)(unsigned char m_u1TotalNum, unsigned char m_u1BackupNum, void *usr_arg);

Description:

Base on the INI file and internal NVRAM file-prefix superset to backup target 's calibration data to PC side's folder.
Note: the function can only be used on load before W0829 (not including W0829)
Note: All 4 sections in INI file will be processed.

Return Value:

Table 6-786 The return value of META_BasicBackupCalibrationData

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-787 The parameter of META_BasicBackupCalibrationData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	req-> m_pIniFilePath: the INI file path req-> m_pBackupFolderPath: the folder path to store the backup files. req->cb_progress: the callback function to display the backup progress. req-> cb_progress_arg: an argument can be passed to the callback function.

6.16.3 META_RestoreCalibrationData

Definition:

META_RESULT __stdcall META_RestoreCalibrationData(const MISC_RESTORE_REQ_T *req, int *p_restorestop);

META_RESULT __stdcall META_RestoreCalibrationData_r(const int meta_handle, const MISC_RESTORE_REQ_T *req, int *p_restorestop);

```
typedef struct
{
        char              *m_pIniFilePath;
        char              *m_pBackupFolderPath;           // the folder which store the backup data
      CALLBACK_MISC_PROGRESS   cb_progress;
        void              *cb_progress_arg;

}MISC_RESTORE_REQ_T;
```

```
typedef void (__stdcall *CALLBACK_MISC_PROGRESS)(unsigned char m_u1TotalNum, unsigned char m_u1BackupNum, void *usr_arg);
```

Description:

Base on the INI file to restore calibration data to target's NVRAM folder, and upload other PC files to target's file system.

Note: the function can only be used on W0829~later MAUI and 08A load.

Note: only sections [Target Backup and Restore List] and [Upload PC files to Target List] will be processed.

Flow

1. **Check Target NVRAM Attribute Completeness**
2. **Check INI settings**
3. **Collect Target NVRAM Backup List**
4. **Check NVRAM folder has all files we need to take care**
   - (*CheckAllTargetNVRAMFolderHasAllFilesWeWant*)
5. **Verify Backup Result about NVRAM files**
   - (VerifyBackupNVRAMResultForRestorePhase)
6. **Restore calibration data**
   - (*META_RestoreCalibrationData_EX*)

There are 3 steps can be customized in restore phase

1. Check the file list of NVRAM folder with the calibration data item set (*CheckAllTargetNVRAMFolderHasAllFilesWeWant*)
2. Verify PC backup folder matches the calibration data item set (VerifyBackupNVRAMResultForRestorePhase)
3. Restore file from the target (*META_RestoreCalibrationData_EX*)

Return Value:

Table 6-788 The return value of META_RestoreCalibrationData

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-789 The parameter of META_RestoreCalibrationData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	req-> m_pIniFilePath: the INI file path req-> m_pBackupFolderPath: the folder path of the backup folder. req->cb_progress: the callback function to display the restore progress. req-> cb_progress_arg: an argument can be passed to the callback function.

6.16.4 META_BasicRestoreCalibrationData

Definition:

META_RESULT __stdcall META_BasicRestoreCalibrationData(const MISC_RESTORE_REQ_T *req, int *p_restorestop);

META_RESULT __stdcall META_BasicRestoreCalibrationData_r(const int meta_handle, const MISC_RESTORE_REQ_T *req, int *p_restorestop);

```
typedef struct
{
        char        *m_pIniFilePath;
        char        *m_pBackupFolderPath;        // the folder which store the backup data
        CALLBACK_MISC_PROGRESS   cb_progress;
          void                  *cb_progress_arg;

}MISC_RESTORE_REQ_T;
```

typedef void (__stdcall *CALLBACK_MISC_PROGRESS)(unsigned char m_u1TotalNum, unsigned char m_u1BackupNum, void *usr_arg);

Description:

Base on the INI file and internal NVRAM file-prefix superset to restore calibration data to target's NVRAM folder, and upload other PC files to target's file system.
Note: the function can only be used on load before W0829 (not including W0829).
Note: All 4 sections in INI file will be processed.

Return Value:

Table 6-790 The return value of META_BasicRestoreCalibrationData

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-791 The parameter of META_BasicRestoreCalibrationData

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	req-> m_pIniFilePath: the INI file path req-> m_pBackupFolderPath: the folder path of the backup folder. req->cb_progress: the callback function to display the restore progress. req-> cb_progress_arg: an argument can be passed to the callback function.

6.16.5 META_GetBackupResultInfo

Definition:

```
META_RESULT __stdcall META_GetBackupResultInfo(const char *backup_folder, BACKUP_RESULT_T *cnf);

META_RESULT __stdcall META_GetBackupResultInfo_r(const int meta_handle, const char *backup_folder, BACKUP_RESULT_T *cnf);


typedef struct

{

        char            m_strBackupFolder[MAX_PATH];

        bool            m_bISNewLoad;

        META_IMEI_LOC_enum   m_enumImeiLoc; // only valid when m_bISNewLoad = true;

        unsigned  char      m_ImeiData[10]; // only valid when m_bISNewLoad = true

        int             m_i4ComPort;

        int             m_i4BackupFileNum;


}BACKUP_RESULT_T;
```

Description:

Return the backup result.

Return Value:

Table 6-792 The return value of META_GetBackupResultInfo

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-793 The parameter of META_GetBackupResultInfo

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().

