MySQLMobile

DLL and LIB Programmer's guide

index

function
mysql_enable_compression(`int enable`)

function description
Use this function to enable/disable compression in MySQLMobile. Compression is ON by default, so you don't have to call this function to enable it.
Make sure you call this function before calling any other MySQLMobile function.

parameters

`[in] enable`	`1`	→	enable compression
	0	→	disable compression

return values
`n/a`

function

```
int mysql_connect(char *pszhost, int nport, char *pszusername,
                  char *pszpassword)
```

function description

Use this function to connect to a mySQL database.

parameters

`[in] pszhost`	host to connect to
`[in] nport`	port to connect to (default mySQL port is 3306)
`[in] pszusername`	username to access database
`[in] pszpassword`	password to access database

return values

`0`	when everything went ok
`not 0`	when an error occurred. Get the error in textual form by calling `mysql_get_last_error(<retval>)`

C/C++ example of usage

```
int nRet;

nRet = mysql_connect("192.168.0.15", 3306, "my_username", "my_passwd");
if (nRet == 0) {
        CString str = _T("mysql_connect() successful");
        MessageBox(NULL, str, _T("info"), MB_OK | MB_ICONINFORMATION);
} else {
        char *errorBuf = new char[255];
        mysql_get_last_error(nRet, errorBuf);
        MessageBox(NULL, CString(errorBuf), _T("error"), MB_OK | MB_ICONINFORMATION);
        delete errorBuf;
}
```

VB.NET / C# example of usage

You can find other examples of usage in \Examples.

function
`int mysql_disconnect()`

function description
Use this function to disconnect from a mySQL database.

parameters
n/a

return values

`0`	when everything went ok
`not 0`	when an error occurred. Get the error in textual form by calling `mysql_get_last_error(<retval>)`

C/C++ example of usage

```
int nRet;

nRet = mysql_disconnect();
if (nRet == 0) {
	CString str = _T("mysql_disconnect() successful");
	MessageBox(NULL, str, _T("info"), MB_OK | MB_ICONINFORMATION);
} else {
	char *errorBuf = new char[255];
	mysql_get_last_error(nRet, errorBuf);
	MessageBox(NULL, CString(errorBuf), _T("error"), MB_OK | MB_ICONINFORMATION);
	delete errorBuf;
}
```

function

```
void mysql_get_last_error(int nErrorCode, char* pszError)
```

function description

Use this function to get a textual description of the last error that has occurred. After a function returns an error code (i.e. `mysql_connect()` returns a value not equal to 0), then you can use this error code to retrieve a textual description of the error.

parameters

`[in]`	`nErrorCode`	error code
`[out]`	`pszError`	buffer for textual error description Note that you should allocate/deallocate this buffer. 255 bytes should be enough to hold all possible error messages.

return values

`n/a`

Possible values for nErrorCode

```
#define EXTERNAL_ERROR                          -2000
#define CONNECTION_TIMED_OUT                    -1001
#define SOCKET_NOT_VALID                        -1002
#define SOCKET_CONNECTION_FAILED                -1003
#define MYSQL_NOT_CONNECTED                     -1004
#define WRONG_PACKET_DATA                       -1005
#define EMPTY_PARAMETER_NOT_ALLOWED             -1006
#define FIELD_NOT_FOUND                         -1007
#define ROW_NOT_FOUND                           -1008
#define UNKNOWN_FIELD_PARAMETER                 -1009
#define NO_SUCH_COLUMN                          -1010
#define NO_SUCH_ROW                             -1011
#define ZLIB_DECOMPRESSION_ERROR                -1012
```

C/C++ example of usage

See example of `mysql_connect()`

function
`int mysql_select_db(char *pszDatabaseName)`

function description
Use this function to select a database to work with.

parameters

`[in] pszDatabaseName`	database name

return values

`0`	when everything went ok
`not 0`	when an error occurred. Get the error in textual form by calling `mysql_get_last_error(<retval>)`

