mDolphin Programming Guide

Version 2.0

For mDolphin Version 2.0

Beijing Feynman Software Technology Co. Ltd.

March, 2008

Contents

1 Introduction

This handbook is a programming guide for mDolphin describes how to develop applications based on mDolphin. This ch introduce mDolphin and the organization of this guide.

1.1About mDolphin

mDolphin is developed by Beijing Feynman Software Tec mDolphin as an embedded browser based on MiniGUI graphic platf modular, scalable, full-featured browser and provid browsing experience.

mDolphin's modular and scalable architecture allows for custom configurations to meet specific device requirements. At present, mDolphin can b mobile phone (3G phone, WiFi phone), IPTV, information terminal and IP-based device. It can run on Linux/uClinux, eCos. Now, mDolphin has been successfully used in mobile communication device and video phone.

1.2Organization of This Guide

This guide contains the following information:

- Chapter 1 provides an introduction to this document.
- Chapter 2 contains a simple mDolphin program to help developer start quickly.
- Chapter 3 contains basic information about mDolphin and provides an overview of the mDolphin APIs.
- Chapter 4 describes the functions and structures of the mDolphin APIs, grouped by functionality.
- Chapter 5 provides sample code for the main activiti develop applications.

2 Beginning mDolphin Programming

This chapter will give a simple program to describe how to develop applications with mDolphin.

2.1A Simple mDolphin Program

The quickest approach to understand the basic pr mDolphin is to analyze structure of a simple program. Listing 2.1 s mDolphin program visiting a specified URL, which will be discussed in detail.

List 2.1 A simple mDolphin program

```
/*MiniGUI header file.*/
#include <minigui/common.h>
#include <minigui/minigui.h>
#include <minigui/gdi.h>
#include <minigui/window.h>
#include <minigui/control.h>

/*mDolphin header file.*/
#include "mdolphin.h"

#define IDC_MDOLPHIN 101

static const char * home_url = "http://www.minigui.com/index.php?id=product";

static int MDolphinProc (HWND hWnd, int message, WPARAM wParam, LPARAM lParam)
{
   switch (message)
   {
      case MSG_CREATE:
         {
```

```
            /*Create a mDolphin control class instance. */
            HWND mdolphin_hwnd = CreateWindow (MDOLPHIN_CTRL,
                  "",
                  WS_VISIBLE | WS_HSCROLL| WS_VSCROLL| WS_CHILD,
                  IDC_MDOLPHIN,
                  0, 0, 400, 300, hWnd, 0);
            /*Open a specified URL*/
            mdolphin_navigate(mdolphin_hwnd, NAV_GOTO, home_url, FALSE);
         }
         break;
      case MSG_DESTROY:
         DestroyAllControls (hWnd);
         return 0;
      case MSG_CLOSE:
         DestroyMainWindow (hWnd);
         PostQuitMessage (hWnd);
         return 0;
   }
   return DefaultMainWinProc (hWnd, message, wParam, lParam);
}

int MiniGUIMain (int args, const char* argv[])
{
   MSG Msg;
   MAINWINCREATE CreateInfo;

   /*register mDolphin control class*/
   RegisterMDolphinControl ();

#ifdef _MGRM_PROCESSES
   JoinLayer(NAME_DEF_LAYER , "mDolphin" , 0 , 0);
#endif

   CreateInfo.dwStyle = WS_VISIBLE|WS_CAPTION|WS_BORDER;
   CreateInfo.dwExStyle = WS_EX_NONE;
   CreateInfo.spCaption = "mDolphin";
   CreateInfo.hMenu = 0;
```

```
    CreateInfo.hCursor = GetSystemCursor(IDC_ARROW);
    CreateInfo.hIcon = 0;
    CreateInfo.MainWindowProc = MDolphinProc;
    CreateInfo.lx = 0;
    CreateInfo.ty = 0;
    CreateInfo.rx = 405;
    CreateInfo.by = 325;
    CreateInfo.iBkColor = COLOR_lightwhite;
    CreateInfo.dwAddData = 0;
    CreateInfo.hHosting = HWND_DESKTOP;

    HWND hMainWnd;
    hMainWnd = CreateMainWindow (&CreateInfo);

    if (hMainWnd == HWND_INVALID)
        return -1;

    ShowWindow(hMainWnd, SW_SHOWNORMAL);

    while (GetMessage(&Msg, hMainWnd)) {
        TranslateMessage(&Msg);
        DispatchMessage(&Msg);
    }

    /*Unregister mDolphin control class*/
    UnregisterMDolphinControl ();

    MainWindowThreadCleanup (hMainWnd);

    return 0;
}
```

As shown in Figure 2.1, the MiniGUI program creates an application window with size of 405◊325 pixels, and embeds mDolphin as a MiniGUI control with size of 400◊300 pixels to browse the Web page of MiniGUI. mDolphin, as a M control, can be easily used by developers.

Figure 2.1 A simple mDolphin program

2.2Header File

Besides MiniGUI header files, mDolphin header file included in the beginning of the simple program, named **mdolphin.h**, should be included for all mDolphin applications. The header file i structures and functions used by mDolphin.

```
#include “mdolphin.h”
```

2.3Registering Control Class

```
RegisterMDolphinControl ();
```

Each control of MiniGUI is an instance of a certain control class, and mDolphin as MiniGUI control is not exceptional. Before creating an n instance in MiniGUI program, you must register mDolphin control c

RegisterMDolphinControl provides this function.

2.4Creating Control Instance

```
HWND mdolphin_hwnd = CreateWindow (MDOLPHIN_CTRL,
                                   "",
                                   WS_VISIBLE | WS_HSCROLL| WS_VSCROLL| WS_CHILD,
                                   IDC_MDOLPHIN,
                                      0, 0, 400, 300, hWnd, 0);
```

By calling the **CreateWindow** function you can create an instance of predefined mDolphin control class. The control class name is *MDOLPHIN_CTRL*. You can specify the control caption, the control style, the control identifier, the initial position and size of the control. This function also specifies the parent window of the mDolphin control window.

2.5Opening a Specified URL

```
mdolphin_navigate(mdolphin_hwnd, NAV_GOTO, home_url, FALSE);
```

After the mDolphin control instance is created, the application can works by calling browsing functions of mDolphin. A lot of functions have been defined in mDolphin. For more information about these functions, please refer to chapter 4. When **mdolphin_navigate** function with *NAV_GOTO* parameter is called, mDolphin will open a URL specified by the third parameter.

2.6Unregistering Control Class

```
UnregisterMDolphinControl ();
```

If your application will not use the mDolphin control class any more, you should use **UnregisterMDolphinControl** function to unregister the mDolphin control class.

3 mDolphin Overview

3.1mDolphin Control

mDolphin is a MiniGUI control which provides a lot of browsing func
including going backward, going forward, opening a specified URL, reload
stopping, zooming, setting rendering mode, etc
developers can develop their applications based on m
quickly.

There is a predefined mDolphin control class in mDolphi
element for mDolphin. This chapter will describe mDolphin control in detail.

Table 3.1mDolphin Control

Control Class	Class Name
mdolphin	MDOLPHIN_CTRL

To develop application with mDolphin control, you should first register a control class with function **RegisterMDolphinControl**. It returns TRUE on success, otherwise returns FALSE.

```
BOOL RegisterMDolphinControl (void)
```

Secondly you should create an instance of mDolphin control class with function **CreateWindow**.

```
HWND GUIAPI CreateWindow(const char* spClassName, const char* spCaption, DWORD dwStyle,
int id, int x, int y, int w, int h, HWND hParentWnd, DWORD dwAddData)
```

SpClassName is the control class name, namely *MDOLPHIN_CTRL*. ***SpCaption,***

dwStyle, ***dwExStyle***, ***caption*** are mDolphin window caption, respectively. ***x***, ***y***, ***w*** and ***h*** are the initial position and size of the mDolphin window. ***hWndParent*** is the parent window of the mDolphin window. ***dwAddData*** is a undesired parameter.