Parameter	IN/OUT	Description
ms_timeout	IN	Timeout value, unit = minisecond
backup_folder	IN	The folder path of the backup folder.
cnf	IN/OUT	cnf-> m_strBackupFolder[MAX_PATH]: The folder path of the backup folder. cnf-> m_bISNewLoad: which API we use to backup: META_BackupCalibrationData(true) or META_BasicBackupCalibrationData(false) cnf-> m_enumImeiLoc: the storage location of IMEI when we do backup. cnf-> m_ImeiData[10: the IMEI content when we do backup. cnf-> m_i4ComPort: the COM port we use when we do backup cnf-> m_i4BackupFileNum: the total number of backup files

6.16.6 META_GetRestoreResultInfo

Definition:

META_RESULT __stdcall META_GetRestoreResultInfo(const char *backup_folder, RESTORE_RESULT_T *cnf);

META_RESULT __stdcall META_GetRestoreResultInfo_r(const int meta_handle, const char *backup_folder, RESTORE_RESULT_T *cnf);

```
typedef struct
{

    char                m_strRestoreFromFolder[MAX_PATH];
    bool                m_bISNewLoad;
    META_IMEI_LOC_enum     m_enumImeiLoc; // only valid when m_bISNewLoad = true;
    unsigned char          m_ImeiData[10]; // only valid when m_bISNewLoad = true;
    int               m_i4ComPort;
    int               m_i4BackupFileNum;

}RESTORE_RESULT_T;
```

Description:

Return the backup result.

Return Value:

Table 6-794 The return value of META_GetRestoreResultInfo

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-795 The parameter of META_GetRestoreResultInfo

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
backup_folder	IN	The folder path of the backup folder.
cnf	IN/OUT	cnf-> m_strRestoreFromFolder [MAX_PATH]: The folder path of the backup folder. cnf-> m_bISNewLoad: which API we use to restore: META_RestoreCalibrationData(true) or META_BasicRestoreCalibrationData(false) cnf-> m_enumImeiLoc: the storage location of IMEI when we do restore. cnf-> m_ImeiData[10: the IMEI content when we do restore. cnf-> m_i4ComPort: the COM port we use when we do restore cnf-> m_i4BackupFileNum: the total number of restore files

6.16.7 META_DeleteAllFilesInBackupFolder

Definition:

META_RESULT __stdcall META_DeleteAllFilesInBackupFolder(const char *pBackupFolderPath);

META_RESULT __stdcall META_DeleteAllFilesInBackupFolder_r(const int meta_handle, const char *pBackupFolderPath);

Description:

Delete all files stored in backup folder.

Return Value:

Table 6-796 The return value of META_DeleteAllFilesInBackupFolder

Return value	Description
META_SUCCESS	Success

Return value	Description
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-797 The parameter of META_DeleteAllFilesInBackupFolder

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
pBackupFolderPath	IN	The folder path of the backup folder.

6.16.8 META_UploadFilesToTarget

Definition:

```
META_RESULT __stdcall META_UploadFilesToTarget(MISC_UPLOAD_REQ_T *req, int *p_uploadstop);

META_RESULT __stdcall META_UploadFilesToTarget_r(const int meta_handle, MISC_UPLOAD_REQ_T *req, int *p_uploadstop);


typedef struct
{
        char                          *m_pIniFilePath; // the INI file path
       CALLBACK_MISC_PROGRESS  cb_progress;
        void                           *cb_progress_arg;

}MISC_UPLOAD_REQ_T;


typedef void (__stdcall *CALLBACK_MISC_PROGRESS)(unsigned char m_u1TotalNum, unsigned char m_u1BackupNum, void *usr_arg);
```

Description:

Upload files specified in the following section which is specified in the INI file. [Upload PC files to Target List]

; xxx full path of PC side = yyy full path of target

Return Value:

Table 6-798 The return value of META_UploadFilesToTarget

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-799 The parameter of META_UploadFilesToTarget

Parameter	IN/OUT	Description
meta_handle	IN	Handle of META_DLL that return from META_GetAvailableHandle().
req	IN	req-> m_pIniFilePath: the INI file path req->cb_progress: the callback function to display the restore progress. req-> cb_progress_arg: an argument can be passed to the callback function.

6.16.9 META_MISC_SetBackupRestoreErrorCallback

Definition:

META_RESULT __stdcall META_MISC_SetBackupRestoreErrorCallback(CALLBACK_BKRS_ERROR_HANDLER cb, void* user_arg);

