

msql

COLLABORATORS

	<i>TITLE :</i> msql		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	mysql	1
1.1	mysql.doc	1
1.2	mysql.library/--background--	2
1.3	mysql.library/--rexxhost--	4
1.4	mysql.library/MsqlAddMHookA	5
1.5	mysql.library/MsqlAllocConnection	6
1.6	mysql.library/MsqlClose	6
1.7	mysql.library/MsqlConnect	7
1.8	mysql.library/MsqlCopyDB	8
1.9	mysql.library/MsqlCreateDB	8
1.10	mysql.library/MsqlDataSeek	9
1.11	mysql.library/MsqlDateOffset	10
1.12	mysql.library/MsqlDateToUnixTime	11
1.13	mysql.library/MsqlDiffDates	12
1.14	mysql.library/MsqlDiffTimes	12
1.15	mysql.library/MsqlDropDB	13
1.16	mysql.library/MsqlFetchField	14
1.17	mysql.library/MsqlFetchRow	15
1.18	mysql.library/MsqlFieldSeek	15
1.19	mysql.library/MsqlFreeConnection	16
1.20	mysql.library/MsqlFreeResult	17
1.21	mysql.library/MsqlGetCharConf	17
1.22	mysql.library/MsqlGetErrMsg	18
1.23	mysql.library/MsqlGetHostInfo	19
1.24	mysql.library/MsqlGetIntConf	19
1.25	mysql.library/MsqlGetProtoInfo	20
1.26	mysql.library/MsqlGetSequenceInfo	20
1.27	mysql.library/MsqlGetServerInfo	21
1.28	mysql.library/MsqlGetServerStats	21
1.29	mysql.library/MsqlListDBs	22

1.30	mysql.library/MsqlListFields	23
1.31	mysql.library/MsqlListIndex	24
1.32	mysql.library/MsqlListTables	25
1.33	mysql.library/MsqlLoadConfigFile	26
1.34	mysql.library/MsqlMoveDB	26
1.35	mysql.library/MsqlNumFields	27
1.36	mysql.library/MsqlNumRows	28
1.37	mysql.library/MsqlQuery	29
1.38	mysql.library/MsqlReloadAcls	30
1.39	mysql.library/MsqlRemMHook	30
1.40	mysql.library/MsqlSelectDB	31
1.41	mysql.library/MsqlShutdown	32
1.42	mysql.library/MsqlStoreResult	32
1.43	mysql.library/MsqlSumTimes	33
1.44	mysql.library/MsqlTimeToUnixTime	34
1.45	mysql.library/MsqlUnixTimeToDate	34
1.46	mysql.library/MsqlUnixTimeToTime	35

Chapter 1

msql

1.1 msql.doc

```
--background--  
  
--rexxhost--  
  
MsqlAddMHookA()  
MsqlAllocConnection()  
MsqlClose()  
MsqlConnect()  
MsqlCopyDB()  
MsqlCreateDB()  
MsqlDataSeek()  
MsqlDateOffset()  
MsqlDateToUnixTime()  
MsqlDiffDates()  
MsqlDiffTimes()  
MsqlDropDB()  
MsqlFetchField()  
MsqlFetchRow()  
MsqlFieldSeek()  
MsqlFreeConnection()  
MsqlFreeResult()
```

MsqlGetCharConf ()
MsqlGetErrMsg ()
MsqlGetHostInfo ()
MsqlGetIntConf ()
MsqlGetProtoInfo ()
MsqlGetSequenceInfo ()
MsqlGetServerInfo ()
MsqlGetServerStats ()
MsqlListDBs ()
MsqlListFields ()
MsqlListIndex ()
MsqlListTables ()
MsqlLoadConfigFile ()
MsqlMoveDB ()
MsqlNumFields ()
MsqlNumRows ()
MsqlQuery ()
MsqlReloadAcls ()
MsqlRemMHook ()
MsqlSelectDB ()
MsqlShutdown ()
MsqlStoreResult ()
MsqlSumTimes ()
MsqlTimeToUnixTime ()
MsqlUnixTimeToDate ()
MsqlUnixTimeToTime ()

1.2 msql.library/--background--

The msql.library is an Amiga shared library that grant access ↔
to a mSQL
database engine over a TCP/IP network (ie include the mSQL client part).

```
MsqNumRows()  
&  
MsqNumFields()  
are macros #defined in libraries/msql.h
```

Please refer to the original documentation for more information.

News from Version 5.3:

- Include 2.0.7 code changes.

News from Version 5.2:

- Fix a bug that make MsqUnixTimeToDate unusable.
- Some code changes and bugs fix from the original API (V2.0.5 & 2.0.6)

News from Version 5:

- Some ARexx host bugs removed.
- ARexx allocation trace (Free all ARexx allocation with one function).
- Add a hook monitoring system.

News from Version 4:

- ARexx host.

News from Version 3:

- Support the final mSQL2 protocol.

News from Version 2:

- Client code included into the library (no external program required any more).
- Functions that don't require a "real" connection like
MsqDataSeek()
don't need a MsqConnection argument anymore (this ↔
argument was
needed with the previous version to exchange information with the
external program).

Requirement:

- A running TCP/IP stack (AmiTCP, Miami) to access distant DB server or mUSD to only access a local mSQL server.