When exiting the application, you should unregister function **UnregisterMDolphinControl**.

```
void UnregisterMDolphinControl (void)
```

3.2mDolphin API Overview

mDolphin enables users to browse the World Wide Web. mDolphin is a browser that you embed into an application to enable users to view Web content from within that application. The host application can control the w displays the Web content, dialogs, soft keys, and so on.

mDolphin provides a lot of browsing functions. By calling these functions, users can use mDolphin in the host program. Table 3.2 lists and des functions. For more information about mDolphin Plug-in APIs, please ref *mDolphin Plug-in Programming Guide*.

Table 3.2mDolphin API Overview

Functionality	**Function Name**	**Called by**	**Description**
Customizing Dialogs	CB_MESSAGE_BOX	mDolphin	Display a message to the user until the user presses OK.
	CB_CONFIRM_BOX	mDolphin	Display a confirmation message to the user.
	CB_PROMPT_BOX	mDolphin	Ask the user a question while the user is inputting data.
	CB_CHOOSEFILE_BOX	mDolphin	Browse through your file system and selects a file.

Customizing Status Information	CB_SET_TITLE	mDolphin	Set the browsing title of the Web page for the window.
	CB_SET_LOCATION	mDolphin	Set the visiting URL for the address field.
	CB_SET_STATUS	mDolphin	Set status messages for the status bar.
	CB_SET_LOADING_STATUS	mDolphin	Set loading status.
	CB_SET_HISTORY_STATUS	mDolphin	Set history status.
	CB_SET_IME_STATUS	mDolphin	Set IME window status.
	CB_ERROR	mDolphin	Notify the user of an error from the HTTP server during a download.
Customizing Popup Menu	CB_CREATE_POPUP_MENU	mDolphin	Create a context menu.
	mdolphin_perform_popup_menu_event	Host application	Perform the popup menu event.
Customizing Popup Window	CB_OPEN_WINDOW	mDolphin	Open a new window.
	CB_CLOSE_WINDOW	mDolphin	Close the specified window.
Setting Rendering Mode	mdolphin_set_rendering_mode	Host application	Set the specified rendering mode on the window.
	mdolphin_get_rendering_mode	Host application	Get the specified rendering mode on the window.
Customizing Navigation	mdolphin_navigate	Host application	Navigate the browser.
Setting Text Size	mdolphin_set_text_size_multiplier	Host application	Set the text size multiplier of the browser.
	mdolphin_get_text_size_multiplier	Host application	Get the text size multiplier of the browser.
Setting Text Encoding	mdolphin_set_text_encoding_name	Host application	Set the text encoding name of the browser.
	mdolphin_get_text_encoding_name	Host application	Get the text encoding name of the browser.
Setting Preferences	mdolphin_import_setup	Host application	Update the current preferences of the browser.
	mdolphin_fetch_setup	Host application	Get the current preferences of the browser.

Customizing Text Search	mdolphin_search_for	Host application	Search a document view for a string and highlights the string if it is found.
	mdolphin_mark_all_matches_for_text	Host application	Mark all matching strings in a document view and highlights or un-highlights the strings.
	mdolphin_unmark_all_text_matches	Host application	Unmark all matching text in a document view.
Setting In View Source Mode	mdolphin_set_in_view_source_mode	Host application	Place a view into a special source-viewing mode.
	mdolphin_in_view_source_mode	Host application	Whether or not the view is in source-view mode for HTML.
Customizing Visited URL	CB_URL_IS_VISITED	mDolphin	Verify whether the URL is visited.
Customizing Tooltip	CB_SET_TOOLTIP	mDolphin	Used to show a tool tip.
Customizing User Agent	CB_CUSTOM_USERAGENT	mDolphin	Used to set User-Agent according to URL.
Setting Proxy	mdolphin_set_proxy	Host application	Set the proxy when you use proxy to connect the network.
	mdolphin_get_proxy	Host application	Get the proxy information.
Setting SSL	CB_PROVIDE_CLIENT_CERT	mDolphin	During a handshake a server may request a certificate from the client. This callback function is used to set a certificate. The certificate/private key combination must be set using the x509 and pkey arguments and TRUE must be returned. The certificate will be installed into SSL.
	CB_VERIFY_SERVER_CERT	mDolphin	Decide whether trusting the current certificate.
	mdolphin_set_caPath	Host application	Specify a certificate directory holding alternate certificates to verify with.

Setting HTTP Auth	CB_PROVIDE_AUTH	mDolphin	Set the user name and the password for HTTP authentication.
Customizing Protocol Extension	CB_SCHEME_HANDLER	mDolphin	Check the scheme of a URL to determine whether or not mDolphin supports it. If mDolphin does not support it, then the host application should pass the content to this callback function that does support that scheme.
	mdolphin_register_scheme_handler	Host application	Register network protocol scheme handler.
	mdolphin_unregister_scheme_handler	Host application	Un-register network protocol scheme handler.
Customizing Downloading	CB_SAVE_FILE_DATA	mDolphin	Deal with data stream. When MIME type is unsupported or content net to be downloaded, users can decide whether saving data to a local file.
Setting Callback Functions	mdolphin_set_callback_info	Host application	Set the callback functions of the browser. In order to meet particular requirements or customization, users need to define some callback functions.
JavaScript Native Binding	mdolphin_define_jsnativeclass	Host application	Define a JavaScript native binding class.
	mdolphin_lookup_jsnativeclass	Host application	Search a JavaScript native binding class with the class name.
	mdolphin_undefine_jsnativeclass	Host application	Undefined a JavaScript native binding class.
	mdolphin_define_jsnativefunction	Host application	Define a JavaScript native binding function.
	mdolphin_undefine_jsnativefunction	Host application	Undefined a JavaScript native binding function.

More details about these functions of mDolphin are respectively described in chapter 4.

4 mDolphin API Reference Tables

This chapter will describe functions of mDolphin in detail.

4.1Customizing Dialogs

4.1.1CB_MESSAGE_BOX

Table 4.1**CB_MESSAGE_BOX** callback function

Syntax	void (*CB_MESSAGE_BOX)(HWND hWnd, const char *pszText, const char* pszCaption)	
Parameters	*hWnd*	The handle of mDolphin control, which will open a message box.
	pszText	The message text with UTF-8 character encoding, which will be displayed in the message box.
	pszCaption	The caption of the message box.
Returns	None	
Description	Used for JavaScript alert() method. This function displays a message box within only one button whose title is "OK".	

4.1.2CB_CONFIRM_BOX

Table 4.2**CB_CONFIRM_BOX** callback function

Syntax	BOOL (*CB_CONFIRM_BOX)(HWND hWnd, const char *pszText, const char* pszCaption)	
Parameters	*hWnd*	The handle of mDolphin control, which will open a confirm box.
	pszText	The message text with UTF-8 character encoding, which will be displayed in the confirm box.
	pszCaption	The caption of the confirm box.
Returns	TRUE if you select "OK", otherwise FALSE.	
Description	Used for JavaScript confirm () method. This function creates a message box within two buttons whose titles are "OK" and "Cancel" respectively.	