META_RESULT __stdcall META_MISC_SetBackupRestoreErrorCallback_r(const int meta_handle, CALLBACK_BKRS_ERROR_HANDLER cb, void* user_arg);

```
typedef struct
{
   // full path to the file
   const char*   fullPath;
   // file size (0: means not-avaialble in the context)
   int         fileSize;
   // LID name or enum value
   const char*   lidOrEnum;
   // type of the NVRAM file (0: normal, 1: imei, 2: SML)
   unsigned char fileType;
}META_MISC_RestoreFileNotFoundInBackupResult_T;
```

```
typedef struct
{
   // file prefix of the NVRAM item
   const char*    filePrefix;
   // verno of the NVRAM item
   const char*    versionNumber;
   // enum value
   unsigned short enumValue;
   // type of the NVRAM file (0: normal, 1: imei, 2: SML)
   unsigned char  fileType;
   // file size (0: means not-avaialble in the context)
   unsigned int   fileSize;
}META_MISC_BackupFileNotFoundInNvram_T;
typedef META_MISC_BackupFileNotFoundInNvram_T META_MISC_RestoreTargetNotFoundInNvram_T;
typedef struct
{
   // key name
   const char*   keyName;
   // value string
   const char*   value;
}META_MISC_BackupMoreFileNotFoundInNvram_T;
typedef struct
{
   // where we download from the target side
   const char*     backupPath;
   unsigned int    fileSize;
   // 1: nvram sec, 2: target sec
   unsigned char   fileSection;
   // where we store the files in PC side
```

```
    const char*     localPath;
    bool            hasLidInfo;
    // meaningful when m_bHasLID == true;
    const char*     lidInfo;
    // -1: not exist 0: general LID, 1: IMEI, 2: SML
    char            lidType;
    // store the target file path we will restore!
    const char*     restorePath;
}META_MISC_BackupFileResultEntry_T;
typedef struct
{
    META_MISC_BackupFileResultEntry_T             backupResult;
    META_MISC_RestoreFileNotFoundInBackupResult_T  restoreFileInfo;
}META_MISC_BackupFileRestoreTargetSizeMismatch_T;
typedef union
{
    META_MISC_RestoreFileNotFoundInBackupResult_T  restoreFileNotFoundInBackupResultInfo;
    META_MISC_BackupFileNotFoundInNvram_T          backupFileNotFoundInNvramInfo;
    META_MISC_RestoreTargetNotFoundInNvram_T       restoreTargetNotFoundInNvramInfo;
    META_MISC_BackupMoreFileNotFoundInNvram_T      backupMoreFileNotFoundInNvramInfo;
    META_MISC_BackupFileRestoreTargetSizeMismatch_T backupFileRestoreTargetSizeMismatchInfo;
    DWORD                                          systemErrorCode;
}META_MISC_BKRSCustomizedInformation;
typedef struct
{
    META_RESULT errorCode;
    const char* message;
    int messageLength;
    META_MISC_BKRSCustomizedInformation info;
}META_MISC_BKRSCustomizedCallbackParameter;
```

typedef int (__stdcall *CALLBACK_BKRS_ERROR_HANDLER)(const META_MISC_BKRSCustomizedCallbackParameter *param, void* userArg);

Description:

Register custom defined callback function to handle certain error condition.

Return Value:

Table 6-800 The return value of META_MISC_SetBackupRestoreErrorCallback

Return value	Description
META_SUCCESS	Success
Other error code	Other error messages please use META_GetErrorString to translate the meaning.

Customizable error condition:

Table 6-801 The parameter of META_MISC_SetBackupRestoreErrorCallback

Error code	Description	Callback parameterer
META_MISC_RETORE_FILE_NOT_FOUND_IN_BACKUP_RESULT	The target ask for certain file to be restored, but the file is not kept in backup phase.	restoreFileNotFoundInBackupResultInfo
META_MISC_BACKUP_FILE_NOT_FOUND_IN_NVRAM	The target ask for certain file to be backuped, but the file is not found in NVRAM folder on target side in backup phase.	backupFileNotFoundInNvramInfo
META_MISC_RESTORE_TARGET_NOT_FOUND_IN_NVRAM	The target ask for certain file to be restored, but the file is not found in NVRAM folder on target side in restore phase. Probably some items are added as calibration data or important data in restore phase.	restoreTargetNotFoundInNvramInfo
META_MISC_BACKUP_MORE_FILE_NOT_FOUND_IN_NVRAM	The entry in [Backup/Restore File Prefix-MORE] section of BACKUP.ini cannot be found in NVRAM folder in backup phase.	backupMoreFileNotFoundInNvramInfo
META_MISC_FILE_SIZE_MISMATCH	The backup file size is inconsistent with the restore target. Probably the item is changed.	backupFileRestoreTargetSizeMismatchInfo
META_MISC_LEGACY_ADC_FILE_NOT_FOUND	The legacy ADC NVRAM file is not found	N/A
META_MISC_LEGACY_BARCODE_FILE_NOT_FOUND	The legacy barcode NVRAM file is not found	N/A

6.17 CMMB Operation

6.17.1 META_CMMB_TurnOn

Definition:

META_RESULT __stdcall META_CMMB_TurnOn(const unsigned int ms_timeout);

META_RESULT __stdcall META_CMMB_TurnOn_r(const int meta_handle, const unsigned int ms_timeout);

Description:

Turn on the CMMB module.

Return Value:

Table 6-802 The return value of META_CMMB_TurnOn

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-803 The parameter of META_CMMB_TurnOn

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.17.2 META_CMMB_TurnOff

Definition:

META_RESULT __stdcall META_CMMB_TurnOff(const unsigned int ms_timeout);

META_RESULT __stdcall META_CMMB_TurnOff_r(const int meta_handle, const unsigned int ms_timeout);

Description:

Turn off the CMMB module.

Return Value:

Table 6-804 The return value of META_CMMB_TurnOff

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-805 The parameter of META_CMMB_TurnOff

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.17.3 META_CMMB_SetBand

Definition:

META_RESULT __stdcall META_CMMB_SetBand(const unsigned int ms_timeout, const META_CMMB_SET_BAND_REQ_T *req);

META_RESULT __stdcall META_CMMB_SetBand_r(const int meta_handle, const unsigned int ms_timeout, const META_CMMB_SET_BAND_REQ_T *req);

```
Typedef enum
{
   META_CMMB_CHINA_U_BAND=0
  ,META_CMMB_TAIWAN_BAND
  ,META_CMMB_UNDEF_BAND
} META_CMMB_BAND_enum;

typedef struct
{
    META_CMMB_BAND_enum  m_rBand;
```

}META_CMMB_SET_BAND_REQ_T;

Description:

Set the band for the CMMB module.