C/C++ example of usage

```
int nRet;

nRet = mysql_select_db("test");
if (nRet == 0) {
	CString str = _T("mysql_select_db() successful");
	MessageBox(NULL, str, _T("info"), MB_OK | MB_ICONINFORMATION);
} else {
	char *errorBuf = new char[255];
	mysql_get_last_error(nRet, errorBuf);
	MessageBox(NULL, CString(errorBuf), _T("error"), MB_OK | MB_ICONINFORMATION);
	delete errorBuf;
}
```

function
`int mysql_execute_query(char *pszQueryString)`

function description
Use this function to execute an SQL query. After a successful calling of this function, you can retrieve the results of a query with the functions:
`mysql_get_last_insert_id()`
`mysql_get_last_servermessage(...)`
`mysql_get_fieldcount()`
`mysql_get_field(...)`
`mysql_get_rowcount()`
`mysql_get_data(...)`

Due to memory and/or bandwidth limitation of mobile devices, it is recommended to use the LIMIT keyword in SELECT queries. This allows you to go through tables step by step with for example 50 records at a time.
Also note that at this moment there is no support for BLOB fields so it is recommended to avoid using these.

parameters

`[in] pszQueryString`	the SQL query

return values

`0`	when everything went ok
`not 0`	when an error occurred. Get the error in textual form by calling `mysql_get_last_error(<retval>)`

C/C++ example of usage

```
int nRet;

nRet = mysql_execute_query("UPDATE my_table SET name='john' WHERE id=1");
if (nRet == 0) {
	CString str = _T("mysql_execute_query() successful");
	MessageBox(NULL, str, _T("info"), MB_OK | MB_ICONINFORMATION);

	char *msgBuf = new char[255];
	mysql_get_last_servermessage(msgBuf);
	MessageBox(NULL, CString(msgBuf), _T("info"), MB_OK | MB_ICONINFORMATION);
	delete msgBuf;
} else {
	char *errorBuf = new char[255];
	mysql_get_last_error(nRet, errorBuf);
	MessageBox(NULL, CString(errorBuf), _T("error"), MB_OK | MB_ICONINFORMATION);
	delete errorBuf;
}
```

See the section on function `mysql_get_field()` for an example of a SELECT query

function
`int mysql_get_last_insert_id()`

function description
Use this function to get the id of the last inserted row, after a successful SQL query. As an alternative you could also fire the SQL query "`SELECT LAST_INSERT_ID()`", which will give the same result.

parameters
`n/a`

return values
the last inserted id

C/C++ example of usage

```
int nRet;
CString str;


nRet = mysql_execute_query("INSERT INTO my_table (name) VALUES ('john')");
if (nRet == 0) {
	str = _T("mysql_execute_query() successful");
	MessageBox(NULL, str, _T("info"), MB_OK | MB_ICONINFORMATION);

	str.Format(_T("last insert id: %d"),  mysql_get_last_insert_id());
	MessageBox(NULL, str, _T("info"), MB_OK | MB_ICONINFORMATION);
} else {
	char *errorBuf = new char[255];
	mysql_get_last_error(nRet, errorBuf);
	MessageBox(NULL, CString(errorBuf), _T("error"), MB_OK | MB_ICONINFORMATION);
	delete errorBuf;
}
```

function

```
void mysql_get_last_servermessage(char *pszMessage)
```

function description

Use this function to retrieve the last server message, after a successful SQL query. You can use this function for example after INSERT or UPDATE queries. The server message has a format like (for example): `Rows matched: 1  Changed: 1  Warnings: 0` This message directly comes from the mySQL server.

parameters

`[out] pszMessage`	buffer for the server message Note that you should allocate/deallocate this buffer. 255 bytes should be enough to hold all possible messages.

return values

`n/a`

C/C++ example of usage

See example of `mysql_execute_query()`

function
`int mysql_get_fieldcount()`

function description
Use this function to retrieve the number of fields (i.e. the number of columns) in an query result, after a successful SQL query.