4.1.3CB_PROMP_BOX

Table 4.3**CB_PROMP_BOX** callback function

Syntax	char *(*CB_PROMPT_BOX)(HWND hWnd, const char *pszText, const char* pszDefault, const char* pszCaption)	
Parameters	*hWnd*	The handle of mDolphin control, which will display a dialog box that prompts the user for input.
	pszText	The message with UTF-8 character encoding, which will be displayed in the prompt box.
	pszDefault	The default input text with UTF-8 character encoding.
	pszCaption	The caption of the prompt box.
Returns	The string you commit, otherwise NULL.	
Description	Used for JavaScript prompt() method.	

4.1.4CB_CHOOSEFILE_BOX

Table 4.4**CB_CHOOSEFILE_BOX** callback function

Syntax	BOOL (*CB_CHOOSEFILE_BOX) (HWND hWnd, char *FileName, size_t BufferSize, BOOL IsSave)	
Parameters	*hWnd*	The handle of mDolphin Control, which will open a file choosing dialog box.
	FileName	The pointer to the buffer receiving the full path name of the file selected by user.
	BufferSize	The maximum length of the full path name of the file.
	IsSave	If IsSave is FALSE, an open file dialog box is created. Otherwise, a save file dialog box is opened.
Returns	TRUE if users choose a file and click OK button, otherwise FALSE.	
Description	Used to create an open file dialog box or a save file dialog box.	

4.2Customizing Status Information

4.2.1CB_SET_IME_STATUS

Table 4.5**CB_SET_IME_STATUS** callback function

Syntax	void (*CB_SET_IME_STATUS) (BOOL bstate)	
Parameters	*bstate*	The status of IME window. If bstate is TRUE, display the IME window. Otherwise, hide the IME window.
Returns	None	
Description	Used to set IME window status.	

4.2.2CB_SET_LOADING_STATUS

Table 4.6**CB_SET_LOADING_STATUS** callback function

Syntax	void (*CB_SET_LOADING_STATUS) (HWND hWnd, BOOL bIsLoading, unsigned int progress)	
Parameters	*hWnd*	The handle of mDolphin control.
	bIsLoading	The loading status. If there are any pending loads, bIsLoading is TRUE. When loading is finished, bIsLoading is FALSE.
	progress	An estimate of the percent complete for a document load. This value will range from 0 to 100.
Returns	None	
Description	Used to set loading status.	

4.2.3CB_SET_LOCATION

Table 4.7**CB_SET_LOCATION** callback function

Syntax	void (*CB_SET_LOCATION) (HWND hWnd, const char * plocText)	
Parameters	*hWnd*	The handle of mDolphin control.
	plocText	The visiting URL with UTF-8 character encoding.
Returns	None	
Descriptio	Used to set the visiting URL for the address field.	

n	

4.2.4CB_SET_STATUS

Table 4.8**CB_SET_STATUS** callback function

Syntax	void (*CB_SET_STATUS)(HWND hWnd, const char *status_msg)	
Parameters	*hWnd*	The handle of mDolphin control.
	status_msg	The status message with UTF-8 character encoding.
Returns	None	
Description	Used to set status messages for the status bar.	

4.2.5CB_SET_TITLE

Table 4.9**CB_SET_TITLE** callback function

Syntax	void (*CB_SET_TITLE) (HWND hWnd, const char *pText)	
Parameters	*hWnd*	The handle of mDolphin control.
	pText	The title of the browsing Web page with UTF-8 character encoding.
Returns	None	
Description	Used to set the browsing title of the Web page for the window.	

4.2.6CB_SET_HISTORY_STATUS

Table 4.10**CB_SET_HISTORY_STATUS** callback function

Syntax	void (*CB_SET_HISTORY_STATUS) (HWND hWnd, unsigned int bListCount, unsigned int fListCount, unsigned int capacity)	
Parameters	*hWnd*	The handle of mDolphin control.
	bListCount	The available backward list count.
	fListCount	The available forward list count.
	capacity	The capacity of history list.
Returns	None	
Descriptio	Used to set history status.	

n	

4.2.7CB_ERROR

Table 4.11**CB_ERROR** callback function

Syntax	void (*CB_ERROR) (HWND hWnd, int errCode, const char *url)	
Parameters	*hWnd*	The handle of mDolphin control.
	errCode	Error code for the error that occurred.
	url	The failed URL.
Returns	None	
Description	Notify the user of an error encountered during a download. Some examples are: unrecognized URL, and DNS not found. For more information about error code of mDolphin, please refer to Appendix B Error Code.	

4.3Customizing Popup Menu

4.3.1CB_CREATE_POPUP_MENU

Table 4.12**CB_CREATE_POPUP_MENU** callback function

Syntax	BOOL (*CB_CREATE_POPUP_MENU) (HWND hWnd, const POPUP_MENU_DATA *data)	
Parameters	*hWnd*	The handle of mDolphin Control.
	data	The structure of context menu information.
Returns	TRUE if users create context menu successfully, otherwise FALSE.	
Description	Used to create a context menu.	

4.3.2mdolphin_perform_popup_menu_event

Table 4.13**mdolphin_perform_popup_menu_event** function

Syntax	BOOL mdolphin_perform_popup_menu_event(HWND hwnd, POPUP_MENU_EVENT_TYPE event)	
Parameters	*hWnd*	The handle of mDolphin Control.
	event	The clicked popup menu item event.
Returns	TRUE on success, FALSE on error.	

Description	Used to perform the popup menu event.

4.4Customizing Popup Window

4.4.1CB_OPEN_WINDOW

Table 4.14**CB_OPEN_WINDOW** callback function

Syntax	HWND (*CB_OPEN_WINDOW) (const char *url, DWORD styles, int x, int y, int width, int height)		
Parameters	*url*	The URL visited by this window.	
	styles	The window styles:	
		CWS_TOOLBAR	Tell the window have tool bar.
		CWS_LOCATION	Tell the window have location bar.
		CWS_MENUBAR	Tell the window have menu bar.
		CWS_STATUS	Tell the window have status bar.
		CWS_SCROLLBARS	Tell the window have scroll bar.
		CWS_RESIZABLE	Tell the window whether it can be resized.
		CWS_MODAL	Tell the window it is a modal dialog.
	x	The x position which window on screen.	
	y	The y position which window on screen.	
	width	The width of this window, which can be -1, means the width is normal.	
	height	The height of this window, which can be -1, means the height is normal.	
Returns	The new handle of mDolphin Control.		
Description	Used to open a new window.		

4.4.2CB_CLOSE_WINDOW

Table 4.15**CB_CLOSE_WINDOW** callback function

Syntax	void (*CB_CLOSE_WINDOW) (HWND hWnd)
Parameters	*hWnd* The handle of mDolphin Control.

Returns	None
Description	Used to close the specified window.

4.5Setting Rendering Mode

4.5.1mdolphin_set_rendering_mode

Table 4.16**mdolphin_set_rendering_mode** function

Syntax	BOOL mdolphin_set_rendering_mode(HWND hwnd, RENDERING_MODE mode, int limit)	
Parameters	*hWnd*	The handle of mDolphin Control.
	mode	The mode to set, can be one of the following values: MD_SCREENVIEW_MODE — Screen mode (True Web Layout) will display Web pages in their original layout. MD_SMALLVIEW_MODE — Small view mode (Smart-Fit Rendering) will render everything in one narrow column, eliminating the need for horizontal scrolling. This rendering mode is usually used on mobile phones. MD_VIRTUALVIEW_MODE — Virtual view mode (Page Overview) will display a miniature of the entire Web page, which enables the user to see his/her current position on the page.
	limit	The max width of the view attached to MD_SMALLVIEW_MODE, the max width of the view attached to MD_VIRTUALVIEW_MODE.
Returns	TRUE on success, FALSE on error.	
Description	Used to set the current rendering mode.	