Note:

- the ixemul.library is not used any more.
- Since new Times functions used static string in the original api, the library keeps buffers for each process that open the library. So, DON'T share the library base between process!

A large parts of this documentation comes from the original msql api documentation which is ©1998 Hughes Technologies Pty Ltd.

msql.library is (C) Copyright 1999 Christophe Sollet, All rights Reserved

1.3 msql.library/--rexxhost--

HOST INTERFACE (V4)

msql.library provides an ARExx function host interface that enables ARExx programs to access mSQL Database server. The functions provided by the interface are directly related to the functions described herein.

The function host library vector is located at offset -30 from the library. This is the value you provide to ARExx in the AddLib() function call.

FUNCTIONS

```

MsqlAllocConnection ()
MsqlClose (MSQLCONNECTION)
MsqlConnect (MSQLCONNECTION, STRING)
MsqlCreateDB (MSQLCONNECTION, STRING)
MsqlDataSeek (M_RESULT, INT)
MsqlDropDB (MSQLCONNECTION, STRING)
MsqlFetchField (M_RESULT)
MsqlFetchRow (M_RESULT)
MsqlFieldSeek (M_RESULT, INT)
MsqlFreeConnection (MSQLCONNECTION)
MsqlFreeResult (M_RESULT)
MsqlGetErrMsg (MSQLCONNECTION)
MsqlGetHostInfo (MSQLCONNECTION)
MsqlGetProtoInfo (MSQLCONNECTION)
MsqlGetServerInfo (MSQLCONNECTION)
MsqlListDBs (MSQLCONNECTION)
MsqlListFields (MSQLCONNECTION, STRING)
MsqlListIndex (MSQLCONNECTION, STRING, STRING)
MsqlListTables (MSQLCONNECTION)
MsqlLoadConfigFile (MSQLCONNECTION, STRING)
MsqlNumFields (M_RESULT)
MsqlNumRows (M_RESULT)
MsqlQuery (MSQLCONNECTION, STRING)
MsqlReloadAcls (MSQLCONNECTION)
MsqlSelectDB (MSQLCONNECTION, STRING)
MsqlShutdown (MSQLCONNECTION)
MsqlStoreResult (MSQLCONNECTION)

```

AREXX ONLY FUNCTIONS

```

MsqlGetField (M_ROW, POS)
    Get the field value at POS of a row

MsqlGetFieldInfo (M_FIELD, TYPE)
    Get info "TYPE" on a field
    TYPE can be: - "name"
                 - "table"
                 - "type"
                 - "length"

MsqlIsNotNull (M_FIELD)
MsqlIsUnique (M_FIELD)
    This two functions test flags of the field.

```


NOTES:

The following functions return true on success:

```

MsqlSelectDB
MsqlQuery
MsqlCreateDB
MsqlDropDB
MsqlGetProtoInfo
MsqlReloadAcls
MsqlDataSeek
MsqlFieldSeek
MsqlLoadConfigFile

```

1.4 msql.library/MsqlAddMHookA

NAME

```

MsqlAddMHookA -- add a library monitoring hook. (V5)
MsqlAddMHook -- Varargs stub for MsqlAddMHook. (V5)

```

SYNOPSIS

```

success = MsqlAddMHookA(hook, TagItems)
D0                A0    A1

BOOL MsqlAddMHookA(struct Hook *, struct TagItems *);

success = MsqlAddMHook(hook, Tag1, ... )

BOOL MsqlAddMHook(struct Hook *, ULONG, ... );

```

FUNCTION

This function adds a callback hook to monitor each library call. The hook will be called at each library function call and return.

Hooks are called with the following parameters:

- A0: struct Hook *: your struct Hook
- A2: APTR: your callback handle
- A1: struct HookMessage *: a pointer to an initialized struct HookMessage describing the called library function.

INPUTS

```

hook - callback hook
TagItems - none are defined for now, must be NULL

```

RESULT

```

success - TRUE on success

```

EXAMPLE

NOTES

The struct HookMessage is read-only!

BUGS

SEE ALSO

MsqlRemMHook()

1.5 msql.library/MsqlAllocConnection

NAME

MsqlAllocConnection -- Alloc a MsqlConnection structure

SYNOPSIS

```
mc = MsqlAllocConnection()  
D0
```

```
struct MsqlConnection *MsqlAllocConnection(void);
```

FUNCTION

Alloc an MsqlConnection structure used by all other function.
A MsqlConnection structure must be created by each task that
access the msql.library

INPUTS

none

RESULT

mc - A ready-to-use structure or NULL on error.

EXAMPLE

NOTES

You must use MsqlFreeConnection to free the returned structure.

BUGS

SEE ALSO

MsqlFreeConnection()

1.6 msql.library/MsqlClose

NAME

MsqlClose -- close a connection to the mSQL engine

SYNOPSIS

```
MsqlClose(mc)  
A1
```

```
void MsqlClose(struct MsqlConnection *);
```

FUNCTION

The connection to the mSQL engine can be closed using `msqlClose()`. The function must be called with the `MsqlConnection` structure returned by

```
MsqlConnect()
    when the initial connection was made.
```

INPUTS

`mc` - a "connected" `MsqlConnection`

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

```
MsqlConnect()
```

1.7 mysql.library/MsqlConnect

NAME

`MsqlConnect` -- Forms an interconnection with the mSQL engine

SYNOPSIS

```
mc = MsqlConnect(mc, host)
D0          A1  A0
```

```
struct MsqlConnection *
    MsqlConnect(struct MsqlConnection *, char *);
```

FUNCTION

`msqlConnect()` forms an interconnection with the mSQL engine. The `host` argument is the name or IP address of the host running the mSQL server. If `NULL` is specified as the `host` argument, a connection is made to a server running on the localhost using the UNIX domain socket `/dev/msqld`. If an error occurs, `NULL` is returned and the external variable `mssqlErrMsg` (returned by

```
MsqlGetErrMsg()
) will contain an appropriate text message.
```

If the connection is made to the server, the `MsqlConnection` is filled with connecton information.

