MRE Developer FAQ

Version:	1.0
Release Date:	2011-1-6

Table of Contents

1 Purpose 7

1.1 Scope 7
1.2 Terminology 7
1.3 References 7

2 Development Environment 8

2.1 Installation of MRE SDK 8
2.1.1 Which VS2008 edition supported 8
2.1.2 Why the icon on VS2008 toolbar as described in the file does not appear after the installation? 8
2.2 Debugging on Simulator 8
2.2.1 How to start application on Simulator 8
2.2.2 How can I check the debugging information on MRE App in the simulator? 9
2.2.3 How can I open the Internet connection from the simulator? 9
2.2.4 Which audio formats does the simulator support? 9
2.2.5 Which formats of audio recording does the simulator support? 9
2.3 Frequently Asked Questions 9
2.3.1 Build Errors 9
2.3.2 Cannot find the MRE App in the Simulator File System 9
2.3.3 How to use C++ compiler? 10
2.3.4 How can I add a new resolution? 10
2.3.5 Can I use the APP generated on MRE 1.0 tool chain directly on MRE 2.0 tool chain? Can I use RVCT3.1 to compile MRE 1.0 App? 10

3 Limitations on Development 11

3.1 Limitations on Development Language 11
3.2 Limitation on Memory Usage 11
3.3 Restrictions on Byte Alignment 11
3.4 Restrictions on Using Functions 11
3.5 Limitations on the Number of Source Files 12
3.6 Limitations on Variable Types 12
3.7 Position-independent Limitation 12
3.8 Limitations on the Size of Executable File 13

4 Program Framework 15

4.1 Framework Overview 15
4.2 Notes 15
4.2.1 Message Queue 15
4.2.2 Handling External Events 15
4.2.3 Processing Background Operation. 15
4.3 Frequently Asked Questions 16
4.3.1 Can I include custom events in an application? 16
4.3.2 Why does the program crash after calling vm_exit_app? 16
4.3.3 When an outbound call occurs during program execution, why does the system interface appear to be in a state of disorder? 16

4.3.4 Does MRE support automatic application launch during phone startup? 16
4.3.5 Does MRE support running applications in the background? 16
4.3.6 Why does the program that I've developed that runs in the background just terminate when I hit the End button? 16
4.3.7 Can the share module be verified on MoDIS? 17
4.3.8 Can the input value of vm_sm_load() carry the absolute path? 17

5 System Interface 18

5.1 Notes 18
5.1.1 Calculating the Amount of Available Memory 18
5.1.2 Memory Allocation and Release 18
5.1.3 Handling Timer 18
5.2 Frequently Asked Questions 19
5.2.1 Can MRE call the standard C functions malloc/realloc for memory allocation? 19
5.2.2 What are the possible reasons behind memory allocation failure? 19
5.2.3 What is the time returned by vm_get_time? 19
5.2.4 How do I generate random numbers? 19
5.2.5 Can I use floating-point numbers and floating-point operations in games? If so, what should I watch out for when using floating point numbers? 19
5.2.6 Why does the program I've developed stop running after the backlight is turned off? 19

6 Communications Interface 20

6.1 Frequently Asked Questions 20
6.1.1 How do you determine whether the phone is in Flight mode? 20
6.1.2 Can the vm_send_sms interface be used for billing/charging purposes? 20
6.2 If I would like to be charged by vm_send_sms, which API authorization should I use? 20
6.2.1 Why does sending text messages via vm_send_sms fail? 20
6.2.2 Does MRE support SMS interception? 20

7 Graphics Interface 21

7.1 Notes 21
7.1.1 Using Layers 21
7.1.2 Using Canvas 21
7.1.3 Using Outline Fonts 22
7.2 Frequently Asked Questions 22
7.2.1 Does MRE support better UI framework which is also standandised across all types of form factors? 22
7.2.2 What image formats are supported by MRE? 23
7.2.3 What are the possible reasons for failure to create a layer? 23
7.2.4 How do I draw an image on the screen? 23
7.2.5 How much memory space does a decoded image resource occupy? 23
7.2.6 Does MRE support animation? 23
7.2.7 How are Alpha effects implemented? 23
7.2.8 Does MRE support setting a picture directly as wallpaper on the handset? 23
7.2.9 How does one implement simultaneous image mirroring and rotation? 24

7.2.10 An image with a transparent background is drawn on a newly created canvas. Then the background turns completely black as the image is drawn on the layer. What is the problem? 24
7.2.11 How do I to determine the scope of the crop area when drawing? 24
7.2.12 I created a canvas that was bigger than the screen size and drew something on it. However, nothing happened when I attempted to draw the portion extended beyond the screen onto the layer? Why? 24
7.2.13 How are colors in MRE defined? 24
7.2.14 Why isn't the color defined with VM_INT_TO_565 (0xFF0000) red? 24
7.2.15 After an image has been opened as a file, decoded and drawn, it appears to be corrupted. Why? 24
7.2.16 Does vm_graphic_blt support layer-to-layer operation? 25
7.2.17 Upon return from an external event, the image drawn on the rapid layer fails to be displayed. What is the problem? 25
7.2.18 How does one obtain the height and width of a character? 25
7.2.19 Why the vm_graphic_set_font API is invalid? 25
7.2.20 Can I set up MRE App or change the direction of screen, for example from portrait to landscape? 25
7.2.21 Why does calling the vm_graphic_set_font interface produce no effect? 25

8 Character Set Interface 26

8.1 Notes 26
8.1.1 Character Conversion for Export Handsets 26
8.1.2 Limit on the Length of String Conversion 26
8.1.3 Using the vm_ucs2_string Interface 26
8.1.4 Using the vm_ucs2_to_gb2312 Interface 26
8.2 Frequently Asked Questions 26
8.2.1 How big should the second parameter in vm_gb2312_to_ucs2 be? 26

9 I/O Interface 27

9.1 Notes 27
9.1.1 Reading and Writing Files 27
9.1.2 Resource Loading 27
9.1.3 Character Input 27
9.1.4 Handling T-Flash Card Swapping 27
9.1.5 Log Output 27
9.1.6 Rules for Saving Files 28
9.2 Frequently Asked Questions 28
9.2.1 What are the possible causes for failure to load a resource? 28
9.2.2 Where are the created files saved to? 28
9.2.3 On a PC, how does MRE simulate a disk drive on the mobile phone emulator? 28
9.2.4 What are the causes of failure to create files? 28
9.2.5 Can MRE create a multi-level directory in one step? 29
9.2.6 How do I utilize the MRE log system? 29
9.2.7 Sometimes calling the log interface will cause the application to crash. What seems to be the problem? 29