4.5.2mdolphin_get_rendering_mode

Table 4.17**mdolphin_get_rendering_mode** function

Syntax	BOOL mdolphin_get_rendering_mode(HWND hwnd, RENDERING_MODE *mode)	
Parameters	*hWnd*	The handle of mDolphin Control.
	mode	Pointer to a buffer receiving the current rendering mode.
Returns	TRUE on success, FALSE on error.	
Description	Used to get the current rendering mode.	

4.6Customizing Navigation

4.6.1mdolphin_navigate

Table 4.18**mdolphin_navigate** function

Syntax	BOOL mdolphin_navigate(HWND hwnd, NAV_OPERATION op, const char* url, BOOL lockHistory)		
Parameters	*hwnd*	The handle of mDolphin Control.	
	op	The navigation operation, can be one of the following values:	
		NAV_BACKWARD	Go backward.
		NAV_FORWARD	Go forward.
		NAV_GOTO	Open a specified URL.
		NAV_RELOAD	Reload the current web.
		NAV_STOP	Stop the current download.
	url	The specified URL attached to NAV_GOTO operation.	
	lockHistory	Whether to lock history or not, which attached to NAV_GOTO operation.	
Returns	TRUE on success, FALSE on error.		
Description	Navigate the browser.		

4.7Setting Text Size

4.7.1mdolphin_set_text_size_multiplier

Table 4.19**mdolphin_set_text_size_multiplier** function

Syntax	BOOL mdolphin_set_text_size_multiplier(HWND hwnd, int factor)
Parameters	*hWnd* The handle of mDolphin Control.
	factor The specified text size multiplier. Factor expresses int percent as an integer (eg 100 means 100 percent).
Returns	TRUE on success, FALSE on error.
Description	Used to set the current text size multiplier of the browser.

4.7.2mdolphin_get_text_size_multiplier

Table 4.20**mdolphin_get_text_size_multiplier** function

Syntax	BOOL mdolphin_get_text_size_multiplier(HWND hwnd, int *factor)
Parameters	*hwnd* The handle of mDolphin Control.
	factor Pointer to a buffer receiving the text size multiplier. Factor expresses int percent as an integer (eg 100 means 100 percent).
Returns	TRUE on success, FALSE on error.
Description	Used to get the current text size.

4.8Setting Text Encoding

4.8.1mdolphin_get_text_encoding_name

Table 4.21**mdolphin_get_text_encoding_name** function

Syntax	BOOL mdolphin_get_text_encoding_name(HWND hwnd, ENCODING_NAME *name)
Parameters	*hwnd* The handle of mDolphin Control.
	name Pointer to a buffer receiving the text encoding name. For more information about character encoding, please refer to appendix A Character Encoding.
Returns	TRUE on success, FALSE on error.
Descriptio	Used to get the current text encoding name of the browser.

n	

4.8.2mdolphin_set_text_encoding_name

Table 4.22**mdolphin_set_text_encoding_name** function

Syntax	BOOL mdolphin_set_text_encoding_name(HWND hwnd, ENCODING_NAME name)
Parameters	*hwnd* The handle of mDolphin Control.
	name The text encoding name. For more information about character encoding, please refer to appendix A Character Encoding.
Returns	TRUE on success, FALSE on error.
Description	Used to set the current text encoding name of the browser.

4.9Setting Preferences

4.9.1mdolphin_import_setup

Table 4.23**mdolphin_import_setup** function

Syntax	BOOL mdolphin_import_setup(HWND hwnd, const SETUP_INFO *setup)
Parameters	*hwnd* The handle of mDolphin Control.
	setup Pointer to a setup information.
Returns	TRUE on success, FALSE on error.
Description	Used to update the current preferences of the browser. You can set the default language, standard font, default encoding, whether auto load images, whether auto detect encoding, whether enable JavaScript, whether enable cookies, whether enable plug-in, and so on.

4.9.2mdolphin_fetch_setup

Table 4.24**mdolphin_fetch_setup** function

Syntax	BOOL mdolphin_fetch_setup(HWND hwnd, SETUP_INFO *setup)

Parameters	*hwnd*	The handle of mDolphin Control.
	setup	Pointer to a buffer receiving the current setup information.
Returns	TRUE on success, FALSE on error.	
Description	Used to get the current preferences of the browser.	

4.10Customizing Text Search

4.10.1mdolphin_search_for

Table 4.25**mdolphin_search_for** function

Syntax	BOOL mdolphin_search_for(HWND hwnd, const char* str, BOOL forward, BOOL caseSensitive, BOOL wrap)	
Parameters	*hwnd*	The handle of mDolphin Control.
	str	The string to search for.
	forward	TRUE to search forward, FALSE to search backwards.
	caseSensitive	TRUE to for case-sensitive search, FALSE for case-insensitive search.
	wrap	TRUE to wrap around, FALSE to avoid wrapping.
Returns	TRUE if found, FALSE if not found.	
Description	Used to searches a document view for a string and highlights the string if it is found.	

4.10.2mdolphin_mark_all_matches_for_text

Table 4.26**mdolphin_search_for** function

Syntax	unsigned int mdolphin_mark_all_matches_for_text(HWND hwnd, const char *search, BOOL caseSensitive, BOOL highlight, unsigned int limit)

Parameters	*hwnd*	The handle of mDolphin Control.
	search	The string to search for.
	caseSensitive	TRUE to for case-sensitive search, FALSE for case-insensitive search.
	highlight	TRUE to highlight, FALSE to un-highlight.
	limit	The specified matching count. A limit of 0 means no limit.
Returns	The matching count, zero if no match.	
Description	Used to mark all matching strings in a document view and highlights or un-highlights the strings.	

4.10.3 mdolphin_unmark_all_text_matches

Table 4.27**mdolphin_unmark_all_text_matches** function

Syntax	BOOL mdolphin_unmark_all_text_matches(HWND hwnd)	
Parameters	*hwnd*	The handle of mDolphin Control.
Returns	TRUE on success, FALSE on error.	
Description	Used to unmark all matching text in a document view.	

4.11Setting In View Source Mode

4.11.1mdolphin_set_in_view_source_mode

Table 4.28**mdolphin_set_in_view_source_mode** function

Syntax	BOOL mdolphin_set_in_view_source_mode(HWND hwnd, BOOL flag)	
Parameters	*hwnd*	The handle of mDolphin Control.
	flag	TRUE for view source mode, FALSE otherwise.
Returns	TRUE on success, FALSE on error.	
Description	Used to place a view into a special source-viewing mode.	

4.11.2mdolphin_in_view_source_mode

Table 4.29**mdolphin_in_view_source_mode** function

Syntax	BOOL mdolphin_in_view_source_mode(HWND hwnd)
Parameters	*hwnd* The handle of mDolphin Control.
Returns	TRUE in view source mode, FALSE otherwise.
Description	Used to get whether or not the view is in source view mode for HTML.

4.12Customizing Visited URL

4.12.1CB_URL_IS_VISITED

Table 4.30**CB_URL_IS_VISITED** callback function

Syntax	BOOL (*CB_URL_IS_VISITED) (const char *url)
Parameters	*url* The URL will be tested.
Returns	TRUE if URL has been visited, otherwise FALSE.
Description	Used to test whether the URL has been visited.