INPUTS

`mc` - a `MsqlConnection` structure returned by `MsqlAllocConnection()`

host - the name or IP address of the host running the mSQL server ←

RESULT

mc - same as the input mc or NULL on error

EXAMPLE**NOTES****BUGS****SEE ALSO**

MsqlClose()

1.8 msql.library/MsqlCopyDB

NAME

MsqlCopyDB -- Undocumented (V3)

SYNOPSIS

```
error = MsqlCopyDB(mc, fromDB, toDB)
D0          A0 A1      A2
```

```
int MsqlCopyDB(struct MsqlConnection *, char *, char *);
```

FUNCTION

Undocumented.

INPUTS

mc - a "connected" MsqlConnection structure
fromDB - ? :)
toDB - ? :)

RESULT

error - -1 on error.

EXAMPLE**NOTES****BUGS****SEE ALSO**

MsqlMoveDB()

1.9 msql.library/MsqlCreateDB

NAME

MsqlCreateDB -- Create a new database

SYNOPSIS

```
error = MsqlCreateDB(mc, name)
D0          A1  A0
```

```
int MsqlCreateDB(struct MsqlConnection *, char *);
```

FUNCTION

Create a new database on the connected server

INPUTS

mc - a MsqlConnection
name - database name

RESULT

error - -1 on error

EXAMPLE

NOTES

It's an Admin function! This function isn't documented in the original API.

BUGS

SEE ALSO

1.10 msql.library/MsqlDataSeek

NAME

MsqlDataSeek -- Move the position of the data cursor

SYNOPSIS

```
MsqlDataSeek(result, pos)
          A0          D0
```

```
void MsqlDataSeek(m_result *, int);
```

FUNCTION

The m_result structure contains a client side "cursor" that holds information about the next row of data to be returned to the calling program. MsqlDataSeek() can be used to move the position of the data cursor. If it is called with a position of 0, the next call to

```
MsqlFetchRow()
```

will return the first row of data

returned by the server. The value of pos can be anywhere from 0 (the first row) and the number of rows in the table. If a seek is made past the end of the table, the next call to

```
MsqlFetchRow()
```

will return a NULL.

INPUTS

result - the m_result structure to seek
pos - the position (0 to number of rows)

RESULT
none

EXAMPLE

NOTES

BUGS

SEE ALSO

MsqlFetchRow()

1.11 msql.library/MsqlDateOffset

NAME

MsqlDateOffset -- Produce a relative date (V3)

SYNOPSIS

```
date = MsqlDateOffset(sdate, dOff, mOff, yOff)
D0          A0    D0    D1    D2
```

```
char *MsqlDateOffset(char *, int, int, int);
```

FUNCTION

The MsqlDateOffset() function allows you to generate an mSQL date string that is a specified period before or after a given date. This routine will determine the correct date based on the varying days of month. It is also aware of leap years and the impact they have on date ranges. The new date is calculated using the specified date and an offset value for the day, month and year. The example below would determine tomorrow's date

~

```
clock = time();
today = MsqlUnixTimeToDate(clock);
tomorrow = MsqlDateOffset( today , 1 , 0 , 0 );
```

INPUTS

```
sdate - starting date
dOff - day offset
mOff - month offset
yOff - year offset
```

RESULT

```
date - new date
```

EXAMPLE

NOTES

The returned string is statically declared in the API so you must make a copy of it before you call the function again. Of course, since msql.library is a shared library, each process have his own buffer.

BUGS

SEE ALSO

```
MsqldiffTimes()  
,  
MsqldiffTimes()  
,  
MsqldiffDates()
```

1.12 msql.library/MsqlDateToUnixTime

NAME

MsqldateToUnixTime -- Convert mSQL date to an unix time value (V3)

SYNOPSIS

```
time = MsqldateToUnixTime(date)  
D0                                   A0
```

```
time_t MsqldateToUnixTime(char *);
```

FUNCTION

MsqldateToUnixDate() converts an mSQL date format string into a UNIX time value. The mSQL date format is "DD-Mon-YYYY" (for example "12-Jun-1997") while the returned value will be the number of seconds since the UNIX epoch. The mSQL date routines will assume the 20th century if only 2 digits of the year value are presented. Although the valid range of mSQL dates is 31st Dec 4096bc to the 31st Dec 4096, the UNIX format cannot represent dates prior to the 1st Jan 1970.

INPUTS

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

```
MsqldateToUnixTime()  
,  
MsqldateToUnixTime()  
,  
MsqldateToUnixTime()
```

1.13 msql.library/MsqlDiffDates

NAME

MsqlDiffDates -- determine days between two dates. (V3)

SYNOPSIS

```
nbday = MsqlDiffDates(date1, date2)
D0                A0      A1
```

```
int MsqlDiffDates(char *, char *);
```

FUNCTION

The MsqlDiffDates() function can be used to determine the number of days between two dates. Date1 must be less than date2 and the two dates must be valid mSQL date formatted strings. In conjunction with the MsqlDiffTimes() function it is possible to determine a complete time difference between two pairs of times and dates.