9.2.8 How do I invoke the system input box? ... 29
9.2.9 After entering data with the MRE system input method and clicking “OK”, how do I access the string that has just been entered? ... 29
9.2.10 Is there a limit on the number of characters used for application name on the handset? ... 29
9.2.11 When the vm_file_rmdir function is called and files exist in the folder specified, are all files in that folder deleted, or no actions are taken? ... 30
9.2.12 Is the path delimiter character '/' supported? ... 30
9.2.13 When searching for a file using vm_find_first, how do I determine if the object found is a directory or a file? ... 30
9.2.14 When writing data to a file, the function call returns without writing the data in its entirety. What is the problem? ... 30
9.2.15 Is there a way to access disk capacity in MRE? ... 30
9.2.16 Can I call vm_ucs2_string multiple times within a single statement? ... 30
9.2.17 When I press the End key to exit, why does file save operation fail? ... 30

10 Network Interface ... 31
10.1 Notes ... 31
10.1.1 XML Parsing ... 31
10.2 Frequently Asked Questions ... 31
10.2.1 How many HTTP or SOCKET connections can be simultaneously created with MRE? ... 31
10.2.2 When transferring large files or data with MRE, is it necessary to have the files or data broken down into packets prior to sending? If so, is it done by MRE, or should it be done by the application itself? ... 31
10.2.3 Does the SOCKET interface in MRE support blocking? ... 31
10.2.4 Can a WAP page be displayed in MRE? ... 31
10.2.5 What do I do if the URL contains Chinese characters? ... 31
10.2.6 How does vm_asyn_http_req work? ... 31

11 Multimedia Interface ... 32
11.1 Frequently Asked Questions ... 32
11.1.1 What audio formats are supported by MRE? ... 32
11.1.2 Can I set the volume with MRE? ... 32
11.1.3 Which audio formats does MRE support? ... 32

12 Handset Debugging ... 33
12.1 Notes ... 33
12.1.1 Memory Requirements ... 33
12.1.2 Binding IMSI Number ... 33
12.2 Frequently Asked Questions ... 33
12.2.1 Why does the application cause a system halt and reboot when it is run on an actual handset? ... 33
12.2.2 When running the program on an actual phone, a “SIM card authentication failed” prompt appears. Why? ... 34

13 Others ... 35
13.1 How to integrate web browser and media player into MRE? ... 35

13.2 Does MRE support OMA DRM1.1? 35
13.3 Does MRE support triggers by push? 35
13.4 How can I update MRE App by OTA? 35
13.5 Does MRE support G-sensor? 35

1 Purpose

This document summarizes the potential problems that a developer may encounter throughout the process of developing MRE products, from development to testing to final production operation. The developer will be able to obtain help quickly when presented with problems.

1.1 Scope

This FAQ applies to the MRE1.0 platform.

1.2 Terminology

MRE	MAUI Runtime Environment.
MTK	MediaTek Inc., a corporation headquartered in Taiwan.
ARM	Advanced RISC Machine, a British company engaged in RISC microprocessor design and has a market share of over 90% in the mobile phone market.
ADS	ARM Development Suite, a software development kit provided by ARM Holdings, plc.
API	Application Program Interface.
SDK	Software Development Kit.
RAM	Random Access Memory.
ROM	Read Only Memory.
NVRAM	Non-Volatile Random Access Memory.
QCIF	Screen resolution, 176×220.
QVGA	Screen resolution, 240×320.
WQVGA	Screen resolution, 240×400.
HVGA	Screen resolution, 320×480.
OTA	Over the Air Technology.

1.3 References

2 Development Environment

2.1 Installation of MRE SDK

2.1.1 Which VS2008 edition supported

MRE SDK1.0 only support:

Microsoft Visual Studio 2008 Express Edition

Microsoft Visual Studio 2008 Professional Edition

MRE SDK1.0 does not support other edition. If you install the MRE SDK without Visual Studio 2008 Express Edition and Visual Studio 2008 Professional Edition, We will only install the general MRE SDK in your PC. You can start run the launcher to create a MRE App project (*.vcproj) . And you could make the MRE App binary file (*.vxp) with ADS1.2 ARM compiler.

2.1.2 Why the icon on VS2008 toolbar as described in the file does not appear after the installation?

The icon as described in the file only appears in the Microsoft Visual Studio 2008 Professional Edition.

2.2 Debugging on Simulator

2.2.1 How to start application on Simulator

Before you build MRE application in VS2008, please use MRE Launcher to open the MRE project.

Then you build your application in VS2008, MRE Launcher will copy the dll file to the file system of your selected model under MRE installation directory.

If you execute your application in VS2008 or run MoDIS.exe under the model directory, Simulator will be launched.

There are several models in MRE SDK release.

In the model of HVGA and resolution above, there is MRE Application Manager (naming Fun&Games). You can find your application in AM(Application Manager).

For other models which resolution below, there is no AM, you have to launch your application from Main Menu->File Manager.

2.2.2 How can I check the debugging information on MRE App in the simulator?

You can use the monitor tool to check the debugging information. Please refer to “MRE SDK IED User Guide.pdf.”

2.2.3 How can I open the Internet connection from the simulator?

MoDIS connects directly to the Internet by its Windows socket, so you do not need to particularly open the connection. However, MoDIS does not currently support proxy so is unable to support if your Windows requires proxy to connect to the Internet

2.2.4 Which audio formats does the simulator support?

MP3, MIDI and ACC.

2.2.5 Which formats of audio recording does the simulator support?

MRE simulator does not support audio recording.

2.3 Frequently Asked Questions

2.3.1 Build Errors

2.3.1.1 fatal error C1083: Cannot open include file: 'vmsys.h': Invalid argument

- Step1 : Close all Visual Studio 2008 Editons.
- Step2: Remove the ***CurrentSettings.vssettings*** and ***VCComponents.dat*** in the path:
  - C:\Documents and Settings\username\My Documents\Visual Studio 2008\Settings\CurrentSettings.vssettings
  - C:\Documents and Settings\username\Local Settings\Application Data\Microsoft\VisualStudio\9.0\VCComponents.dat
  - Username: Current log in user in the windows
- Step3: Please reopen the VS2008 Tool. VS2008 will reset all setting items.
- Step4: Please add the include files and library files for VS2008:
  - Tools → Option → Projects and Solutions →VC++ Directories → Include files
    - C:\Program Files\MRE SDK v1.0\include
  - Tools → Option → Projects and Solutions →VC++ Directories → Library files
    - C:\Program Files\MRE SDK v1.0\lib

2.3.2 Cannot find the MRE App in the Simulator File System

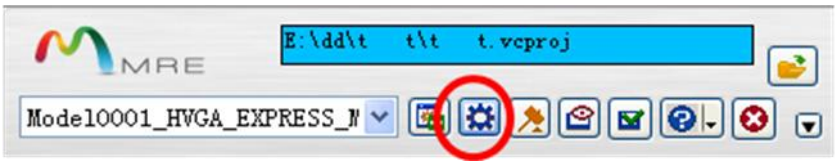
Reason: MRE SDK cannot related the MRE App project. Please run MRE Launcher tool. And open the MRE App (*.vcproj) by MRE Launcher tool. Then rebuild the MRE App project by Visual Studio 2008 Edition. It will build a Dll to simulator file system.