4.13Customizing Tool Tip

4.13.1CB_SET_TOOLTIP

Table 4.31**CB_SET_TOOLTIP** callback function

Syntax	void (* CB_SET_TOOLTIP) (HWND hWnd, int x, int y, const char * text,BOOL bShow)	
Parameters	*hWnd*	The handle of mDolphin Control.
	x	The x position of tool tip in window.
	y	The y position of tool tip in window.
	text	The tool tip message with UTF-8 character encoding.
	bShow	Whether show tip or not.

Returns	None
Description	Used to show a tool tip.

4.14Customizing User Agent

4.14.1CB_CUSTOM_USERAGENT

Table 4.32**CB_CUSTOM_USERAGENT** callback function

Syntax	char *(*CB_CUSTOM_USERAGENT) (const char *url)
Parameters	*url* The URL of the visiting site.
Returns	The user defined User Agent encoded by UTF-8 for mDolphin.
Description	Used to set User-Agent according to URL.

4.15Setting Proxy

4.15.1mdolphin_set_proxy

Table 4.33**mdolphin_set_proxy** function

Syntax	BOOL mdolphin_set_proxy(const PROXY_INFO *proxy_info, BOOL enable)	
Parameters	*proxy_info*	The proxy information.
	enable	Active proxy.
Returns	TRUE if you set proxy successfully, otherwise FALSE.	
Description	Used to set the proxy when you use proxy to connect the network.	

4.15.2mdolphin_get_proxy

Table 4.34**mdolphin_get_proxy** function

Syntax	BOOL mdolphin_get_proxy(PROXY_TYPE type, PROXY_ITEM *item)

Parameters	*type*	The proxy type.
	item	Proxy item.
Returns	TRUE on success, FALSE on error.	
Description	Used to get the proxy information.	

4.16Setting SSL

4.16.1CB_PROVIDE_CLIENT_CERT

Table 4.35**CB_PROVIDE_CLIENT_CERT** callback function

Syntax	BOOL (*CB_PROVIDE_CLIENT_CERT) (CERT_DATA **cert, SSL_PKEY **pkey, const CERT_NAME **names, int count)	
Parameters	*cert*	The certificate.
	pkey	The private key.
	names	The client CA names of the list of client CAs sent from the server.
	count	The client CA number of the list of client CAs sent from the server.
Returns	TRUE if you set certificates successful, otherwise FALSE.	
Description	Used to set a certificate.	

4.16.2CB_VERIFY_SERVER_CERT

Table 4.36**CB_VERIFY_SERVER_CERT** callback function

Syntax	CERT_RESULT (*CB_VERIFY_SERVER_CERT) (CERT_RESULT result, CERT_DATA *cert)	
Parameters	*result*	The result verified by mDolphin. X509_V_RESULT is same with openssl verified code. For more information about verified result, please refer to x509_vfy.h in openssl.
	cert	The current certificate.
Returns	User verified result.	
Description	Used to decide whether trusting the current certificate.	

4.16.3mdolphin_set_caPath

Table 4.37**mdolphin_set_caPath** function

Syntax	BOOL mdolphin_set_caPath(const char *path)
Parameters	*path* The directory holding multiple CA certificates.
Returns	TRUE if you set certificate path successfully, otherwise FALSE.
Description	Used to specify a certificate directory holding alternate certificates to verify with.

4.17Setting HTTP Auth

4.17.1CB_PROVIDE_Auth

Table 4.38**CB_PROVIDE_AUTH** callback function

Syntax	BOOL (*CB_PROVIDE_AUTH) (const char *title, char *userName, char *password)	
Parameters	*title*	The title for authentication.
	userName	The user name for authentication.
	password	The password for authentication.
Returns	TRUE if users want to authenticate, otherwise FALSE.	
Description	Used to set the user name and the password for HTTP authentication.	

4.18Customizing Protocol Extension

4.18.1CB_SCHEME_HANDLER

Table 4.39**CB_SCHEME_HANDLER** callback function

Syntax	typedef BOOL(*CB_SCHEME_HANDLER)(const char *url, void *param)	
Parameters	*url*	The handled url with the specified scheme.
	param	The parameter of the callback function.
Returns	TRUE on success, FALSE on error.	
Descriptio	The callback function of the scheme handler.	

n	

4.18.2mdolphin_unregister_scheme_handler

Table 4.40**mdolphin_unregister_scheme_handler** function

Syntax	BOOL mdolphin_unregister_scheme_handler(const char* scheme)	
Parameters	*sheme*	The scheme of protocol.
Returns	TRUE on success, FALSE on error.	
Description	Used to un-register network protocol scheme handler.	

4.18.3mdolphin_register_scheme_handler

Table 4.41**mdolphin_register_scheme_handler** function

Syntax	BOOL mdolphin_register_scheme_handler(const char * scheme, CB_SCHEME_HANDLER cb, void *data)	
Parameters	*sheme*	The scheme of protocol.
	cb	The callback of the scheme handler.
	data	The parameter of the callback function.
Returns	TRUE on success, FALSE on error.	
Description	Used to register network protocol scheme handler.	

4.19Customizing Downloading

4.19.1CB_SAVE_FILE_DATA

Table 4.42**CB_SAVE_FILE_DATA** callback function

Syntax	BOOL (*CB_SAVE_FILE_DATA) (int id, const char *url, const void *Data, size_t DataSize, unsigned long TotalSize, int isFinish)

Parameters	*id*	The download ID. Make a difference among the different downloading contents with the same URL.
	url	The URL of the downloading page.
	Data	The downloading data stream.
	DataSize	The size of the current downloading data.
	TotalSize	The total size of the data stream.
	isFinish	If downloading is finished, isFinish is TRUE. Otherwise, isFinish is FALSE.
Returns	TRUE if users want to continue downloading the data stream, otherwise FALSE.	
Description	mDolphin provides this function for users to set save page as. This callback function is When MIME type is unsupported or content need to be downloaded, users can decide whether saving data to a local file.	

4.20Setting Callback Functions

4.20.1mdolphin_set_callback_info

Table 4.43**mdolphin_set_callback_info** function

Syntax	BOOL mdolphin_set_callback_info(HWND hwnd, const CB_INFO *cb)	
Parameters	*hwnd*	The handle of mDolphin Control.
	cb	The callback functions structure. CB_INFO is a structure, which contains some callback functions. For more information about CB_INFO, please refer to 4.18.2.
Returns	TRUE on success, FALSE on error.	
Description	In order to meet particular requirements or customization, users need to define some callback function. For setting callback functions, the application should call this callback function.	