INPUTS

date1 - a mSQL date formatted string.
date2 - another mSQL date formatted string.

RESULT

nbday - the difference between date1 & date2.

EXAMPLE

NOTES

BUGS

SEE ALSO

```
MsqlDateOffset()
,
MsqlSumTimes()
,
MsqlDiffTimes()
```

1.14 msql.library/MsqlDiffTimes

NAME

MsqlDiffTimes -- determine the time diff between time values (V3)

SYNOPSIS

```
time = MsqlDiffTimes(time1, time2)
D0                A0      A1
```

```
char *MsqlDiffTimes(char *, char *);
```

FUNCTION

To determine the time difference between two time values, the `MsqlDiffTimes()` function can be used. The two time values must be mSQL time formatted text strings and the returned value is also an mSQL time string. A restriction is placed on the times in that `time1` must be less than `time2`.

INPUTS

`time1` - a mSQL time formatted string.
`time2` - another mSQL time formatted string.

RESULT

`time` - the difference between `time1` & `time2`.

EXAMPLE

NOTES

The returned string is statically declared in the API so you must make a copy of it before you call the function again. Of course, since `msql.library` is a shared library, each process have his own buffer.

BUGS

SEE ALSO

```
MsqlSumTimes()  
,  
MsqlDateOffset()  
,  
MsqlDiffDates()
```

1.15 msql.library/MsqlDropDB

NAME

`MsqlDropDB` -- Drop a database

SYNOPSIS

```
error = MsqlDropDB(mc, name)  
D0          A1  A0  
  
int MsqlDropDB(struct MsqlConnection *, char *);
```

FUNCTION

Drop a database on the connected server

INPUTS

`mc` - a `MsqlConnection`
`name` - database name

RESULT

`error` - -1 on error

EXAMPLE

NOTES

It's an Admin function! This function isn't documented in the original API.

BUGS

SEE ALSO

1.16 msql.library/MsqlFetchField

NAME

MsqlFetchField -- Get information about the data fields selected

SYNOPSIS

```
field = MsqlFetchField(result)
D0          A0
```

```
m_field *MsqlFetchField(m_result *);
```

FUNCTION

Along with the actual data rows, the server returns information about the data fields selected. This information is made available to the calling program via the MsqlFetchField() function. Like

```
MsqlFetchRow()
```

```
, this function returns one element
```

of information at a time and returns NULL when no further information is available. The data is returned in a m_field structure which contains the following information:

```
typedef struct
{
    char    *name,        // name of field
           *table;       // name of table
    int     type,         // data type of field
           length,       // length in bytes of field
           flags;        // attribute flags
} m_field;
```

Possible values for the type field are defined in msql.h. Please consult the header file if you wish to interpret the value of the type or flags field of the m_field structure.

INPUTS

result - a previously returned result structure

RESULT

field - data fields information or NULL when no further information is available

EXAMPLE

NOTES

BUGS

SEE ALSO

`MsqlFetchRow()`

1.17 msql.library/MsqlFetchRow

NAME

`MsqlFetchRow` -- Access individual db rows returned by a select

SYNOPSIS

```
row = MsqlFetchRow(result)
D0          A0

m_row MsqlFetchRow(m_result *);
```

FUNCTION

The individual database rows returned by a select are accessed via the `MsqlFetchRow()` function. The data is returned in a variable of type `m_row` which contains a char pointer for each field in the row. For example, if a select statement selected 3 fields from each row returned, the value of the 3 fields would be assigned to elements [0], [1], and [2] of the variable returned by `MsqlFetchRow()`.

INPUTS

`result` - the data to fetch

RESULT

`row` - a row structure or NULL when the end of the data has been reached

EXAMPLE

NOTES

A NULL value is represented as a NULL pointer in the row.

BUGS

SEE ALSO

1.18 msql.library/MsqlFieldSeek

NAME

`MsqlFieldSeek` -- Move the field data cursor

SYNOPSIS

```
MsqlFieldSeek(result, pos)
          A0          D0
```

```
void MsqlFieldSeek(m_result *, int);
```

FUNCTION

The result structure includes a "cursor" for the field data. It's position can be moved using the MsqlFieldSeek() function. See

```
    MsqlDataSeek()
    for further details.
```

INPUTS

result - the m_result structure to seek
pos - position to move

RESULT

none

EXAMPLE

NOTES

BUGS

SEE ALSO

```
    MsqlDataSeek()
```

1.19 msql.library/MsqlFreeConnection

```
NAME
MsqlFreeConnection -- Free a MsqlConnection structure
```

SYNOPSIS

```
MsqlFreeConnection(mc)
    A0
```

```
void MsqlFreeConnection(stuct MsqlConnection *);
```

FUNCTION

Free a MsqlConnection structure returned by MsqlAllocConnection.