parameters
`n/a`

return values
The number of fields

C/C++ example of usage

See example of `mysql_get_data()`

function

```
int mysql_get_field(char *char_val, int &int_val,
                    int nParam, int nIndex)
```

function description

You can use this function to get information about fields, after you've successfully fired a query, for example a SELECT query.
Note that you can also get (more) information about fields when you fire the query SHOW COLUMNS FROM <tablename>
This is useful when you want the know the default field value, for example.

parameters

`[out] char_val`	buffer to hold characters Note that you should allocate/deallocate this buffer. 255 bytes should normally be enough to hold all possible values, but this depends on your mySQL definitions.
`[out] int_val`	buffer to hold integer value
`[in] nParam`	which field parameter to retrieve (see below for a list of possible parameters)
`[in] nIndex`	0-based index of field (get number of fields with mysql_get_fieldcount())

return values

`0`	when everything went ok
`not 0`	when an error occurred. Get the error in textual form by calling `mysql_get_last_error(<retval>)`

possible values for nParam

```
#define FIELD_PARAM_CATALOG   1      // (catalog, only supported by mysql 4.1+)
#define FIELD_PARAM_DB        2      // (database, only supported by mysql 4.1+)
#define FIELD_PARAM_TABLE     3      // (table name)
#define FIELD_PARAM_ORIGTABLE 4      // (original table name, only supported by
                                     //  mysql 4.1+)
#define FIELD_PARAM_FIELD     5      // (field name)
#define FIELD_PARAM_ORIGFIELD 6      // (original field name, only supported by
                                     //  mysql 4.1+)
#define FIELD_PARAM_LENGTH    7      // (field length, as specified in the mysql table)
#define FIELD_PARAM_TYPE      8      // (field type; possible values below)
#define FIELD_PARAM_FLAGS     9      // (field flags; possible values below)
#define FIELD_PARAM_DECIMALS  10     // (nr of decimals)
```

possible values for field type

```
enum enum_field_types
{
       FIELD_TYPE_DECIMAL=0x00,
       FIELD_TYPE_TINY=0x01,
       FIELD_TYPE_SHORT=0x02,
       FIELD_TYPE_LONG=0x03,
       FIELD_TYPE_FLOAT=0x04,
       FIELD_TYPE_DOUBLE=0x05,
       FIELD_TYPE_NULL=0x06,
```

```
        FIELD_TYPE_TIMESTAMP=0x07,
        FIELD_TYPE_LONGLONG=0x08,
        FIELD_TYPE_INT24=0x09,
        FIELD_TYPE_DATE=0x0a,
        FIELD_TYPE_TIME=0x0b,
        FIELD_TYPE_DATETIME=0x0c,
        FIELD_TYPE_YEAR=0x0d,
        FIELD_TYPE_NEWDATE=0x0e,
        FIELD_TYPE_VARCHAR=0x0f,
        FIELD_TYPE_BIT=0x10,
        FIELD_TYPE_NEWDECIMAL=0xf6,
        FIELD_TYPE_ENUM=0xf7,
        FIELD_TYPE_SET=0xf8,
        FIELD_TYPE_TINY_BLOB=0xf9,
        FIELD_TYPE_MEDIUM_BLOB=0xfa,
        FIELD_TYPE_LONG_BLOB=0xfb,
        FIELD_TYPE_BLOB=0xfc,
        FIELD_TYPE_VAR_STRING=0xfd,
        FIELD_TYPE_STRING=0xfe,
        FIELD_TYPE_GEOMETRY=0xff
};
```

possible values for field flags

```
#define NOT_NULL_FLAG                1      // Field can't be NULL
#define PRI_KEY_FLAG                 2      // Field is part of a primary key
#define UNIQUE_KEY_FLAG              4      // Field is part of a unique key
#define MULTIPLE_KEY_FLAG            8      // Field is part of a key
#define BLOB_FLAG                    16     // Field is a blob
#define UNSIGNED_FLAG                32     // Field is unsigned
#define ZEROFILL_FLAG                64     // Field is zerofill
#define BINARY_FLAG                  128    // Field is binary
/* The following are only sent to new clients */
#define ENUM_FLAG                    256    // field is an enum
#define AUTO_INCREMENT_FLAG          512    // field is a autoincrement field
#define TIMESTAMP_FLAG               1024   // Field is a timestamp
#define SET_FLAG                     2048   // field is a set
#define NO_DEFAULT_VALUE_FLAG        4096   // Field doesn't have default value
#define NUM_FLAG                     32768  // Field is num (for clients)
```