Return Value:

Table 6-806 The return value of META_CMMB_SetBand

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-807 The parameter of META_CMMB_SetBand

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The band value

6.17.4 META_CMMB_AutoScanGetFreq

Definition:

META_RESULT __stdcall META_CMMB_AutoScanGetFreq(const unsigned int ms_timeout, META_CMMB_AUTO_SCAN_GET_FREQ_CNF_T *cnf);

META_RESULT __stdcall META_CMMB_AutoScanGetFreq_r(const int meta_handle, const unsigned int ms_timeout, META_CMMB_AUTO_SCAN_GET_FREQ_CNF_T *cnf);

```
typedef struct
{
   unsigned char   m_u1FreqPointId;
    unsigned char   m_u1BandWidth;
    unsigned int    m_u4Freq;
}META_CMMB_FreqBandStruct_T;
```

```
typedef struct
{
        unsigned char                      m_u1MainFreqNum;
        META_CMMB_FreqBandStruct_T     m_rMainFreqBand[META_CMMB_FREQ_BAND_NUM];
}META_CMMB_AUTO_SCAN_GET_FREQ_CNF_T;
```

Description:

Request the CMMB module of the target to perform auto-scan operations to get the frequency after a band is selected.

Return Value:

Table 6-808 The return value of META_CMMB_AutoScanGetFreq

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-809 The parameter of META_CMMB_AutoScanGetFreq

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	OUT	How many frequencies supported by this band, and their detailed information

6.17.5 META_CMMB_AutoScan

Definition:

META_RESULT __stdcall META_CMMB_AutoScan(const unsigned int ms_timeout, META_CMMB_AUTO_SCAN_CNF_T *cnf);

META_RESULT __stdcall META_CMMB_AutoScan_r(const int meta_handle, const unsigned int ms_timeout, META_CMMB_AUTO_SCAN_CNF_T *cnf);

```
#define        META_CMMB_BLK_NUM                                        8
#define                          META_CMMB_SERV_BLOCK_NUM               20
```

```
#define                 META_CMMB_DATA_BLK_NUM                                      128
#define                         META_CMMB_FRAME_INFO_NUM                              4

typedef struct

{

     unsigned char              Nit_NitUpdateSeq;

    unsigned char              Nit_SysTime[5]   ;

    unsigned int               Nit_CountryCode ;

    unsigned char              Nit_Net_NetLevel;

    unsigned short              Nit_Net_NetId;

    unsigned char              Nit_NetNameLen  ;

    unsigned char              Nit_NetName[128];

     unsigned char              Nit_FreqBand_FreqPointId;

    unsigned char               Nit_FreqBand_BandWidth;

    unsigned int                Nit_FreqBand_CenterFreq;

    unsigned char              Nit_OtherFreqNum;

    unsigned char              m_ucOtherFreqNumWeCarry;



   unsigned char              Nit_OtherFreqBandList_FreqPointId[META_CMMB_BLK_NUM];

   unsigned char              Nit_OtherFreqBandList_BandWidth[META_CMMB_BLK_NUM];

   unsigned int               Nit_OtherFreqBandList_CenterFreq[META_CMMB_BLK_NUM];

   unsigned char              Nit_NeighborNetNum;

   unsigned char              m_ucNeightborNetWeCarray;

   unsigned char              Nit_NeighborNetList_NetLevel[META_CMMB_BLK_NUM];

   unsigned short              Nit_NeighborNetList_NetId[META_CMMB_BLK_NUM];

   unsigned char              Nit_NeighborNetList_FreqPointId[META_CMMB_BLK_NUM];

   unsigned char              Nit_NeighborNetList_BandWidth[META_CMMB_BLK_NUM];

   unsigned int              Nit_NeighborNetList_CenterFreq[META_CMMB_BLK_NUM];
```

```
}META_CMMB_NitStruct_T;


typedef struct
{
    unsigned char              MctUpdateSeq;
    unsigned char              FreqPointId;
    unsigned char              MfNum;
    unsigned char              m_ucMfNumWeCarray;
    unsigned char              Mf_MfId[META_CMMB_BLK_NUM];
    unsigned char              Mf_RsRate[META_CMMB_BLK_NUM];
    unsigned char              Mf_ByteInterleaveMode[META_CMMB_BLK_NUM];
    unsigned char              Mf_LdpcRate[META_CMMB_BLK_NUM];
    unsigned char              Mf_ModulationMode[META_CMMB_BLK_NUM];
    unsigned char              Mf_ScrambleMode[META_CMMB_BLK_NUM];