INPUTS

mc - a MsqlConnection

RESULT

none

EXAMPLE

NOTES

BUGS

SEE ALSO

```
    MsqlAllocConnection()
```

1.20 mysql.library/MsqlFreeResult

NAME
MsqlFreeResult -- Free a query result

SYNOPSIS
MsqlFreeResult(result)
A0

```
void MsqlFreeResult(m_result *);
```

FUNCTION
When a program no longer requires the data associated with a particular query result, the data must be freed using MsqlFreeResult(). The result handle associated with the data, as returned by MsqlStoreResult() is passed to MsqlFreeResult() to identify the data set to be freed.

INPUTS
result - a m_result structure returned by MsqlStoreResult()
RESULT
none

EXAMPLE

NOTES

BUGS

SEE ALSO
MsqlStoreResult()

1.21 mysql.library/MsqlGetCharConf

NAME
MsqlGetCharConf -- Undocumented (used by msqldadmin ?) (V3)

SYNOPSIS
x = MsqlGetCharConf(mc, y, z)
D0 A1 A0 A2

```
int MsqlGetCharConf(struct MsqlConnection *, char *, char *);
```

FUNCTION
Undocumented.

INPUTS

mc - a MySqlConnection
y - ???
z - ???

RESULT

x - ???

EXAMPLE

NOTES

This is a private Msql API function. No information was given about it.

This function is not part of the mSQL API. Any use of this function is discouraged as the interface may change in future releases

BUGS

SEE ALSO

1.22 mysql.library/MsqlGetErrMsg

NAME

MsqlGetErrMsg -- Get an error message

SYNOPSIS

```
errmsg = MsqlGetErrMsg(mc)
D0          A0
```

```
char *MsqlGetErrMsg(struct MySqlConnection *);
```

FUNCTION

If a mysql function failed, an error message will be stored in an internal buffer. This function return a pointer on this buffer.

INPUTS

mc - a valid MySqlConnection structure

RESULT

errmsg - a null terminated string describing a previous error

EXAMPLE

NOTES

There is no guarantee as to the value returned from MsqlGetErrMsg() after a successful operation.

BUGS

SEE ALSO

1.23 msql.library/MsqlGetHostInfo

NAME

MsqlGetHostInfo -- Undocumented

SYNOPSIS

```
x = MsqlGetHostInfo(mc)
D0                               A0
```

```
char *MsqlGetHostInfo(struct MsqlConnection *);
```

FUNCTION

Undocumented.

INPUTS

mc - a MsqlConnection

RESULT

x - ???

EXAMPLE

NOTES

This is a private Msql API function. No information was given about it.

BUGS

SEE ALSO

1.24 msql.library/MsqlGetIntConf

NAME

MsqlGetIntConf -- Undocumented (used by msqladmin ?) (V3)

SYNOPSIS

```
x = MsqlGetIntConf(mc, y, z)
D0                               A1 A0 A2
```

```
int MsqlGetIntConf(struct MsqlConnection *, char *, char *);
```

FUNCTION

Undocumented.

INPUTS

mc - a MsqlConnection
y - ???
z - ???

RESULT

x - ???

EXAMPLE

NOTES

This is a private Msql API function. No information was given about it.

This function is not part of the mSQL API. Any use of this function is discouraged as the interface may change in future releases

BUGS

SEE ALSO

1.25 mysql.library/MsqlGetProtoInfo

NAME

MsqlGetProtoInfo -- Undocumented

SYNOPSIS

```
x = MsqlGetProtoInfo(mc)
D0                               A0

int MsqlGetProtoInfo(struct MsqlConnection *);
```

FUNCTION

Undocumented.

INPUTS

mc - a MsqlConnection

RESULT

x - ???

EXAMPLE

NOTES

This is a private Msql API function. No information was given about it.

BUGS

SEE ALSO

1.26 mysql.library/MsqlGetSequenceInfo

NAME

MsqlGetSequenceInfo -- (V3)

SYNOPSIS

```
seq = MsqlGetSequenceInfo(mc, table)
D0                               A0 A1

m_seq *MsqlGetSequenceInfo(struct MsqlConnection *, char *);
```


FUNCTION

INPUTS

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

1.27 msql.library/MsqlGetServerInfo

NAME

MsqlGetServerInfo -- Undocumented

SYNOPSIS

```
x = MsqlGetServerInfo(mc)
D0                               A0
```

```
char *MsqlGetServerInfo(struct MsqlConnection *);
```

FUNCTION

Undocumented.

INPUTS

mc - a MsqlConnection

RESULT

x - ???

EXAMPLE

NOTES

This is a private Msql API function. No information was given about it.

BUGS

SEE ALSO

1.28 msql.library/MsqlGetServerStats

NAME

MsqlGetServerStats -- Private (V3)

SYNOPSIS

```
error = MsqlGetServerStats(mc, buffer, size)
```

```

D0                                A0 A1      D0

int MsqlGetServerStats(struct MsqlConnection *, char *, ULONG);

FUNCTION
    Undocumented.

INPUTS
    mc - a "connected" MsqlConnection structure
    buffer - output buffer
    size - size of buffer

RESULT
    error - -1 on error

EXAMPLE

NOTES
    Original API writes to the standard output, not in a buffer.

BUGS

SEE ALSO