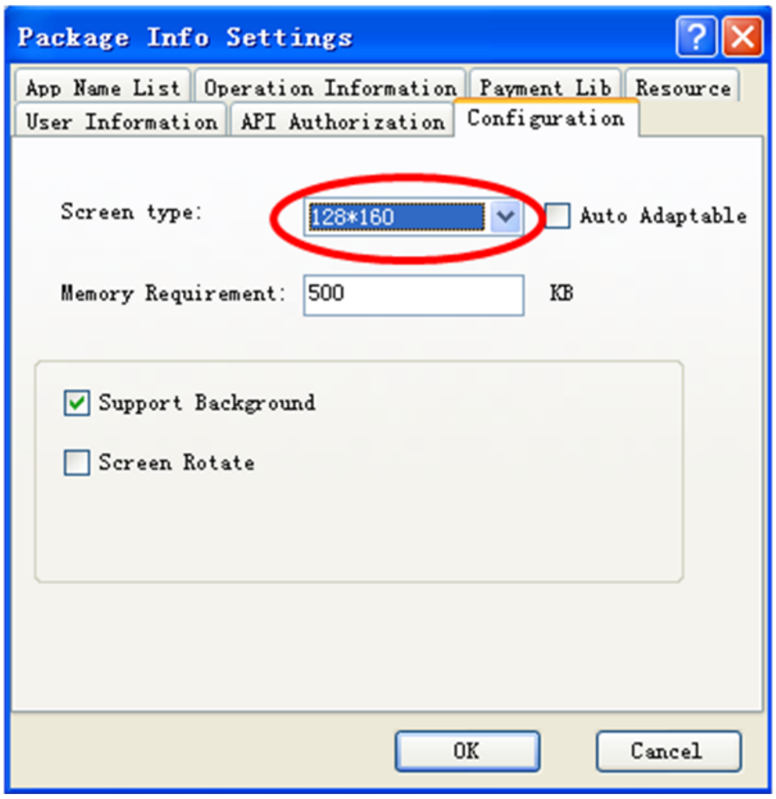
2.3.3 How to use C++ compiler?

If your source document is a .cpp document, MRE SDK will automatically select the C++ compiler for compilation. No other setup processes are needed.

2.3.4 How can I add a new resolution?

1. Open the "resolution.ini" file under the installation directory of MRE SDK (C:\Program Files\MRE SDK v1.0\Tools\resolution.ini
2. Add a new resolution. Format: width*height, e.g. 128*160




Package Info Settings

App Name List | Operation Information | Payment Lib | Resource | User Information | API Authorization | Configuration

Screen type: 128*160 ☐ Auto Adaptable

Memory Requirement: 500 KB

☑ Support Background

☐ Screen Rotate

OK Cancel

2.3.5 Can I use the APP generated on MRE 1.0 tool chain directly on MRE 2.0 tool chain? Can I use RVCT3.1 to compile MRE 1.0 App?

Yes, MRE SDK 2.0 is forward compatible. MRE 2.0 tool chain can open the App created by MRE SDK 1.0 tool chain. MRE 2.0 tool chain can also create the MRE 1.0 App by wizard. MRE 2.0 tool chain supports ADS1.2 and RVCT3.1 ARM compiler to build MRE 1.0 App.

3 Limitations on Development

3.1 Limitations on Development Language

Currently MRE supports ANSI C only.

If you want use C++ language in the MRE App. After generate helloworld project by wizard , please modify the helloworld.c to helloworld.cpp.

3.2 Limitation on Memory Usage

When started, MRE will request a block of memory, the size of which will vary according to the hardware platform being used. For example, on MT6225/MT6253, MRE can request up to 800K of RAM; on MT6235, MRE can request up to 1.5M of RAM; and on MT6238, MRE can request up to 3M of RAM; MRE uses a set of memory management algorithms to manage this block of memory space. The memory that an application requires at runtime can be requested from this block of memory. Note that MRE itself will occupy about 70K of RAM; in other words, the maximum amount of memory an application can use will be the size of MRE RAM minus this 70K. The memory that an application will need during runtime can be broken down into two parts. One is the memory occupied by the application code itself. This part of the data is loaded into memory at program startup, and will stay in memory until it terminates. The second part is the memory dynamically allocated at runtime. If the total size of the memory that is required to be allocated and the memory already occupied by the application code exceeds the maximum limit that the application can use, memory allocation will fail and program exceptions will occur.

With the handset's limitation, the stack space that the application can access cannot be more than 3K. As a result, do not attempt to define local variables that are too big within the body of a function. It is recommended that vm_malloc be used to allocate memory for larger local variables. Otherwise, when the level call to the function becomes too deep, stack overflow and subsequently blank screen and reboot of the phone may occur.

3.3 Restrictions on Byte Alignment

ARM processors are very strict about byte alignment requirements. If it is necessary to access the content of 4 bytes in one step, then the starting address of that content must be positioned on the 4-byte boundary. If it is necessary to access the content of 2 bytes in a single step, then the starting address of that content must be positioned on the 2-byte boundary, or else an exception will occur. Alignment problems occur mainly with incorrect pointer conversion, such as converting a byte pointer into a structure pointer. This type of pointer conversion should be avoided. The pointer returned by vm_malloc must be typecast. The pointer returned by vm_malloc, regardless of the size of its allocated space, must be aligned properly.

3.4 Restrictions on Using Functions

MRE supports only part of the commonly used standard C library functions. If the developer uses a function that is not supported by MRE, then the application will not start when deployed on an actual device. The following contains a number of functions that are not supported. These include file operation functions, assertion functions and memory operation functions:

File operation interfaces, e.g. fopen / fread / fwrite.

Abort and exit interfaces, e.g. abort / exit.

Character access interfaces, e.g. getchar / getc / putchar / putc.

Input display interfaces, e.g. scanf / printf.