    unsigned char              Mf_TimeSlotNum[META_CMMB_BLK_NUM];
    unsigned char              m_ucMf_TimeSlotNumWeCarray[META_CMMB_BLK_NUM];
    unsigned char              Mf_TimeSlotId[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];
    unsigned char              Mf_SubMfNum[META_CMMB_BLK_NUM];
    unsigned char              m_ucMf_SubMfNumWeCarry[META_CMMB_BLK_NUM];
    unsigned char              Mf_SubMfId[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];
    unsigned short             Mf_serviceId[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];

}META_CMMB_MctStruct_T;


typedef struct
{
   unsigned char           SctUpdateSeq;
   unsigned short          ServiceNum;
```

```
  unsigned char              m_u1ServiceNumWeCarray;
  unsigned short             ServiceId[META_CMMB_SERV_BLOCK_NUM];
  unsigned char              FreqPointId[META_CMMB_SERV_BLOCK_NUM];

}META_CMMB_SctStruct_T;

typedef struct
{
  unsigned char           EsgUpdateSeq;
  unsigned char           NetLevel;
  unsigned short          NetId;
  unsigned char           LocalTimeOffset;
  unsigned char           CharSet;
  unsigned char           EsgServiceNum;
  unsigned char           m_ucEsgServiceNumWeCarry;
  unsigned char           EsgService_EsgServiceIndex[META_CMMB_BLK_NUM];
  unsigned short          EsgService_EsgServiceId[META_CMMB_BLK_NUM];
  unsigned char           EsgDataNum;
  unsigned char           m_ucEsgDataNumWeCarry;
  unsigned char           EsgData_EsgDataType[META_CMMB_BLK_NUM];
  unsigned char           EsgData_EsgDataBlockNum[META_CMMB_BLK_NUM];
  unsigned char           m_ucEsgData_EsgDataBlockNumWeCarry[META_CMMB_BLK_NUM];
  unsigned char           EsgDataBlock_EsgDataBlockId[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];
  unsigned char         EsgDataBlock_EsgDataBlockVersion[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];
  unsigned char           EsgDataBlock_EsgServiceIndex[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];

}META_CMMB_EsgListStruct_T;
```

```
typedef struct
{
    unsigned char         CaUpdateSeq;
    unsigned short        CaDataNum;
    unsigned char         m_ucCaDataNumWeCarry;
    unsigned short        CaId[META_CMMB_BLK_NUM];
    unsigned short        ServiceId[META_CMMB_BLK_NUM];
    unsigned char         EMM_BlockUnitType[META_CMMB_BLK_NUM];
    unsigned char         ECM_BlockUnitType[META_CMMB_BLK_NUM];
    unsigned char         ECM_TransmissionType[META_CMMB_BLK_NUM];

}META_CMMB_CaListStruct_T;


typedef struct
{

    META_CMMB_NitStruct_T        m_rNit;
    META_CMMB_MctStruct_T        m_rCSmct[2];  // [0] for Cmct, [1] for Smct
    META_CMMB_SctStruct_T        m_rCSsct[2];   // [0] for Csct, [1] for Ssct
    unsigned char                Eb_EbUpdateSeq;
    unsigned char                Eb_EbMsgNum;
    unsigned short               Eb_DataBlockLen;
    unsigned char                m_ucDataBlockLenWeCarray;
    unsigned char                         Eb_DataBlock[META_CMMB_DATA_BLK_NUM];
    unsigned char                     m_ucHasEsg;
    META_CMMB_EsgListStruct_T            m_rEsg;
    unsigned char                     m_ucHasCa;
    META_CMMB_CaListStruct_T             m_rCa;
}META_CMMB_CtrlInfoTable_T;


typedef struct
{
        unsigned char               m_u1NitUpdateSeq;
        unsigned char               m_u1CmctUpdateSeq;
        unsigned char               m_u1SmctUpdateSeq;
        unsigned char               m_u1CsctUpdateSeq;
        unsigned char               m_u1SsctUpdateSeq;
        unsigned char               m_u1EsgUpdateSeq;
        unsigned char                m_u1FreqPointId;
        unsigned char               m_u1NetLevel;
        unsigned short               m_u2NetId;
        unsigned char               m_u1HasCtrlTable;  // 0: no, 1: yes
        META_CMMB_CtrlInfoTable_T       m_rCtrlTableInfo;

}META_CMMB_FrameInfo_T;


typedef struct
{
     unsigned char             m_u1FrmNum;
      META_CMMB_FrameInfo_T       m_rFrmInfo[META_CMMB_FRAME_INFO_NUM];
}META_CMMB_AUTO_SCAN_CNF_T;
```