4.20.2CB_INFO

Table 4.44**CB_INFO** struct

Syntax	typedef struct _CB_INFO{ void (*CB_MESSAGE_BOX)(HWND hWnd, const char *pszText, const char* pszCaption);

	BOOL (*CB_CONFIRM_BOX)(HWND hWnd, const char *pszText, const char* pszCaption); char *(*CB_PROMPT_BOX)(HWND hWnd, const char *pszText, const char* pszDefault, const char *pszCaption); void (*CB_SET_TITLE) (HWND hWnd, const char *pText); void (*CB_SET_LOCATION) (HWND hWnd, const char *plocText); void (*CB_SET_STATUS)(HWND hWnd, const char *status_msg); void (*CB_SET_LOADING_STATUS) (HWND hWnd, BOOL bIsLoading, unsigned int progress); void (*CB_SET_HISTORY_STATUS) (HWND hWnd, unsigned int bListCount, unsigned int fListCount, unsigned int capacity); void (*CB_ERROR) (HWND hWnd, int errCode, const char *url); BOOL (*CB_CHOOSEFILE_BOX) (HWND hWnd, char *FileName, size_t BufferSize, BOOL IsSave); BOOL (*CB_SAVE_FILE_DATA) (int id, const char *url, const void *Data, size_t DataSize, unsigned long TotalSize, int isFinish); void (*CB_SET_IME_STATUS) (BOOL bstate); BOOL (*CB_PROVIDE_AUTH) (const char *title, char *userName, char *password); BOOL (*CB_URL_IS_VISITED) (const char *url); BOOL (*CB_CREATE_POPUP_MENU) (HWND hWnd, const POPUP_MENU_DATA *data); HWND (*CB_OPEN_WINDOW) (const char *url, DWORD styles, int x, int y, int width, int height); void (*CB_CLOSE_WINDOW) (HWND hWnd); BOOL (*CB_PROVIDE_CLIENT_CERT) (CERT_DATA **cert, SSL_PKEY **pkey, const CERT_NAME **names, int count); CERT_RESULT (*CB_VERIFY_SERVER_CERT) (CERT_RESULT result, CERT_DATA *cert); } CB_INFO;

Parameters	*CB_MESSAGE_BOX*	Please refer to 4.1.1.
	CB_CONFIRM_BOX	Please refer to 4.1.2.
	CB_PROMPT_BOX	Please refer to 4.1.3.
	CB_SET_TITLE	Please refer to 4.2.5.
	CB_SET_LOCATION	Please refer to 4.2.3.
	CB_SET_STATUS	Please refer to 4.2.4.
	CB_SET_LOADING_STATUS	Please refer to 4.2.2.
	CB_SET_HISTORY_STATUS	Please refer to 4.2.6.
	CB_ERROR	Please refer to 4.2.7.
	CB_CHOOSEFILE_BOX	Please refer to 4.1.4.
	CB_SAVE_FILE_DATA	Please refer to 4.17.1.
	CB_SET_IME_STATUS	Please refer to 4.2.1.
	CB_PROVIDE_AUTH	Please refer to 4.15.1.
	CB_URL_IS_VISITED	Please refer to 4.12.1.
	CB_CREATE_POPUP_MENU	Please refer to 4.3.1.
	CB_OPEN_WINDOW	Please refer to 4.4.1.
	CB_CLOSE_WINDOW	Please refer to 4.4.2.
	CB_PROVIDE_CLIENT_CERT	Please refer to 4.14.1.
	_CB_VERIFY_SERVER_CERT	Please refer to 4.14.2.
Description	CB_INFO is an important structure, which contains many callback functions. Based on these callback functions, users can define some interactive functions. It is used by mdolphin_set_callback_info function, please refer to 4.18.1.	

4.21JavaScript Native Binding

4.21.1mdolphin_define_jsnativeclass

Table 4.45**mdolphin_define_jsnativeclass** function

Syntax	BOOL mdolphin_define_jsnativeclass(JSNativeClass *nativeClass)	
Parameters	*nativeClass*	The description of the Native Class.
Returns	TRUE on success, FALSE on error.	
Description	Define a JavaScript native binding class.	

4.21.2mdolphin_define_jsnativefunction

Table 4.46**mdolphin_define_jsnativefunction** function

Syntax	BOOL mdolphin_define_jsnativefunction(JSNativeFunction *func)
Parameters	*func* The description of the Native Function.
Returns	TRUE on success, FALSE on fail.
Description	Define a javascript native binding function.

4.21.3mdolphin_lookup_jsnativeclass

Table 4.47**mdolphin_lookup_jsnativeclass** function

Syntax	const JSClassRef mdolphin_lookup_jsnativeclass(const char* name)
Parameters	*name* The name of the Native Class.
Returns	Non-NULL JSClassRef on success, NULL on fail.
Description	Undefined a JavaScript native binding class.

4.21.4mdolphin_undefine_jsnativeclass

Table 4.48**mdolphin_undefine_jsnativeclass** function

Syntax	BOOL mdolphin_undefine_jsnativeclass(JSNativeClass *nativeClass)
Parameters	*nativeClass* The description of the Native Class.
Returns	TRUE on success, FALSE on fail.
Description	Undefined a JavaScript native binding class.

4.21.5mdolphin_undefine_jsnativefunction

Table 4.49**mdolphin_undefine_jsnativefunction** function

Syntax	BOOL mdolphin_undefine_jsnativefunction(JSNativeFunction *func)
Parameters	*func* The description of the Native Function.
Returns	TRUE on success, FALSE on fail.
Description	Undefined a JavaScript native binding function.

5 Sample Code

This chapter presents sample code for the major activities you can perform by using the mDolphin API. These activities include:

- Loading Content.
- Setting the Rendering Mode.
- Customizing the Dialogs.

For more sample codes, please refer to mDolphin application demos.

5.1Loading Content

This example in this section shows how to load content from a specified URL. The following example shows how to download a Web page with a specified URL using function. This chapter will describe all functions of mDolphin in detail.

```
static const char * home_url = "http://www.minigui.com/index.php?id=product";
if (mdolphin_hwnd )
mdolphin_navigate(mdolphin_hwnd, NAV_GOTO, home_url , FALSE);
```

Notes on the code:

- **mdolphin_hwnd** is the mDolphin handle.
- **FALSE** means that the history list will save the visiting home_url.

5.2Setting the Rendering Mode

The following example shows how to set the global view rendering mode for the current browser.

```
if (mdolphin_hwnd )
```

```
mdolphin_set_rendering_mode(mdolphin_hwnd, MD_VIRTUALVIEW_MODE, 400);
```

Notes on the code:

- **mdolphin_hwnd** is the mDolphin handle.
- **400** means the max width of the view.

5.3Customizing the Dialogs

The following example shows how to customize the alert dialog.

```
static void my_message_callback (HWND parent, const char * text, const char * caption)
{
	MessageBox (parent, text, caption, MB_OK);
}

void set_callback_info(HWND hWnd)
{
	CB_INFO cb_info;
	memset (&cb_info, 0, sizeof(CB_INFO));

	cb_info.CB_MESSAGE_BOX = my_message_callback;
	.
	.
	.
	if (mdolphin_hwnd)
	mdolphin_set_callback_info(mdolphin_hwnd, &cb_info);
}
```

Notes on the code:

- **mdolphin_hwnd** is the mDolphin handle.
- **mdolphin_set_callback_info** is mDolphin API used to set callb functions for the browser.
- **my_message_callback** is user defined function used to d

message to the user until the user presses OK.

- **MessageBox** is MiniGUI API used to display a message box.