Assertion and debug interfaces, e.g. ASSERT / TRACE.

Memory operation interfaces, e.g. malloc / realloc / free.

3.5 Limitations on the Number of Source Files

MRE supports compiling 500 source files simultaneously.

3.6 Limitations on Variable Types

When compiling and generating an executable file, the ARM compiler will carry out strict type checking. If variable types do not match, the compilation will fail, at which point explicit variable type cast will be required. Also note that the on the MTK platform, the char type is unsigned. Therefore VMUINT8 is recommended when byte variables are required.

3.7 Position-independent Limitation

At runtime an MRE program needs to be dynamically loaded into RAM. In other words, the addresses of the variables and functions can be dynamically assigned when they are loaded. This requires that the code be position-independent. If the code needs to determine the addresses at compile time, then it will fail to compile.

The following code exemplifies this error.

1.
   ```
   int a[10], b[10];
   struct c
   {
      int*    d;
      int*    e;
   };
   struct c f = {a, b};
   ```

2.
   ```
   void func1(void) {}
   typedef struct struct1
   {
      int a;
      void (*fun)(void);
   } struct1;
   struct1 array1[1] = {0, func1};
   ```

```
3. char *str = "test";

   char * list[] = {"zero", "one", "two"};
```

The solution is as follows:

```
1. int a[10], b[10];

   struct c

   {
      int*      d;
      int*      e;
   };
   struct c f;
   void init(void)

   {
      f.d = a;

      f.e = b;

   }
2. void func1(void) {}

   typedef struct struct1

   {
      int a;

      void (*fun)(void);

   } struct1;

   struct1 array1[1];

   void init(void)

   {
      array1[0].a = 0;

      array1[0].fun = func1;

   }
3. char str[] = "test";

   char list[][10] = {"zero", "one", "two"};
```

3.8 Limitations on the Size of Executable File

For the application itself, because the code segment of the application is already loaded into the memory when the program starts up, with the available memory space being constant, the greater the size of application code, the larger the memory space occupied on a long-term basis, and the less the available dynamically allocated memory. This can limit the operating efficiency of the application. For users, the greater the executable file, the longer the OTA download time. It is quite possible that users may cancel the download if they are impatient. For mobile phone manufacturers, because of the limited ROM size, they are more willing to incorporate smaller programs as their built-in applications. From the above considerations, it is recommended that the size of a generated executable file be kept under 400K.

4 Program Framework

4.1 Framework Overview

The main program entry function of an MRE program is vm_main. The application will need to register three types of message handlers within this function. These three types of events are: system messages, key event messages and pen event messages. When the program is running, user operations will constantly trigger these messages. The system will call the corresponding message handlers to process these messages. In addition, the application can also obtain the program's operating state periodically by setting a timer so that it can handle the different states accordingly.

4.2 Notes

4.2.1 Message Queue

An MRE program is message-driven. When a program is running, messages that need to be handled will enter the message queue. Only when the code that processes the current message is completed will the next message in the queue be retrieved and processed. Therefore if it takes too long to process a message, the other messages in the queue may not receive responses in a timely manner. This may not only affect user experience but may lead to exceptions as well. It is recommended that operations that require long process times be broken down into multiple steps, and a timer can be used to control these processes. For example, if it is necessary to load a large number of resources, a timer can be created so that a part of the resources may be loaded after each timeout of the timer.

4.2.2 Handling External Events

When an external event such as inbound call, SMS and USB operation occurs, the system will trigger a deactivation message (if the program does not support background operation, the message will be VM_MSG_INACTIVE; if the program supports background operation, the message will be VM_MSG_HIDE), and the application must delete all previously created layers, cease all rendering operations and preserve the program's current operating state after receiving this message. When the application has finished handling the external event, the system will trigger an activation message (if the program does not support background operation, the message will be VM_MSG_ACTIVE; if the program supports background operation, the message will be VM_MSG_PAINT). After receiving this message, the application needs to recreate the layers, redraw the screen interface and return to the state prior to the arrival of external event.

4.2.3 Processing Background Operation.

When developing an application that supports background operation, note the following:

1. Do not create layers, decode images and other image rendering operations within the message handling logic of VM_MSG_CREATE. Attempting to do so will fail.

2. When an interrupt event occurs or when the user presses the End button, the VM_MSG_HIDE message will be triggered, and upon return from handling the interrupt event,

the VM_MSG_PAINT message will be triggered. The VM_MSG_INACTIVE and VM_MSG_ACTIVE messages will not be triggered;

3. When generating the VXP file, the "support BG" option must be checked.

4.3 Frequently Asked Questions

4.3.1 Can I include custom events in an application?

Custom events can be implemented and simulated in an application via the following two approaches:

1. An application can implement the handling logic for custom events in message handlers and make calls to these message handlers elsewhere in the program. This is the synchronized approach.

2. An application can also utilize a timer to maintain a custom event queue. Messages will be retrieved from the queue periodically and then message handlers will be called. This is the asynchronous approach.

4.3.2 Why does the program crash after calling vm_exit_app?

This is because vm_exit_app is supposed to release all resources used by the application. After calling vm_exit_app, it will take a few moments before the program actually terminates, so if other operations are still accessing these resources, exceptions may occur. Therefore, we must ensure that no code is executed after vm_exit_app is called.

4.3.3 When an outbound call occurs during program execution, why does the system interface appear to be in a state of disorder?

During an outbound call, the MRE platform triggers the VM_MSG_INACTIVE or VM_MSG_HIDE message, after which control will be transferred to the system. At this time it is necessary for the application to delete the layer and cease any drawing operation in progress.

4.3.4 Does MRE support automatic application launch during phone startup?

No, currently MRE does not support automatic application launch during phone startup.

4.3.5 Does MRE support running applications in the background?

MRE1.0 already supports running applications in the background.

4.3.6 Why does the program that I've developed that runs in the background just terminate when I hit the End button?

Make sure that when generating the executable file, the "support BG" option is checked. Otherwise the program will not receive the VM_MSG_HIDE message when running on an actual device but instead receives the VM_MSG_QUIT message.

4.3.7 Can the share module be verified on MoDIS?

MoDIS does not support loading of share module. It can only be debugged on a real device.

4.3.8 Can the input value of vm_sm_load() carry the absolute path?

Sending only the name of the sm document, not the absolute path is recommended. The engine will search by the T card, system disk and ROM for the document to be loaded.

5 System Interface

5.1 Notes

5.1.1 Calculating the Amount of Available Memory

The size of the memory that a running application can allocate dynamically can be calculated with the following formula: Size of memory that can be dynamically allocated = Size of memory requested by MRE - Size of memory occupied by the application file. Here the total memory space requested by MRE varies by handset models, and possible values are 800K, 1.5M, 2M, 3M, etc. The application uses memory roughly equal to the sizes of RW + ROM segments.