Description:

Request the CMMB module of the target to perform auto-scan operation.

Return Value:

Table 6-810 The return value of META_CMMB_AutoScan

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-811 The parameter of META_CMMB_AutoScan

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	out	The auto-scan results of the target's CMMB module

6.17.6 META_CMMB_AutoScanWithFreqRange

Definition:

META_RESULT __stdcall META_CMMB_AutoScanWithFreqRange(const unsigned int ms_timeout, META_CMMB_FREQ_RANGE_FOR_AUTO_SCAN_REQ_T *req, META_CMMB_AUTO_SCAN_CNF_T *cnf);

META_RESULT __stdcall META_CMMB_AutoScanWithFreqRange_r(const int meta_handle, const unsigned int ms_timeout, META_CMMB_FREQ_RANGE_FOR_AUTO_SCAN_REQ_T *req, META_CMMB_AUTO_SCAN_CNF_T *cnf);

```
typedef struct
{
  unsigned char  m_u1StartFreqPointId;  // the start channel
  unsigned char  m_u1EndFreqPointId;  // the stop channel

}META_CMMB_FREQ_RANGE_FOR_AUTO_SCAN_REQ_T;



#define          META_CMMB_BLK_NUM                     8
#define          META_CMMB_SERV_BLOCK_NUM              20
#define          META_CMMB_DATA_BLK_NUM                128
#define          META_CMMB_FRAME_INFO_NUM              4
```

```
typedef struct
{
     unsigned char          Nit_NitUpdateSeq;
    unsigned char          Nit_SysTime[5]   ;
    unsigned int           Nit_CountryCode ;
    unsigned char          Nit_Net_NetLevel;
    unsigned short         Nit_Net_NetId;
    unsigned char          Nit_NetNameLen  ;
    unsigned char          Nit_NetName[128];
     unsigned char          Nit_FreqBand_FreqPointId;
    unsigned char           Nit_FreqBand_BandWidth;
    unsigned int            Nit_FreqBand_CenterFreq;
    unsigned char          Nit_OtherFreqNum;
    unsigned char          m_ucOtherFreqNumWeCarry;


   unsigned char          Nit_OtherFreqBandList_FreqPointId[META_CMMB_BLK_NUM];
   unsigned char          Nit_OtherFreqBandList_BandWidth[META_CMMB_BLK_NUM];
   unsigned int           Nit_OtherFreqBandList_CenterFreq[META_CMMB_BLK_NUM];
   unsigned char          Nit_NeighborNetNum;
   unsigned char          m_ucNeightborNetWeCarray;
   unsigned char          Nit_NeighborNetList_NetLevel[META_CMMB_BLK_NUM];
   unsigned short          Nit_NeighborNetList_NetId[META_CMMB_BLK_NUM];
   unsigned char          Nit_NeighborNetList_FreqPointId[META_CMMB_BLK_NUM];
   unsigned char          Nit_NeighborNetList_BandWidth[META_CMMB_BLK_NUM];
   unsigned int          Nit_NeighborNetList_CenterFreq[META_CMMB_BLK_NUM];

}META_CMMB_NitStruct_T;
```

```
typedef struct
{
    unsigned char           MctUpdateSeq;
    unsigned char           FreqPointId;
    unsigned char           MfNum;
    unsigned char           m_ucMfNumWeCarray;
    unsigned char           Mf_MfId[META_CMMB_BLK_NUM];
    unsigned char           Mf_RsRate[META_CMMB_BLK_NUM];
    unsigned char           Mf_ByteInterleaveMode[META_CMMB_BLK_NUM];
    unsigned char           Mf_LdpcRate[META_CMMB_BLK_NUM];
    unsigned char           Mf_ModulationMode[META_CMMB_BLK_NUM];
    unsigned char           Mf_ScrambleMode[META_CMMB_BLK_NUM];
    unsigned char           Mf_TimeSlotNum[META_CMMB_BLK_NUM];
    unsigned char           m_ucMf_TimeSlotNumWeCarray[META_CMMB_BLK_NUM];
    unsigned char           Mf_TimeSlotId[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];
    unsigned char           Mf_SubMfNum[META_CMMB_BLK_NUM];
    unsigned char           m_ucMf_SubMfNumWeCarry[META_CMMB_BLK_NUM];
    unsigned char           Mf_SubMfId[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];
    unsigned short          Mf_serviceId[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];

}META_CMMB_MctStruct_T;

typedef struct
{
    unsigned char           SctUpdateSeq;
    unsigned short          ServiceNum;
    unsigned char           m_u1ServiceNumWeCarray;
    unsigned short          ServiceId[META_CMMB_SERV_BLOCK_NUM];
```

```
  unsigned char              FreqPointId[META_CMMB_SERV_BLOCK_NUM];

}META_CMMB_SctStruct_T;

typedef struct
{
  unsigned char            EsgUpdateSeq;
  unsigned char            NetLevel;
  unsigned short           NetId;
  unsigned char            LocalTimeOffset;
  unsigned char            CharSet;
  unsigned char            EsgServiceNum;
  unsigned char            m_ucEsgServiceNumWeCarry;
  unsigned char            EsgService_EsgServiceIndex[META_CMMB_BLK_NUM];
  unsigned short           EsgService_EsgServiceId[META_CMMB_BLK_NUM];
  unsigned char            EsgDataNum;
  unsigned char            m_ucEsgDataNumWeCarry;
  unsigned char            EsgData_EsgDataType[META_CMMB_BLK_NUM];
  unsigned char            EsgData_EsgDataBlockNum[META_CMMB_BLK_NUM];
  unsigned char            m_ucEsgData_EsgDataBlockNumWeCarry[META_CMMB_BLK_NUM];
  unsigned char            EsgDataBlock_EsgDataBlockId[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];
  unsigned char          EsgDataBlock_EsgDataBlockVersion[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];
  unsigned char            EsgDataBlock_EsgServiceIndex[META_CMMB_BLK_NUM][META_CMMB_BLK_NUM];

}META_CMMB_EsgListStruct_T;

typedef struct
{
    unsigned char           CaUpdateSeq;
    unsigned short          CaDataNum;
    unsigned char           m_ucCaDataNumWeCarry;
    unsigned short          CaId[META_CMMB_BLK_NUM];
    unsigned short          ServiceId[META_CMMB_BLK_NUM];
    unsigned char           EMM_BlockUnitType[META_CMMB_BLK_NUM];
    unsigned char           ECM_BlockUnitType[META_CMMB_BLK_NUM];
    unsigned char           ECM_TransmissionType[META_CMMB_BLK_NUM];

}META_CMMB_CaListStruct_T;

typedef struct
{

    META_CMMB_NitStruct_T        m_rNit;
    META_CMMB_MctStruct_T        m_rCSmct[2];  // [0] for Cmct, [1] for Smct
    META_CMMB_SctStruct_T       m_rCSsct[2];    // [0] for Csct, [1] for Ssct
    unsigned char                     Eb_EbUpdateSeq;
    unsigned char                     Eb_EbMsgNum;
    unsigned short                    Eb_DataBlockLen;
    unsigned char                     m_ucDataBlockLenWeCarray;
    unsigned char                            Eb_DataBlock[META_CMMB_DATA_BLK_NUM];
    unsigned char                         m_ucHasEsg;
    META_CMMB_EsgListStruct_T                m_rEsg;
    unsigned char                         m_ucHasCa;
    META_CMMB_CaListStruct_T                 m_rCa;

}META_CMMB_CtrlInfoTable_T;
```

```
typedef struct
{
        unsigned char               m_u1NitUpdateSeq;
        unsigned char               m_u1CmctUpdateSeq;
        unsigned char               m_u1SmctUpdateSeq;
        unsigned char               m_u1CsctUpdateSeq;
        unsigned char               m_u1SsctUpdateSeq;
        unsigned char               m_u1EsgUpdateSeq;
        unsigned char                m_u1FreqPointId;
        unsigned char               m_u1NetLevel;
        unsigned short               m_u2NetId;
        unsigned char               m_u1HasCtrlTable;  // 0: no, 1: yes
        META_CMMB_CtrlInfoTable_T       m_rCtrlTableInfo;

}META_CMMB_FrameInfo_T;


typedef struct
{
     unsigned char              m_u1FrmNum;
      META_CMMB_FrameInfo_T       m_rFrmInfo[META_CMMB_FRAME_INFO_NUM];
}META_CMMB_AUTO_SCAN_CNF_T;
```

Description:

Request the CMMB module of the target to perform auto-scan operation on the channels [m_u1StartFreqPointId, m_u1EndFreqPointId].