C/C++ example of usage

```
int nRet;
CString str;
int nFieldIndex=0;


// Execute SELECT query
nRet = mysql_execute_query("SELECT * FROM my_table LIMIT 20");
if (nRet == 0) {
        CString str = _T("mysql_execute_query() successful");
        MessageBox(NULL, str, _T("info"), MB_OK | MB_ICONINFORMATION);
} else {
        char *errorBuf = new char[255];
        mysql_get_last_error(nRet, errorBuf);
        MessageBox(NULL, CString(errorBuf), _T("error"), MB_OK | MB_ICONINFORMATION);
        delete errorBuf;
        return;
}

char *char_buffer = new char[255];
int int_buffer;

// Get field name
nRet = mysql_get_field(char_buffer, int_buffer, FIELD_PARAM_FIELD, nFieldIndex);
// char_buffer now contains field name
MessageBox(NULL, CString(char_buffer), _T("Field name"), MB_OK | MB_ICONINFORMATION);
```

```
// Get field length
nRet = mysql_get_field(char_buffer, int_buffer, FIELD_PARAM_LENGTH, nFieldIndex);
str.Format(_T("%d"), int_buffer);
MessageBox(NULL, CString(char_buffer), _T("Field length"), MB_OK | MB_ICONINFORMATION);


// Get field flag(s)
nRet = mysql_get_field(char_buffer, int_buffer, FIELD_PARAM_FLAGS, nFieldIndex);
if (int_buffer & AUTO_INCREMENT_FLAG) {
        MessageBox(NULL, _T("Field has auto-increment flag"), _T("Field flags"),
                    MB_OK | MB_ICONINFORMATION);
}

delete char_buffer;
```

function
`int mysql_get_rowcount()`

function description
Use this function to retrieve the number of rows in an query result, after a successful SQL query.

parameters
`n/a`

return values
The number of rows

C/C++ example of usage

See example of `mysql_get_data()`

function

```
int mysql_get_data(char *pszElem, int nRow, int nColumn)
```

function description

Use this function to retrieve row data, after a successful SQL query.

parameters

[out]	pszElem	buffer to hold row element Note that you should allocate/deallocate this buffer. 8K bytes should normally be enough to hold all possible values, but this depends on your mySQL definitions.
[in]	nRow	row number
[in]	nColumn	column number

return values

0	when everything went ok
not 0	when an error occurred. Get the error in textual form by calling mysql_get_last_error(<retval>)

C/C++ example of usage

```
int nRet;

char* data_buffer = new char[8192]; // max data size = 8K
CString str = _T("");
for (int i=0; i < mysql_get_rowcount(); i++) {
	for (int j=0; j < mysql_get_fieldcount(); j++) {
		nRet = mysql_get_data(data_buffer, i, j);
		if (nRet == 0) {
			str += (_T("[") + CString(data_buffer) + _T("] "));
		} else {
			char *errorBuf = new char[255];
			mysql_get_last_error(nRet, errorBuf);
			MessageBox(NULL, CString(errorBuf), _T("error"),
					MB_OK | MB_ICONINFORMATION);
			delete errorBuf;
		}
	}
	MessageBox(NULL, str, _T("row"), MB_OK | MB_ICONINFORMATION);
	str = _T("");
}
delete data_buffer;
```

Using binary data

If you want to use binary data, you should make sure it's base64 encoded in a text field (for example a *mediumtext* field) in mySQL. After getting the field data with MySQLMobile, you can decode the base64 encoded string back to binary data.
→ Query the least amount of records possible to minimize memory requirements and maximize response times.