For example

Suppose the total memory space requested by MRE is 800K. The Macro window will show the following information when VXP is being generated:

```
        Total RO  Size(Code + RO Data)                13616 (  13.30kB)
        Total RW  Size(RW Data + ZI Data)              1956 (   1.91kB)
        Total ROM Size(Code + RO Data + RW Data)      13620 (  13.30kB)
```

Then the size of memory that can be dynamically allocated will be about 800K-1.91K-13.30K=784.79K

When the program is running, you can use the vm_get_malloc_stat interface to access the status of dynamic memory allocation. Here, current represents the size of memory that has been dynamically allocated, avail_heap_size represents the amount of remaining memory available for dynamic allocation.

5.1.2 Memory Allocation and Release

The application must determine and act on the value returned from vm_malloc. If the allocation fails, appropriate error handling must be performed, and when vm_free is used to release memory, it is also necessary to ensure that the address of the memory to be released is correct. In addition, the memory requested by the application must be freed by the application itself. Otherwise exceptions will occur due to memory leak.

5.1.3 Handling Timer

MRE provides two types of timers. One offers high precision, but will stop operating when the handset's backlight is off. The function vm_create_timer can be used to create this type of timer; The other type of timer has low precision, but it will continue to operate with the handset's backlight turned off. The function vm_create_timer_ex can be used to create this type of timer. The minimum precision of the timer is 10ms. When a timer is created, the system will invoke the timer handling callback function periodically until the timer is deleted by the application. Since the next round of time-keeping operations will not commence before a timer handling function has completed its execution, and since the timer handling functions for other timers will not be executed before the timer handling function of the current time has been completed, timer callback involves certain time delay. The more timers, the more complicated the timer handling functions, and the longer the accumulated time delay. Also note that when the program exits, the application must delete all timers.

5.2 Frequently Asked Questions

5.2.1 Can MRE call the standard C functions malloc/realloc for memory allocation?

No. MRE has its own memory management interfaces. Please use vm_malloc/vm_realloc to request memory and vm_free to free memory.

5.2.2 What are the possible reasons behind memory allocation failure?

First of all, because of the limited amount of memory available on a handset, if the system cannot provide enough memory at the time of allocation, allocation failure will occur;

Secondly, if the application calls the vm_malloc and vm_free functions too frequently, it is possible that memory fragmentation will occur. Even though the total remaining free memory is sufficient, it may be too scattered and no adequate contiguous memory block is available for allocation. In this scenario, allocation failure is also a possibility;

Thirdly, if the memory out of bounds error occurs in the code during execution and modifies the control information used in memory allocation management and recovery, then allocation failure is also possible.

5.2.3 What is the time returned by vm_get_time?

It is the time from the handset's RTC.

5.2.4 How do I generate random numbers?

Please use rand, the standard C interface, and modify the random seed with srand.

5.2.5 Can I use floating-point numbers and floating-point operations in games? If so, what should I watch out for when using floating point numbers?

You can use floating point numbers in MRE. However, since floating point numbers are implemented by underlying software, their performance is not as good as hardware-based implementation. In addition, they are not suitable for large-scale high-density computing.

5.2.6 Why does the program I've developed stop running after the backlight is turned off?

A timer created with use vm_create_timer will stop running when the backlight is turned off. If your program depends on this type of timer, it will cease to operate when the backlight is off. If you need your application to continue running when the backlight is off, create timers with vm_create_timer_ex instead.

6 Communications Interface

6.1 Frequently Asked Questions

6.1.1 How do you determine whether the phone is in Flight mode?

You can use the vm_sim_card_count interface; a return value of 99 indicates that the handset is in Flight mode.

6.1.2 Can the vm_send_sms interface be used for billing/charging purposes?

Yes, but authorization is required.

6.2 If I would like to be charged by vm_send_sms, which API authorization should I use?

SMS (SP).

6.2.1 Why does sending text messages via vm_send_sms fail?

First of all, check to see if the SIM card is securely inserted and if the SIM card contains any balance;

Secondly, check to see if the input parameters are correct. The vm_ucs2_string interface may not be used twice at the same time for string conversion purposes;

Thirdly, if the message is sent to the SP, the appropriate permissions must first be granted.

6.2.2 Does MRE support SMS interception?

No. Currently MRE does not provide this function. The application can only obtain SMS events via registration with the SMS event notification function.

7 Graphics Interface

7.1 Notes

7.1.1 Using Layers

MRE1.0 is capable of creating multiple layers, including the base layer and the rapid layer. The sizes of these two layers may not exceed the size of the hardware screen. However, they do not occupy the memory space available to the application, and the speed of rendering is faster than those of other layers. It is recommended that frequently updated content be drawn on these two layers. Although other layers do occupy the memory space available to the application and their rendering speed is slower, given enough memory, the height and width of these layers can exceed the dimensions of the hardware screen. Note that it is necessary to create the base layer first before creating other layers, and when deleting layers, other layers must be deleted before the base layer. In addition, vm_graphic_create_layer can only be used to create the base layer and rapid layer, and vm_graphic_create_layer_ex can be used to create any layer. Suppose a layer is created via the canvas. When the canvas is freed by calling vm_graphic_release_canvas_ex, the corresponding layer will be deleted as well.

In a layer buffer, each pair of bytes represents a pixel and contains no other attribute information. As a result, after the application obtains the top address of the layer buffer, it can operate directly on the layer buffer as required.

7.1.2 Using Canvas

In addition to the base layer and rapid layer, other layers must be created via canvas. A canvas is a block of off-screen buffer and has the following structure:

9BYTE	1BYTE	2BYTE	n	n	
Flags field	No. of frames	Reserved	Data for one frame	Data for one frame	.

The structure of a frame of data on the canvas is as follows:

1BYTE	2BYTE	2BYTE	2BYTE	2BYTE	2BYTE
Flag	Left	Top	Width	Height	Delay

1BYTE	2BYTE	2BYTE	4BYTE	Offset
Trans color reserved	Trans color	Reserved	Offset	Image data

Field Description	Length	Description

Flag	1	An image logo
Left	2	Distance of the image from the left end of the
Top	2	Distance of the image from the right end of the
Width	2	Width of image
Height	2	Height of image
Delay	2	Delay of image
Tran color reserved	1	Reserved value of transparent color
Trans color	2	Color transparency value
Reserved	2	Reserved
Offset	4	Size of image data block
Image data	Offset	Image data

When creating a canvas, the canvas handle returned is actually the top address of the canvas buffer. If you use the canvas to create a layer, then the top address of the canvas buffer plus 32 bytes will be the top address of the layer buffer. Therefore, after the application has accessed the top address of the canvas buffer or the top address of the layer buffer, it can easily modify the properties of the canvas and image data.