```

1.29 mysql.library/MsqlListDBs

```

NAME
MsqlListDBs -- return a list of existing database

SYNOPSIS
    result = MsqlListDBs(mc)
D0                                A0

    m_result *MsqlListDBs(struct MsqlConnection *);

FUNCTION
    A list of the databases known to the mSQL engine can be obtained
    via the MsqlListDBs() function. A result handle is returned to
    the calling program that can be used to access the actual
    database names. The individual names are accessed by calling

        MsqlFetchRow()
        passing it the result handle. The m_row data
        structure returned by each call will contain one field being the
        name of one of the available databases. As with all functions
        that return a result handle, the data associated with the result
        must be freed when it is no longer required using
        MsqlFreeResult()
        .

INPUTS
    mc - a "connected" MsqlConnection

RESULT
    result - data containing the list of known dbs.

EXAMPLE

```

NOTES

BUGS

SEE ALSO

```

MsqlFetchRow()
,
MsqlFreeResult()
,
MsqlListTables()

```

1.30 msql.library/MsqlListFields

NAME

MsqlListFields -- Get information about table fields

SYNOPSIS

```

result = MsqlListFields(mc, tableName)
D0          A0 A1

```

```

m_result *MsqlListFields(struct MsqlConnection *, char *);

```

FUNCTION

Information about the fields in a particular table can be obtained using MsqlListFields(). The function is called with the name of a table in the current database as selected using

```

MsqlSelectDB()
and a result handle is returned to the caller.

```

Unlike

```

MsqlListDBs()
and
MsqlListTables()
, the field information

```

is contained in field structures rather than data rows. It is accessed using

```

MsqlFetchField()
. The result handle must be freed
when it is no longer needed by calling
MsqlFreeResult()
.

```

INPUTS

```

mc - a "connected" MsqlConnection
tableName - a null terminated string containing the name of the
table

```

RESULT

```

result - data about the table structure

```

EXAMPLE

NOTES

BUGS

SEE ALSO

```

MsqlSelectDB()
,
MsqlFetchField()
,
MsqlFreeResult()

```

1.31 msql.library/MsqlListIndex

NAME

MsqlListIndex -- Get the structure of a table index

SYNOPSIS

```

result = MsqlListIndex(mc, tableName, index)
D0                A2  A0                A1

```

```

m_result *MsqlListIndex(struct MsqlConnection *, char *, char *);

```

FUNCTION

The structure of a table index can be obtained from the server using the MsqlListIndex() function. The result table returned contains one field.

The first row of the result contains the symbolic name of the index mechanism used to store the index. Rows 2 and onwards contain the name of the fields that comprise the index.

For example, if a compound index was defined as an AVL Tree index and was based on the values of the fields first_name and last_name, then the result table would look like:

```

-----
|   row[0]   |
|-----|
|   avl     |
|-----|
| first_name |
|-----|
| last_name  |
-----

```

Currently the only valid index type is 'avl' signifying a memory mapped AVL tree.

INPUTS

mc - a "connected" MsqlConnection

tableName - a null terminated string containing the name of the table

index - a null terminated string containing the name of the index

RESULT

result - index information

EXAMPLE

NOTES

BUGS

SEE ALSO

`MsqlFreeResult()`

1.32 msql.library/MsqlListTables

NAME

`MsqlListTables` -- return a table list of selected database

SYNOPSIS

```
result = MsqlListTables(mc)
D0          A0
```

```
m_result *MsqlListTables(struct MsqlConnection *);
```

FUNCTION

Once a database has been selected using `MsqlSelectDB()`, a list of the tables defined in that database can be retrieved using `MsqlListTables()`. As with `MsqlListDBs()`, a result handle is returned to the calling program and the names of the tables are contained in data rows where element [0] of the row is the name of one table in the current database. The result handle must be freed when it is no longer needed by calling `MsqlFreeResult()`.

INPUTS

`mc` - a "connected" `MsqlConnection`

RESULT

`result` - data containing a list of tables

EXAMPLE

NOTES

BUGS

SEE ALSO

```
MsqlFetchRow()
,
MsqlFreeResult()
,
MsqlSelectDB()
,
```

MsqlListDBs()

1.33 mysql.library/MsqlLoadConfigFile

NAME

MsqlLoadConfigFile -- Load a non-default configuration

SYNOPSIS

```
error = MsqlLoadConfigFile(mc, file)
D0                A1  A0
```

```
int MsqlLoadConfigFile(struct MsqlConnection *, char *);
```

FUNCTION

The MsqlLoadConfigFile() function can be used to load a non-default configuration file into your client application. The configuration file can include information such as the TCP/IP and UNIX ports on which the desired mSQL server will be running. The file to be loaded is determined by the value of the file parameter. If the value of the parameter is new, the MsqlLoadConfigFile() function would search for the file in the following places (and in the order specified).

~

```
Inst_Dir/new
Inst_Dir/new.conf
new
```

~

That is, if a file called "new" exists in the installation directory, it is loaded. Otherwise, an attempt will be made to load a file called new.conf from the installation directory. If that fails, the filename specified is assumed to be a complete, absolute pathname and an attempt to open the file is made.

INPUTS

```
mc - a MsqlConnection
file - a configuration file
```

RESULT

error - 1 on failure, otherwise a value of 0 is returned

EXAMPLE

NOTES

BUGS

SEE ALSO

1.34 mysql.library/MsqlMoveDB

NAME

MsqlMoveDB -- Undocumented (V3)

SYNOPSIS

```
x = MsqlMoveDB(mc, fromDB, toDB)
D0                A0 A1    A2
```

```
int MsqlMoveDB(struct MsqlConnection *, char *, char *);
```

FUNCTION

Undocumented.