Return Value:

Table 6-812 The return value of META_CMMB_AutoScanWithFreqRange

Return value	Description
META_SUCCESS	Success

Return value	Description
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-813 The parameter of META_CMMB_AutoScanWithFreqRange

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	out	The auto-scan results of the target's CMMB module

6.17.7 META_CMMB_StopAutoScan

Definition:

META_RESULT __stdcall META_CMMB_StopAutoScan(const unsigned int ms_timeout);

META_RESULT __stdcall META_CMMB_StopAutoScan_r(const int meta_handle, const unsigned int ms_timeout);

Description:

Request the CMMB module to stop the auto-scan operation.

Return Value:

Table 6-814 The return value of META_CMMB_StopAutoScan

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-815 The parameter of META_CMMB_StopAutoScan

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond

6.17.8 META_CMMB_SetFreq

Definition:

META_RESULT __stdcall META_CMMB_SetFreq(const unsigned int ms_timeout, const CMMB_SET_FREQ_REQ_T *req, META_CMMB_SET_FREQ_CNF_T *cnf);

META_RESULT __stdcall META_CMMB_SetFreq_r(const int meta_handle, const unsigned int ms_timeout, const CMMB_SET_FREQ_REQ_T *req, META_CMMB_SET_FREQ_CNF_T *cnf);

```
typedef struct
{
     unsigned char m_u1FreqPointId;

}CMMB_SET_FREQ_REQ_T;

typedef struct
{
    unsigned char           m_u1FrmNum;
     META_CMMB_FrameInfo_T      m_rFrmInfo[META_CMMB_FRAME_INFO_NUM];
}META_CMMB_SET_FREQ_CNF_T;
```

Description:

Set the frequency of the CMMB module.

Return Value:

Table 6-816 The return value of META_CMMB_SetFreq

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-817 The parameter of META_CMMB_SetFreq

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The frequency which the user selects
cnf	OUT	The current CMMB frame information of the selected band and frequency

6.17.9 META_CMMB_SelServOnly

Definition:

META_RESULT __stdcall META_CMMB_SelServOnly(const unsigned int ms_timeout, const CMMB_SEL_SERV_REQ_ONLY_T *pSelServReq);

META_RESULT __stdcall META_CMMB_SelServOnly_r(const int meta_handle, const unsigned int ms_timeout, const CMMB_SEL_SERV_REQ_ONLY_T *pSelServReq);

```
typedef struct
{
        unsigned char   m_u1FrmId;
        unsigned short  m_u2ServId;

}CMMB_SEL_SERV_REQ_ONLY_T;
```

Description:

Request the CMMB module of the target to select a CMMB service to measure the signal strength.

Return Value:

Table 6-818 The return value of META_CMMB_SelServOnly

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-819 The parameter of META_CMMB_SelServOnly

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().

Parameter	IN/OUT	Description
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The CMMB service which the user wants to select

6.17.10 META_CMMB_PauseServ

Definition:

META_RESULT __stdcall META_CMMB_PauseServ(unsigned int ms_timeout, const META_CMMB_PAUSE_SERV_REQ_T *req);

META_RESULT __stdcall META_CMMB_PauseServ_r(const int meta_handle, unsigned int ms_timeout, const META_CMMB_PAUSE_SERV_REQ_T *req);

```
typedef struct
{
        unsigned char   m_u1FrmId;
        unsigned short  m_u2ServId;
}META_CMMB_PAUSE_SERV_REQ_T;
```

Description:

Request the CMMB module of the target to stop the CMMB service which the user selected before.

Return Value:

Table 6-820 The return value of META_CMMB_PauseServ

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-821 The parameter of META_CMMB_PauseServ

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
req	IN	The service that the user wants to recall

6.17.11 META_CMMB_GetSignalStrength

Definition:

META_RESULT __stdcall META_CMMB_GetSignalStrength(const unsigned int ms_timeout, META_CMMB_GET_SIGNAL_STRENGTH_CNF_T *cnf);

META_RESULT __stdcall META_CMMB_GetSignalStrength_r(const int meta_handle, const unsigned int ms_timeout, META_CMMB_GET_SIGNAL_STRENGTH_CNF_T *cnf);

```
typedef struct
{
    unsigned char   m_u1FreqPointId;
    char        m_i1Rssi;             // unit: -dBm 0~100, 0 is best  , -1 means no such kinds of value
    char        m_i1Snr;              // unit: dBm  0~100, 100 is best, -1 means no such kinds of value
    char        m_i1CurLdpcErrPercent;    // unit: %   0~100, 0 is best  , -1 means no such kinds of value

    int         m_i4TotalLdpcErrCnt;     // unit: -1 means no such kinds of value
    int         m_i4TotalLdpcCnt;        // unit: -1 means no such kinds of value

    int         m_i4CurRsErrorCnt;    // -1 means no such kinkds of value
    int         m_i4TotalRsErrorCnt;  // -1 means no such kinkds of value
    /* Added in W1112 */
    int         m_i4InBandPwr; // In band power (dBm)
    unsigned int    m_u4IsDemodLocked;
    unsigned char   m_u1ReceptionQuality;
    unsigned int    m_u4signal_strength_indication;
}META_CMMB_GET_SIGNAL_STRENGTH_CNF_T;
```

Description:

Query the signal strength information that the CMMB module measured after the user selected a service.