Also note that, since canvases require access to the memory available to the application, they must be freed before the application terminates.

7.1.3 Using Outline Fonts

Currently many phones have integrated outline (vector) fonts. Between any two characters in a string rendered with an outline font, there is padding, and the padding may be different for different character pairs. The padding will change according to the different configurations of the vector fonts. The value returned from vm_graphic_measure_character or vm_graphic_get_character_width contains the width of the individual character without padding. Therefore if a vector font is used, calculating the length of a string by the aggregate lengths of the individual characters will produce unpredictable and inconsistent results under different handset models and versions. Therefore it is recommended that vm_graphic_get_string_width be used to obtain the width of the string in one step and vm_graphic_textout be used to draw the entire string.

7.2 Frequently Asked Questions

7.2.1 Does MRE support better UI framework which is also standandised across all types of form factors?

MRE support better UI framework after MRE 2.0， but not support 176x220.

7.2.2 What image formats are supported by MRE?

MRE1.0 supports these four image formats: GIF/JPEG/PNG/BMP.

7.2.3 What are the possible reasons for failure to create a layer?

There are several reasons why layer creation may fail, which are explained below:

1. An application that supports background operation attempts to create a layer within the message handling logic of VM_MSG_CREATE;
2. The first layer being created is not the base layer;
3. The base layer and the rapid layer already exist, and a call to vm_graphic_create_layer is attempted to create an additional layer;
4. The height and width passed into the function call are incorrect. The height and width of the base layer or the rapid layer may not exceed the screen size. The height and width of other layers may not exceed 65535;
5. Using VM_CREATE_CANVAS to create a layer, but the available memory is insufficient.

7.2.4 How do I draw an image on the screen?

Follow these steps to draw an image:

1. Use the vm_load_resource or the vm_file_read interface to load image data into memory;
2. Use the vm_graphic_load_image interface to decode the image onto a canvas;
3. Use the vm_graphic_blt interface to draw the image data on the canvas onto the layer or use the canvas to create a new layer directly;
4. Use the vm_graphic_flush_layer interface to refresh the screen.

7.2.5 How much memory space does a decoded image resource occupy?

The memory used by a decoded image resource is approximately equal to the image's width × height × 2 bytes. For example, an image of 100×100 pixels occupies about 100×100×2 ≈ 20K of memory space.

7.2.6 Does MRE support animation?

No. Animation effects must be implemented by the application.

7.2.7 How are Alpha effects implemented?

There are two approaches:

1. Use the vm_graphic_blt_ex interface;
2. Set the layer transparency.

7.2.8 Does MRE support setting a picture directly as wallpaper on the handset?

MRE does not provide support for such a function at this time.

7.2.9 How does one implement simultaneous image mirroring and rotation?

There are two methods available as follows:

1. Create a new canvas as an intermediate conversion buffer. First call the vm_graphic_mirror interface to draw the decoded image onto this new canvas. Then call the vm_graphic_rotate interface to draw the data located on the canvas data onto the layer;

2. Create a new layer, draw the decoded image on the layer. Then perform mirroring and rotation operations on the layer directly.

7.2.10 An image with a transparent background is drawn on a newly created canvas. Then the background turns completely black as the image is drawn on the layer. What is the problem?

At the time a canvas is created, the image data contained in the canvas is set to zero throughout. When an image with a transparent background is drawn on the canvas, the transparent portion of the image remains zero. Therefore the background will become black as image is being drawn on the layer. If you wish to keep the background of the image transparent, you may use the vm_graphic_canvas_set_trans_color interface to set the transparent color of the canvas to black.

7.2.11 How do I to determine the scope of the crop area when drawing?

The crop area is global. In other words, as long as it is not reset, the crop area will not change. In addition, the origin of the crop area's coordinates will always coincide with the origin of the target drawing area's coordinates.

7.2.12 I created a canvas that was bigger than the screen size and drew something on it. However, nothing happened when I attempted to draw the portion extended beyond the screen onto the layer? Why?

This is because the default size of the crop area is the screen size, and you did not set the crop area to the entire canvas before drawing on it, which resulted in the portion extended beyond the screen not being drawn on the canvas, and of course its content could not be drawn on the layer, either.

7.2.13 How are colors in MRE defined?

MRE uses the RGB565 color format.

7.2.14 Why isn't the color defined with VM_INT_TO_565 (0xFF0000) red?

The color component order of the INT format in MRE is BGR.

7.2.15 After an image has been opened as a file, decoded and drawn, it appears to be corrupted. Why?

This is because the file is not opened as a binary file, and this will result in incorrectly decoded data.

7.2.16 Does vm_graphic_blt support layer-to-layer operation?

vm_graphic_blt does not support the drawing from a layer to another layer. Instead you can use memcpy to achieve the same result.

7.2.17 Upon return from an external event, the image drawn on the rapid layer fails to be displayed. What is the problem?

This is because as the external event arrives, the order of deleting layers is in error. The correct order is to delete the rapid layer first, followed by the base layer. If you delete the base layer first, the operation will fail, in which case only the rapid layer will be deleted. Upon return from the external event, if an attempt is made to create the base layer, since the base layer already exists, the layer that will actually be created is the rapid layer. As MRE supports up to a maximum of two layers, further attempts at creating layers will fail, and so will subsequent draw operations.

7.2.18 How does one obtain the height and width of a character?

You can use the vm_graphic_measure_character interface to access the height and width of the character simultaneously; the input character parameters can be in ASCII or Chinese character format. Furthermore, use the vm_graphic_get_character_width interface to access the width of ASCII and Chinese characters and vm_graphic_get_character_height to obtain the height of ASCII characters.

7.2.19 Why the vm_graphic_set_font API is invalid?

The ROM space should be fully utilized. Therefore you often find there is only one Chinese font bank on phones. In this case, the port for setting up Chinese font size is invalid.

7.2.20 Can I set up MRE App or change the direction of screen, for example from portrait to landscape?

No, these functions are not currently supported. You can find the 320*240 simulator in MRE SDK.

7.2.21 Why does calling the vm_graphic_set_font interface produce no effect?

With the limited ROM size, most mobile phones are shipped with only one set of Chinese font. Setting font size with this function has no effect at all.