INPUTS

mc - a "connected" MsqlConnection structure.
 fromDB - ?
 toDB - ?

RESULT

x - ?

EXAMPLE

NOTES

BUGS

SEE ALSO

MsqlCopyDB()

1.35 msql.library/MsqlNumFields

NAME

MsqlNumFields -- Get the number of fields of a row

SYNOPSIS

```
num = MsqlNumFields(result)
```

```
int MsqlNumFields(m_result *);
```

FUNCTION

The number of fields returned by a query can be ascertained by calling MsqlNumFields() and passing it the result handle. The value returned by MsqlNumFields() indicates the number of elements in the data vector returned by

```
MsqlFetchRow()
```

. It is

wise to check the number of fields returned before, as with all arrays, accessing an element that is beyond the end of the data vector can result in a segmentation fault (crash).

INPUTS

result - a m_result data structure

RESULT
num - the number of fields.

EXAMPLE

NOTES
This function is not part of the mysql.library but was defined (#define) in mysql/mysql.h

BUGS

SEE ALSO

1.36 mysql.library/MsqlNumRows

NAME
MsqlNumRows -- Get the number of rows of data

SYNOPSIS
num = MsqlNumRows(result)

```
int MsqlNumRows(m_result *);
```

FUNCTION
The number of rows returned by a query can be found by calling MsqlNumRows() and passing it the result handle returned by MsqlStoreResult(). The number of rows of data sent as a result of the query is returned as an integer value. If a select query didn't match any data, MsqlNumRows() will indicate that the result table has 0 rows (note: earlier versions of mSQL returned a NULL result handle if no data was found. This has been simplified and made more intuitive by returning a result handle with 0 rows of result data)

INPUTS
result - a m_result data structure

RESULT
num - the number of rows.

EXAMPLE

NOTES
This function is not part of the mysql.library but was defined (#define) in mysql/mysql.h

BUGS

SEE ALSO

1.37 mysql.library/MsqlQuery

NAME

MsqlQuery -- send a sql query to the mSQL engine

SYNOPSIS

```
error = MsqlQuery(mc, query)
D0                A1  A0
```

```
int MsqlQuery(struct MsqlConnection *, char *);
```

FUNCTION

A query in SQL terminology is not the same as a query in the English language. In English, the word query relates to asking a question whereas in SQL a query is a valid SQL command. It is a common mistake that people believe that the msqlQuery function can only be used to submit SELECT commands to the database engine. In reality, msqlQuery can be used for any valid mSQL command including SELECT, DELETE, UPDATE etc.

Queries are sent to the engine over the connection associated with mc as plain text strings using MsqlQuery(). As usual, a returned value of -1 indicates an error and msqlErrMsg will be updated.

If the query generates output from the engine, such as a SELECT statement, the data is buffered in the API waiting for the application to retrieve it. If the application submits another query before it retrieves the data using msqlStoreResult(), the buffer will be overwritten by any data generated by the new query.

In previous versions of mSQL, the return value of msqlQuery() was either -1 (indicating an error) or 0 (indicating success). mSQL2 adds to these semantics by providing more information back to the client application via the return code. If the return code is greater than 0, not only does it imply success, it also indicates the number of rows "touched" by the query (i.e. the number of rows returned by a SELECT, the number of rows modified by an update, or the number of rows removed by a delete).

INPUTS

mc - a "connected" MsqlConnection.

query - a SQL query.

RESULT

error - == -1 on error.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.38 msql.library/MsqlReloadAcls

NAME

MsqlReloadAcls -- Force server to reload access list

SYNOPSIS

```
x = MsqlReloadAcls(mc)
D0                          A0
```

```
int MsqlReloadAcls(struct MsqlConnection *);
```

FUNCTION

Force the server to reload the access list

INPUTS

mc - a MsqlConnection

RESULT

error - -1 on error

EXAMPLE

NOTES

It's an Admin function! This function isn't documented in the original API.

BUGS

SEE ALSO

1.39 msql.library/MsqlRemMHook

NAME

MsqlRemMHook -- remove a library monitoring hook. (V5)

SYNOPSIS

```
MsqlRemMHook(hook)
                          A0
```

```
void MsqlRemMHook(struct Hook *);
```

FUNCTION

Remove a callback hook previously installed by MsqlAddMHookA()

INPUTS

hook - an installed callback hook

RESULT

none

EXAMPLE

NOTES

You have to call `MsqlMRemHook` for each hook you have installed before closing the `msql.library`.

BUGS

SEE ALSO

`MsqlAddMHookA()`

1.40 msql.library/MsqlSelectDB

NAME

`MsqlSelectDB` -- instructs engine which database is to be accessed

SYNOPSIS

```
error = MsqlSelectDB(mc, dbName)
D0                A1  A0
```

```
int MsqlSelectDB(struct MsqlConnection *, char *);
```

FUNCTION

Prior to submitting any queries, a database must be selected. `mSQLSelectDB()` instructs the engine which database is to be accessed. `mSQLSelectDB()` is called with the `MsqlConnection` returned by

```
MsqlConnect()
```

and the name of the desired database.

A return value of `-1` indicates an error with `mSQLErrMsg` set to a text string representing the error. `MsqlSelectDB()` may be called multiple times during a program's execution. Each time it is called, the server will use the specified database for future accesses. By calling `mSQLSelectDB()` multiple times, a program can switch between different databases during its execution.