5.4Binding native code

5.4.1Binding Native Function

Below, JavaScript invoke custom function **myprint** to print message.

```
<script type="javascript>
myprint("Hi!, It's myprint");
</script>
```

The following example shows how to define a JSNativeFunction and how to do JavaScript can access **myPrint**.

```
/* 1.declare JSNativeFunction Print */
static JSNativeFunction Print = {
       "myprint", myPrint, 0
};

/* 2.implement  binded native function myPrint */
static   JSValueRef   myPrint(JSContextRef   context,   JSObjectRef   function,   JSObjectRef
thisObject, size_t argumentCount, const JSValueRef arguments[], JSValueRef* exception)
{
       if (argumentCount >= 1) {
              JSStringRef str = JSValueToStringCopy(context, arguments[0], exception);
              size_t len = JSStringGetLength(str) + 1;
              char *buf = new char[len];
              memset(buf, 0x0, len);
              JSStringGetUTF8CString(str, buf, len-1);
              JSStringRelease(str);
              printf("str=%s\n", buf);
              delete[] buf;
       }
       return JSValueMakeUndefined(context);
```

```
}
.......
/*3.register Print for Javascript */
int MDolphinProc (HWND hWnd, int message, WPARAM wParam, LPARAM lParam)
{
	switch (message) {
		case MSG_CREATE:
		...
		mdolphin_define_jsnativefunction(&Print);

		...
		break;
	}
}
```

5.4.2Binding Native Class

The following example shows how to define a JSNativeClass variable **player** and bind it for the following JavaScript.

```
<script>
var player= new MediaPlayer;
player.open("rtsp://www.mingui.com");
player.play();
</script>

/* 1. define MediaPlayer class and implent the member functions open,play,stop and close
*/
struct MediaPlayer{
	int   x;
	int   y;
	int   w;
	int   h;
	int open(const char *url)
	{
		print("url=%s\n",, url);
		return 0;
```

```
	}

	int play()
	{
		print("[%s,%d]\n", __FUNCTION__, __LINE__);
		return 0;
	}
	void stop() {print("[%s,%d]\n", __FUNCTION__, __LINE__);}
	void close() {print("[%s,%d]\n", __FUNCTION__, __LINE__);}
};

/* 2.declare the binded method */
static JSNativeFunction playerMethod[] = {
	{"open", player_open, kJSPropertyAttributeDontDelete},
	{"play", player_play, kJSPropertyAttributeDontDelete},
	{"stop", player_stop, kJSPropertyAttributeDontDelete},
	{"close", player_close, kJSPropertyAttributeDontDelete},
	{0, 0, 0}
};

/* 3. declare the binded property */
static JSProperty playerProperty[] = {
	{"x", player_getX, player_setX, kJSPropertyAttributeDontDelete},
	{0, 0, 0, 0}
};
/* 4.declare the JSNativeClass variable Player */
static JSNativeClass Player = {
	"MediaPlayer",
	0,
	player_constructor,
	player_destructor,
	playerMethod,
	playerProperty,
};

/*  5.  implent  binded  functions    player_constructor,  player_destructor,  player_open,
player_play,
* player_stop, player_close, player_getX, and player_setX
```

```
*/

/* create a instance of the MediaPlayer class and attach it to a JSObjectRef object; */
static JSObjectRef player_constructor(JSContextRef context, JSObjectRef constructor,
size_t argumentCount, const JSValueRef arguments[], JSValueRef* exception)
{
	MediaPlayer *mplayer = new MediaPlayer;
	JSObjectRef newObj = JSObjectMake(context,
mdolphin_lookup_jsnativeclass(Player.name), mplayer);
	return newObj;
}

// destroy the MediaPlayer instance and the JSObjectRef object;
static void player_destructor(JSObjectRef object)
{
	MediaPlayer *mplayer = (MediaPlayer*)JSObjectGetPrivate(object);
	delete mplayer;
	JSObjectSetPrivate(object, 0);
}

static JSValueRef player_open(JSContextRef context, JSObjectRef function, JSObjectRef
thisObject, size_t argumentCount, const JSValueRef arguments[], JSValueRef* exception)
{
	if (argumentCount >= 1) {
		MediaPlayer *mplayer = (MediaPlayer*)JSObjectGetPrivate(thisObject);
		JSStringRef strUrl = JSValueToStringCopy(context, arguments[0], exception);
		char buf[20];
		memset(buf, 0x0, sizeof(buf));
		JSStringGetUTF8CString(strUrl, buf, 19);
		mplayer->open(buf);
		JSStringRelease(strUrl);
	}
	return JSValueMakeUndefined(context);
}

static JSValueRef player_play(JSContextRef context, JSObjectRef function, JSObjectRef
thisObject, size_t argumentCount, const JSValueRef arguments[], JSValueRef* exception)
```

```
{
	MediaPlayer *mplayer = (MediaPlayer*)JSObjectGetPrivate(thisObject);
	mplayer->play();
	return JSValueMakeUndefined(context);
}

static JSValueRef player_stop(JSContextRef context, JSObjectRef function, JSObjectRef thisObject, size_t argumentCount, const JSValueRef arguments[], JSValueRef* exception)
{
	MediaPlayer *mplayer = (MediaPlayer*)JSObjectGetPrivate(thisObject);
	mplayer->stop();
	return JSValueMakeUndefined(context);
}

static JSValueRef player_close(JSContextRef context, JSObjectRef function, JSObjectRef thisObject, size_t argumentCount, const JSValueRef arguments[], JSValueRef* exception)
{
	MediaPlayer *mplayer = (MediaPlayer*)JSObjectGetPrivate(thisObject);
	mplayer->close();
	return JSValueMakeUndefined(context);
}

/* get the value of the property x */
static JSValueRef player_getX(JSContextRef context, JSObjectRef object, JSStringRef propertyName, JSValueRef* exception)
{
	MediaPlayer *mplayer = (MediaPlayer*)JSObjectGetPrivate(object);
	return JSValueMakeNumber(context, mplayer->x);
}

/* set the value of the property x */
static bool player_setX(JSContextRef context, JSObjectRef object, JSStringRef propertyName, JSValueRef value, JSValueRef* exception)
{
	MediaPlayer *mplayer = (MediaPlayer*)JSObjectGetPrivate(object);
	mplayer->x = (int)JSValueToNumber(context, value, exception);
	printf("[%s,%d]x=%d\n", __FUNCTION__, __LINE__, mplayer->x);
```

```
        return true;
}
```

Appendix A Character Encoding

mDolphin support several character encoding, you can use the following value to set the default character encoding for your browser.

```
typedef enum {
	MD_CHARSET_AUTODETECT = -1,
	MD_CHARSET_ISO8859_1,
	MD_CHARSET_ISO8859_2,
	MD_CHARSET_ISO8859_3,
	MD_CHARSET_ISO8859_4,
	MD_CHARSET_ISO8859_5,
	MD_CHARSET_ISO8859_6,
	MD_CHARSET_ISO8859_7,
	MD_CHARSET_ISO8859_8,
	MD_CHARSET_ISO8859_9,
	MD_CHARSET_ISO8859_10,
	MD_CHARSET_ISO8859_11,
	MD_CHARSET_ISO8859_12,
	MD_CHARSET_ISO8859_13,
	MD_CHARSET_ISO8859_14,
	MD_CHARSET_ISO8859_15,
	MD_CHARSET_ISO8859_16,
	MD_CHARSET_EUC_CN,
	MD_CHARSET_GB1988_0,
	MD_CHARSET_GB2312_0,
	MD_CHARSET_GBK,
	MD_CHARSET_GB18030_0,
	MD_CHARSET_BIG5,
	MD_CHARSET_KSC5636_0,
	MD_CHARSET_KSC5601_0,
	MD_CHARSET_EUCJP,
	MD_CHARSET_JISX0201_0,
	MD_CHARSET_JISX0208_0,
	MD_CHARSET_SHIFTJIS,
```

```
      MD_CHARSET_JISX0201_1,
      MD_CHARSET_JISX0208_1,
      MD_CHARSET_ISO_10646_1,
      MD_CHARSET_UTF8,
} ENCODING_NAME;
```