8 Character Set Interface

8.1 Notes

8.1.1 Character Conversion for Export Handsets

Since Chinese character fonts are not in general included in handsets shipped overseas, gb2312 encoding is therefore not supported in these models. As a result, interfaces such as vm_gb2312_to_ucs2 and vm_ucs2_string cannot be used for character conversion. Only the vm_ascii_to_ucs2 and vm_default_to_ucs2 interfaces can be used instead, otherwise an exception will be produced due to conversion error.

8.1.2 Limit on the Length of String Conversion

If the vm_gb2312_to_ucs2, vm_ucs2_to_gb2312, vm_ascii_to_ucs2 or vm_ucs2to_ascii interface is used, the length of the second parameter, i.e. the target string after conversion, may not exceed 4096 bytes.

If the vm_ucs2_string or vm_gb2312_string interface is used, it is necessary to keep the length of the source character string at 255 or fewer characters.

8.1.3 Using the vm_ucs2_string Interface

Since vm_ucs2_string uses a static variable to hold the converted target string, one must take care not to call this function twice or more in a single statement. The same rule applies to the vm_gb2312_string interface.

8.1.4 Using the vm_ucs2_to_gb2312 Interface

Note that the second parameter passed to the interface must be greater than 1, or memory out of bounds exception will occur on an actual device.

8.2 Frequently Asked Questions

8.2.1 How big should the second parameter in vm_gb2312_to_ucs2 be?

The second parameter is the size of the space used to store the converted string, and the unit is byte.

9 I/O Interface

9.1 Notes

9.1.1 Reading and Writing Files

When reading and writing a file, absolute pathname must be used. Opening multiple files is not recommended. A file that is no longer in use should be closed promptly, or it is possible that opening another file may fail due to the lack of available file handle resources. Furthermore, do not open a file within the message handling logic of VM_MSG_QUIT, or the operation will fail.

When reading or writing a file, proper error handling must be performed for any errors that may occur, or it will result in an exception.

9.1.2 Resource Loading

When generating a vxp file, resources that have been added to the Resource Manager will be written to the vxp file. When an application calls vm_load_resource to add a resource, it will read the corresponding resource from the vxp file into memory. Therefore, if file handle resources are insufficient while loading a resource, the operation will fail as well. Proper error handling must be performed to deal with resource loading failure, or it will result in an exception.

A maximum of 800 resource files may be added to the Resource Manager.

9.1.3 Character Input

After calling the input method interface, the VM_MSG_INACTIVE or VM_MSG_HIDE message will be triggered. When message handling is completed, the execution of the current function will continue. The control will then be transferred to the system and the system input method window will be opened. After the user has finished entering the data, the VM_MSG_ACTIVE or VM_MSG_PAINT message will be triggered, and after message handling has been completed, the user's input text will be returned to the application via callback.

9.1.4 Handling T-Flash Card Swapping

Some handsets support T-Flash card hot swapping. If a T-Flash card is unplugged while the program is running, the system will trigger the VM_MSG_QUIT message. At this point the application needs to release the resources previously used and then terminate by calling vm_exit_app. During program execution, if a T-Flash card is unplugged, the system will need to wait for the program to exit before triggering the VM_MSG_QUIT message. As a result, the application must take special care to protect I/O operations, including file read/write operations and resource loading.

9.1.5 Log Output

Each exported log may not exceed 256 bytes in length, or an exception will result.

9.1.6 Rules for Saving Files

If a running program needs to create a file, the following rules must be followed: Suppose XXX-YX-12345 represents a product code. Files related to this product must then be saved under the drv:\\mre\\Save\\XXX-YX-12345\\ directory. Here drv represents the accessed removable drive or the system drive letter. Note that the directory specified by Save may not exist at all, so it is necessary to first create the directory with the vm_file_mkdir function, and the XXX-YX-12345 directory should also be created by the application.

9.2 Frequently Asked Questions

9.2.1 What are the possible causes for failure to load a resource?

There are several reasons for resource loading to fail, as shown below:

1. The path specified in the MRE Resource Manager is incorrect;
2. The filename parameter passed to vm_load_resource contains invalid information;
3. Insufficient memory;
4. Too many open files resulting in no file handles available;
5. Loading resources within the message handling logic of VM_MSG_QUIT.

9.2.2 Where are the created files saved to?

For an application that is authorized to access the Global File System (GFS), absolute pathnames must be used for all file operations. MRE developers can utilize system drive and removable drive (T-Flash card). The drive letter must be specified. Since the system drive on a handset has limited space, developers are advised to use the removable drive as the preferred file saving destination.

If an application has not been authorized, MRE will create a subdirectory for the application under the HOME directory. All files created by the application will then be placed in this subdirectory and the application will not be able to access files and directories outside the subdirectory. MRE will generate a virtual system drive and removable drive.

As the system drive of the handset is used mainly for storing important data, some phones do not allow applications to write files to the system drive. MRE enables and disables the interception of applications' write operations to the system drive via compilation. When interception is enabled, all disk write operations to the system drive are redirected to the memory card. If a memory card does not exist, the write operation will fail.

9.2.3 On a PC, how does MRE simulate a disk drive on the mobile phone emulator?

The C drive on the PC simulates the phone's system drive, and the D drive simulates the removable drive.

9.2.4 What are the causes of failure to create files?

There are several reasons for resource loading to fail, as shown below:

1. Incorrect file path parameter;
2. Directory specified already exists;

3. A number of previously opened files that have not yet been closed, which results in no file handles available;
4. Attempting to create a file within the message handling logic of VM_MSG_QUIT.

9.2.5 Can MRE create a multi-level directory in one step?

No, MRE is unable to create a multi-level directory in a single step. The different levels of directories must be created one at a time.

9.2.6 How do I utilize the MRE log system?

Before starting the log system, you must first call the vm_log_init interface to initialize the system and set the log's output path and log level. Afterwards you can use functions such as vm_log_debug to output the log. Before exiting the program, make a call to vm_log_close to close the log system.

9.2.7 Sometimes calling the log interface will cause the application to crash. What seems to be the problem?

If you need to output a log entry that is longer than 256 bytes, write the data to a file instead.

9.2.8 How do I invoke the system input box?

MRE provides three input method interfaces. In particular, vm_input_text3 can be used to set the default character string, input method and the maximum number of input characters. After the call is made, the MRE platform will trigger the VM_MSG_INACTIVE or VM_MSG_HIDE message and then open the system input window. After the user has entered the data, the MRE platform will trigger the VM_MSG_ACTIVE or VM_MSG_PAINT message and then make a call to the callback function to pass the user's input to the application.

9.2.9 After entering data with the MRE system input method and clicking “OK”, how do I access the string that has just been entered?