INPUTS

`mc` - a "connected" `MsqlConnection`.

`dbName` - the name of the database to select.

RESULT

`error` - `== -1` on error.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.41 mysql.library/MsqlShutdown

NAME

MsqlShutdown -- Shutdown a mSQL server

SYNOPSIS

```
x = MsqlShutdown(mc)
D0                      A0
```

```
int MsqlShutdown(struct MsqlConnection *);
```

FUNCTION

Shutdown the connected server

INPUTS

mc - a MsqlConnection

RESULT

error - -1 on error

EXAMPLE

NOTES

It's an Admin function! This function isn't documented in the original API.

BUGS

SEE ALSO

1.42 mysql.library/MsqlStoreResult

NAME

MsqlStoreResult -- Store a query result

SYNOPSIS

```
result = MsqlStoreResult(mc)
D0                      A0
```

```
m_result *MsqlStoreResult(struct MsqlConnection *);
```

FUNCTION

Data returned by a SELECT query must be stored before another query is submitted or it will be removed from the internal API buffers. Data is stored using the MsqlStoreResult() function which returns a result handle to the calling routines. The result handle is a pointer to a m_result structure and is passed to other API routines when access to the data is required. Once the result handle is allocated, other queries may be submitted. A program may have many result handles active simultaneously.

INPUTS

mc - a MsqlConnection

RESULT
result - result handle of the previous request

EXAMPLE

NOTES

BUGS

SEE ALSO

MsqFreeResult()

1.43 msql.library/MsqlSumTimes

NAME

MsqSumTimes -- Sum two mSQL time (V3)

SYNOPSIS

```
time = MsqSumTimes(time1, time2)
D0                A0    A1
```

```
char *MsqSumTimes(char *, char *);
```

FUNCTION

The MsqSumTimes() routine provides a mechanism for performing addition between two mSQL time formatted strings. A literal addition of the values is returned to the calling routine in mSQL time format. As an example, calling MsqSumTimes with the values "1:30:25" and "13:15:40" would return "14:46:05".

INPUTS

time1 - a mSQL time formatted string.
time2 - another mSQL time formatted string.

RESULT

time - a literal addition of time1 & time2.

EXAMPLE

NOTES

The returned string is statically declared in the API so you must make a copy of it before you call the function again. Of course, since msql.library is a shared library, each process have his own buffer.

BUGS

SEE ALSO

```
MsqDiffTimes()
,
MsqDateOffset()
,
```

MsqldiffDates()

1.44 msql.library/MsqlTimeToUnixTime

NAME

MsqlTimeToUnixTime -- Convert mSQL time to unix time (V3)

SYNOPSIS

```
time = MsqlTimeToUnixTime(date)
D0                                     A0
```

```
time_t MsqlTimeToUnixTime(char *);
```

FUNCTION

MsqlTimeToUnixTime() converts an mSQL time value to a standard UNIX time value. The mSQL time value must be a character string in the 24 hour format of "HH:MM:SS" and the returned value will be the number of seconds since 1 Jan 1970 (the normal UNIX format).

INPUTS

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

```
MsqlUnixTimeToDate()
,
MsqlUnixTimeToTime()
,
MsqlDateToUnixTime()
```

1.45 msql.library/MsqlUnixTimeToDate

NAME

MsqlUnixTimeToDate -- Convert Unix time to mSQL date string (V3)

SYNOPSIS

```
date = MsqlUnixTimeToDate(clock)
D0                                     D0
```

```
char *MsqlUnixTimeToDate(time_t);
```

FUNCTION

MsqlUnixTimeToDate() converts a standard UNIX time value to an

mSQL date string. The time value is specified as seconds since the UNIX epoch (1st Jan 1970) while the mSQL date string will contain the date formatted as "DD-Mon-YYYY" (e.g. "12-Jun-1997"). The returned string is statically declared in the API so you must make a copy of it before you call the function again.

INPUTS

clock - the time to convert

RESULT

date - the clock "value" in a mSQL date format

EXAMPLE

NOTES

The returned string is statically declared in the API so you must make a copy of it before you call the function again. Of course, since `msql.library` is a shared library, each process has his own buffer.

BUGS

SEE ALSO

```

MsqlTimeToUnixTime()
,
MsqlUnixTimeToTime()
,
MsqlDateToUnixTime()

```

1.46 msql.library/MsqlUnixTimeToTime

NAME

MsqlUnixTimeToTime -- Convert Unix time to mSQL time format (V3)

SYNOPSIS

```

time = MsqlUnixTimeToTime(clock)
D0                                     D0

```

```

char *MsqlUnixTimeToTime(time_t);

```

FUNCTION

MsqlUnixTimeToTime() converts a UNIX time value (seconds since the UNIX epoch) into a character string representing the same time in mSQL time format (i.e. "HH:MM:SS" 24 hour format).

INPUTS

clock - the time to convert

RESULT

time - the clock value in a mSQL time string format.

EXAMPLE

NOTES

The returned string is statically declared in the API so you must make a copy of it before you call the function again. Of course, since `msql.library` is a shared library, each process have his own buffer.

BUGS

SEE ALSO

```
MsqUnixTimeToDate()  
,  
MsqTimeToUnixTime()  
,  
MsqDateToUnixTime()
```