Return Value:

Table 6-822 The return value of META_CMMB_GetSignalStrength

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-823 The parameter of META_CMMB_GetSignalStrength

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
cnf	OUT	The signal strength information

6.18 Exported Functions for Customization on META Mode

From w0952, a module called FTC (FT Customer) will be running when target operates in Factory Mode.Customer can customize the source code of FTC (mcu\meta\ftc_main.c) and use the META DLL APIs depicted in this section to customize the target behavior by themselves.

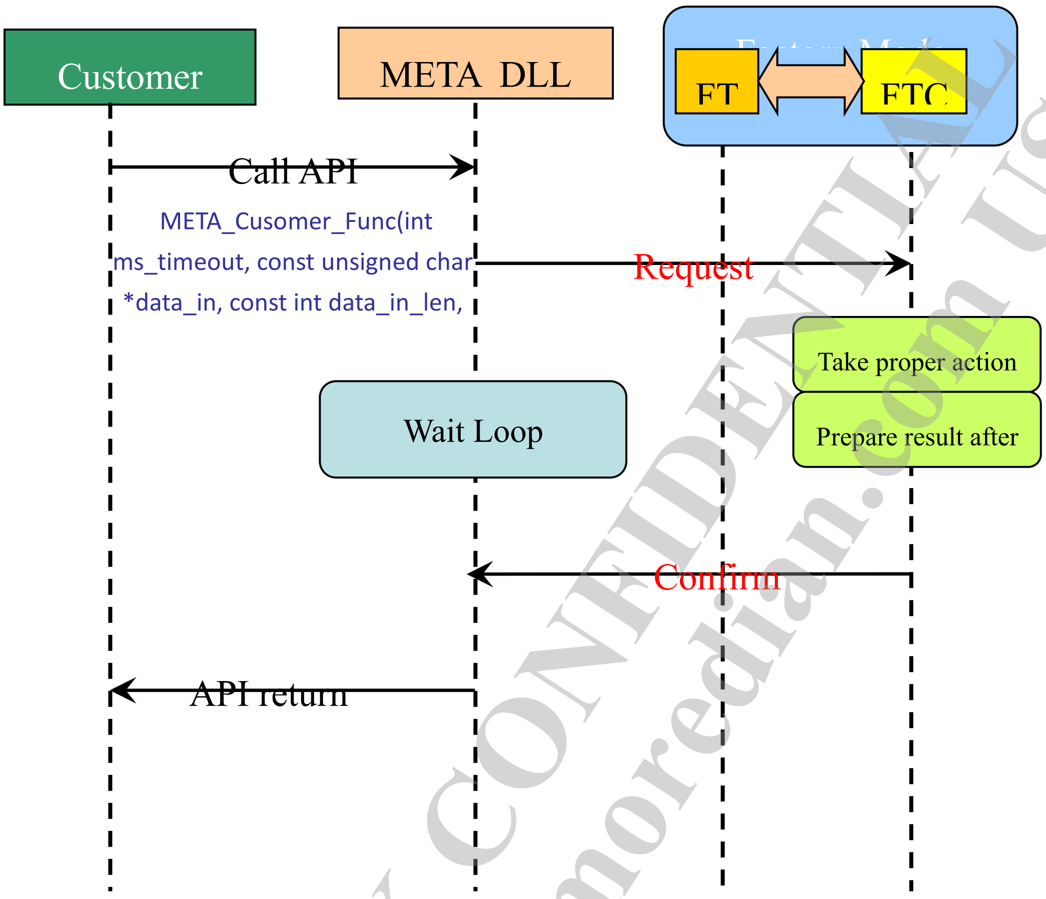

Figure 6-1 Exported Functions for Customization on META Mode

6.18.1 META_Customer_Func

Definition:

META_RESULT __stdcall META_Customer_Func(int ms_timeout, const unsigned char *data_in, const int data_in_len, unsigned char *data_out, int *data_out_len);

META_RESULT __stdcall META_Customer_Func_r(int meta_handle, int ms_timeout, const unsigned char *data_in, const int data_in_len, unsigned char *data_out, int *data_out_len);

Description:

Send a variable-length data (at most 2000 bytes) to FTC module, and receive a variable-length data (at most 2000 bytes) from FTC module. Default behavior of FTC task is to do echo operation (return the same data to PC-side tool), i.e. target will return the content of data_in directly.

Return Value:

Table 6-824 The return value of META_Customer_Func

Return value	Description
META_SUCCESS	Success
Other error code	For other error messages, please use META_GetErrorString to translate the meaning.

Parameter:

Table 6-825 The parameter of META_Customer_Func

Parameter	IN/OUT	Description
meta_handle	IN	Handling of META_DLL that returned from META_GetAvailableHandle().
ms_timeout	IN	Timeout value, unit = minisecond
data_in	IN	A data buffer will be send to target
data_in_len	IN	The length of data_in buffer
data_out	IN/OUT	A data buffer will be filled with the content returned from target, needs to be allocated by upper application in advance.
data_out_len	OUT	The length of data_out buffer returned from target.
cnf	OUT	The signal strength information

6.18.2 Sample code

6.18.2.1 Request of Customer-defined Protocols

data_in[0~3]: command type. (0x01: read IMEI, 0x02:Write IMEI, others: echo)

data_in[4~data_in_len]: user defined.

6.18.2.2 Confirm of Customer-defined Protocols

data_out[0~3]: command type (0x01: read IMEI, 0x02:Write IMEI, others: echo)

data_out[4~7]: status (0x00: OK, others: Error code)

data_out[8~data_out_len]: data replied by FTC task.