You can set the callback function in a parameter of the input method interface. After the user has entered the data, the MRE platform will make a call to this function and pass the user's input to the application. The first parameter of the function is used to determine whether the user has clicked on the OK or Cancel button. If the value is TRUE, then the second parameter contains the user's input.

9.2.10 Is there a limit on the number of characters used for application name on the handset?

When creating a new project and generating icons, there is no limit on the length of the application name. However, there is a 40-byte limit on the handset, i.e. 20 Chinese characters or 40 ASCII characters.

9.2.11 When the vm_file_rmdir function is called and files exist in the folder specified, are all files in that folder deleted, or no actions are taken?

If the folder contains a file or another folder, calling this function will not delete the folder. If you wish to delete the folder, the contents in that folder must first be emptied.

9.2.12 Is the path delimiter character '/' supported?

No. Only “\\” can be used as the delimiter.

9.2.13 When searching for a file using vm_find_first, how do I determine if the object found is a directory or a file?

The vm_find_first interface cannot determine whether the object found is a directory or a file. You can use the vm_find_first_ext interface instead. If result's member variable attributes contain the value 0x10, then it is a directory. Otherwise it is a file.

9.2.14 When writing data to a file, the function call returns without writing the data in its entirety. What is the problem?

There are two ways to write data to a file, binary and ASCII character mode. Make sure you are using the appropriate mode.

9.2.15 Is there a way to access disk capacity in MRE?

You can call the vm_get_disk_free_space interface to obtain this information.

9.2.16 Can I call vm_ucs2_string multiple times within a single statement?

It is not possible to call vm_ucs2_string multiple times in a single statement. Use vm_gb2312_to_ucs2 or vm_ascii_to_ucs2 instead.

9.2.17 When I press the End key to exit, why does file save operation fail?

Do not call vm_file_open within the message handling logic of VM_MSG_QUIT to open a file, or the call will fail.

10Network Interface

10.1 Notes

10.1.1 XML Parsing

Currently only XML encoded in UTF-8 is supported.

10.2 Frequently Asked Questions

10.2.1 How many HTTP or SOCKET connections can be simultaneously created with MRE?

The maximum number of concurrent connections for HTTP and SOCKET is 3.

10.2.2 When transferring large files or data with MRE, is it necessary to have the files or data broken down into packets prior to sending? If so, is it done by MRE, or should it be done by the application itself?

When using SOCKET to transfer data, calling the vm_tcp_write interface will return the actual amount of data sent. If the quantity returned is smaller than the expected amount of data, it indicates that the data has not been sent. At this time the pointer must be moved to the end of the portion of the data that has been sent and transfer should be attempted once again.

10.2.3 Does the SOCKET interface in MRE support blocking?

On MRE, SOCKET is non-blocking synchronous. When the SOCKET is connected for read or write, the messages VM_TCP_EVT_CAN_READ / VM_TCP_EVT_CAN_WRITE will be returned via the callback function registered when the connection is established.

10.2.4 Can a WAP page be displayed in MRE?

An application can call the vm_open_wap_url interface to open the WAP browser on the mobile phone system. After calling the function, MRE will deactivate the current application and jump to the WAP browser of the underlying system. When access to a WAP page has been completed and the WAP browser is terminated, MRE will reactivate the current application.

10.2.5 What do I do if the URL contains Chinese characters?

You can use the vm_url_encode_gb2312 interface to encode the text.

10.2.6 How does vm_asyn_http_req work?

vm_asyn_http_req is an asynchronous implementation. Response message and HTTP status are returned via the callback function.

11Multimedia Interface

11.1Frequently Asked Questions

11.1.1 What audio formats are supported by MRE?

MRE supports these formats: MIDI/MP3/AAC/WAV/AMR.

11.1.2 Can I set the volume with MRE?

Volume can be set with the vm_set_volume interface.

11.1.3 Which audio formats does MRE support?

The audio formats MRE support depends on the current condition of MTK platform.

12 Handset Debugging

12.1 Notes

12.1.1 Memory Requirements

When generating a vxp file, the application must fill in the memory requirement based on its needs. The system will then allocate memory for the application based on this value. If the actual memory requirement of the application exceeds this value during execution, the allocation will fail due to insufficient memory. However, this value may not exceed the amount of memory that MRE is allowed to request. If the value exceeds this limit, application will simply fail at startup.

12.1.2 Binding IMSI Number

When generating a vxp file, the developer will be prompted to fill in IMEI/IMSI number. Only the IMSI on the SIM card is required. A vxp generated in this manner can only be executed on a handset containing that particular SIM card.

12.2 Frequently Asked Questions

12.2.1 Why does the application cause a system halt and reboot when it is run on an actual handset?

When wild pointers or null pointers, memory out of bounds, stack overflow, infinite loop or similar situations occur during program execution, the handset will experience critical exceptions such as system halt or reboot;

Developers will need to check the program code in accordance with the following requirements:

1. Locate throughout the code where memory is allocated. Check the return values of the function calls, and if memory allocation fails, perform the necessary error handling;

2. Locate throughout the code where memory is freed to ensure that the address of the released memory is legal;

3. Locate in the code where a memory allocation interface, such as vm_load_resource, is called and find out if the return value of the function call is checked. If the return value indicates an error, perform the necessary error handling;

4. Locate calls to memory manipulation functions throughout the code, e.g. memcpy and memset, to ensure that the memory addresses used in those function are legal and that the length of parameters are within bounds;

5. Locate calls to string manipulation functions throughout the code, e.g. strcpy, strlen, strcat and strstr, to ensure that the strings operated upon are legal, and that the length of parameters are within bounds;

6. Locate all arrays in the code and ensure that array out of bound problems do not occur during their use;

7. Check to see if the code contains large local variables. If a local variable occupies more than 3K of memory during program execution, a stack overflow exception will occur.

8. Locate all the loops within the code and make sure that each loop will eventually exit and there are no infinite loops;

12.2.2 When running the program on an actual phone, a “SIM card authentication failed” prompt appears. Why?

Check to see if a SIM card is inserted into the phone and if the IMSI number matches that indicated in the VXP when the file is generated.

13 Others

13.1 How to integrate web browser and media player into MRE?

You can enable the two Apps directly from MRE.

13.2 Does MRE support OMA DRM1.1?

No.

13.3 Does MRE support triggers by push?

No.

13.4 How can I update MRE App by OTA?

Down the new MRE App to replace the existing one in your phone or memory card. The download process is controlled by the customer end of App store.

13.5 Does MRE support G-sensor?

Yes. Please refer to document “vmsensor.h”.