Appendix B Error Code

When mDolphin network module fails to connect with the server, it will return error code. All error codes are listed below.

```
typedef enum {
	/** The following error codes are network error codes.*/
	/** It's not error code, just as a base number for network error code.*/
	MDEC_NET_BASE = 400000,

	/** The following error codes are network normal error codes.*/
	/** It's not error code, just as a base number for network normal error code.*/
	MDEC_NET_NORMAL_BASE = MDEC_NET_BASE +0,
	/** The URL was not properly formatted. */
	MDEC_NET_URL_ERROR = MDEC_NET_NORMAL_BASE +1,
	/** The protocol was not supported. */
	MDEC_NET_UNSUPPORTED_PROTOCOL,
	/** Couldn't resolve the host. */
	MDEC_NET_DNS_ERROR,
	/** Failed to connect() to host or proxy. */
	MDEC_NET_COULDNT_CONNECT,
	/** It's an unknown network error.*/
	MDEC_NET_UNKNOWN_ERROR,

	/** The following error codes are network HTTP error codes.*/
	/** It's not error code, just as a base number for network HTTP error.*/
	MDEC_NET_HTTP_BASE = MDEC_NET_BASE +1000,
	#if ENABLE_NATIVEERROR
	/** It's the same as the HTTP server response code 400. */
	MDEC_NET_HTTP_400 = MDEC_NET_HTTP_BASE +400,
	/** It's the same as the HTTP server response code 401. */
	MDEC_NET_HTTP_401,
	/** It's the same as the HTTP server response code 402. */
	MDEC_NET_HTTP_402,
	/** It's the same as the HTTP server response code 403. */
	MDEC_NET_HTTP_403,
```

```
/** It's the same as the HTTP server response code 404. */
MDEC_NET_HTTP_404,
/** It's the same as the HTTP server response code 405. */
MDEC_NET_HTTP_405,
/** It's the same as the HTTP server response code 406. */
MDEC_NET_HTTP_406,
/** It's the same as the HTTP server response code 407. */
MDEC_NET_HTTP_407,
/** It'sthe same as the HTTP server response code 408. */
MDEC_NET_HTTP_408,
/** It's the same as the HTTP server response code 409. */
MDEC_NET_HTTP_409,
/** It's the same as the HTTP server response code 410. */
MDEC_NET_HTTP_410,
/** It's the same as the HTTP server response code 411. */
MDEC_NET_HTTP_411,
/** It's the same as the HTTP server response code 412. */
MDEC_NET_HTTP_412,
/** It's the same as the HTTP server response code 413. */
MDEC_NET_HTTP_413,
/** It's the same as the HTTP server response code 414. */
MDEC_NET_HTTP_414,
/** It's the same as the HTTP server response code 415. */
MDEC_NET_HTTP_415,
/** It's the same as the HTTP server response code 416. */
MDEC_NET_HTTP_416,
/** It's the same as the HTTP server response code 417. */
MDEC_NET_HTTP_417,
/** It's the same as the HTTP server response code 500. */
MDEC_NET_HTTP_500 = MDEC_NET_HTTP_BASE +500,
/** It's the same as the HTTP server response code 501. */
MDEC_NET_HTTP_501,
/** It's the same as the HTTP server response code 502. */
MDEC_NET_HTTP_502,
/** It's the same as the HTTP server response code 503. */
MDEC_NET_HTTP_503,
/** It's the same as the HTTP server response code 504. */
```

```
MDEC_NET_HTTP_504,
/** It's the same as the HTTP server response code 505. */
MDEC_NET_HTTP_505,
#endif

/** The following error codes are network FTP error codes.*/
/** It's not error code, just as a base number for network FTP error.*/
MDEC_NET_FTP_BASE = MDEC_NET_BASE +2000,
/** It's the same as the FTP server response code 421. */
MDEC_NET_FTP_421 = MDEC_NET_FTP_BASE +421,
/** It's the same as the FTP server response code 425. */
MDEC_NET_FTP_425 = MDEC_NET_FTP_BASE +425,
/** It's the same as the FTP server response code 426. */
MDEC_NET_FTP_426,
/** It's the same as the FTP server response code 450. */
MDEC_NET_FTP_450 = MDEC_NET_FTP_BASE +450,
/** It's the same as the FTP server response code 451. */
MDEC_NET_FTP_451,
/** It's the same as the FTP server response code 452. */
MDEC_NET_FTP_452,
/** It's the same as the FTP server response code 500. */
MDEC_NET_FTP_500 = MDEC_NET_FTP_BASE +500,
/** It's the same as the FTP server response code 501. */
MDEC_NET_FTP_501,
/** It's the same as the FTP server response code 502. */
MDEC_NET_FTP_502,
/** It's the same as the FTP server response code 503. */
MDEC_NET_FTP_503,
/** It's the same as the FTP server response code 504. */
MDEC_NET_FTP_504,
/** It's the same as the FTP server response code 530. */
MDEC_NET_FTP_530 = MDEC_NET_FTP_BASE +530,
/** It's the same as the FTP server response code 532. */
MDEC_NET_FTP_532 = MDEC_NET_FTP_BASE +532,
/** It's the same as the FTP server response code 550. */
MDEC_NET_FTP_550 = MDEC_NET_FTP_BASE +550,
/** It's the same as the FTP server response code 551. */
MDEC_NET_FTP_551,
```

```
/** It's the same as the FTP server response code 552. */
MDEC_NET_FTP_552,
/** It's the same as the FTP server response code 553. */
MDEC_NET_FTP_553,
/** It's an unknown FTP error. */
MDEC_NET_FTP_UNKNOWN_ERROR = MDEC_NET_FTP_BASE +900,

/** The following error codes are network FILE error codes.*/
/** It's not error code, just as a base number for network FILE error.*/
MDEC_NET_FILE_BASE = MDEC_NET_BASE +3000,
/** It's a FILE error. A file given with FILE:// couldn't be opened.*/
MDEC_NET_FILE_READ_ERROR = MDEC_NET_FILE_BASE +1,

/** The following error codes are network SSL error codes.*/
/** It's not error code, just as a base number for network SSL error.*/
MDEC_NET_SSL_BASE = MDEC_NET_BASE +4000,
/** It's an SSL error which occurred somewhere in the SSL/TLS handshake. */
MDEC_NET_SSL_CONNECT_ERROR = MDEC_NET_SSL_BASE +1,
/** The remote server's SSL certificate was deemed not OK.*/
MDEC_NET_SSL_PEER_CERTIFICATE,
/** The specified crypto engine wasn't found.*/
MDEC_NET_SSL_ENGINE_NOTFOUND,
/** Failed setting the selected SSL crypto engine as default.*/
MDEC_NET_SSL_ENGINE_SETFAILED,
/** Problem with the local client certificate.*/
MDEC_NET_SSL_CERTPROBLEM,
/** Couldn't use specified cipher.*/
MDEC_NET_SSL_CIPHER,
/** Peer certificate cannot be authenticated with known CA certificates.*/
MDEC_NET_SSL_CACERT,
/** Requested FTP SSL level failed.*/
MDEC_NET_SSL_FTP_ERROR,
/** Initiating the SSL Engine failed.*/
MDEC_NET_SSL_ENGINE_INITFAILED,
/** Problem with reading the SSL CA cert (path? access rights?)*/
MDEC_NET_SSL_CACERT_BADFILE,

/** The following error codes are network PROXY error codes.*/
/** It's not error code, just as a base number for network PROXY error.*/
```

```
      MDEC_NET_PROXY_BASE = MDEC_NET_BASE +5000,
      /** It's a PROXY error. */
      MDEC_NET_PROXY_ERROR = MDEC_NET_PROXY_BASE +1,
} ERROR_CODE;
```