Informix

Introduction

The Informix driver for Informix (IDS) 7.x, SE 7.x, Universal Server (IUS) 9.x and IDS 2000 is implemented in "ifx.ec" and "php_informix.h" in the informix extension directory. IDS 7.x support is fairly complete, with full support for BYTE and TEXT columns. IUS 9.x support is partly finished: the new data types are there, but SLOB and CLOB support is still under construction.

Note
This extension has been moved to the » PECL repository and is no longer bundled with PHP as of PHP 5.2.1.

Installing/Configuring

Requirements

Note
Configuration notes You need a version of ESQL/C to compile the PHP Informix driver. ESQL/C versions from 7.2x on should be OK. ESQL/C is now part of the Informix Client SDK. Make sure that the "INFORMIXDIR" variable has been set, and that $INFORMIXDIR/bin is in your PATH before you run the "configure" script.

Installation

To be able to use the functions defined in this module you must compile your PHP interpreter using the configure line *--with-informix[=DIR]*, where DIR is the Informix base install directory, defaults to nothing.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

Note
Make sure that the Informix environment variables INFORMIXDIR and INFORMIXSERVER are available to the PHP ifx driver, and that the INFORMIX bin directory is in the PATH. Check this by running a script that contains a call to phpinfo() before you start testing. The phpinfo() output should list these environment variables. This is true for both CGI php and Apache mod_php. You may have to set these environment variables in your Apache startup script. The Informix shared libraries should also be available to the loader (check LD_LIBRARY_PATH or ld.so.conf/ldconfig).

Note
Some notes on the use of BLOBs (TEXT and BYTE columns) BLOBs are normally addressed by BLOB identifiers. Select queries return a "blob id"

for every BYTE and TEXT column. You can get at the contents with "string_var = ifx_get_blob($blob_id);" if you choose to get the BLOBs in memory (with: "ifx_blobinfile(0);"). If you prefer to receive the content of BLOB columns in a file, use "ifx_blobinfile(1);", and "ifx_get_blob($blob_id);" will get you the filename. Use normal file I/O to get at the blob contents.

For insert/update queries you must create these "blob id's" yourself with " ifx_create_blob();". You then plug the blob id's into an array, and replace the blob columns with a question mark (?) in the query string. For updates/inserts, you are responsible for setting the blob contents with ifx_update_blob().

The behaviour of BLOB columns can be altered by configuration variables that also can be set at runtime:

configuration variable: ifx.textasvarchar

configuration variable: ifx.byteasvarchar

runtime functions:

ifx_textasvarchar(0): use blob id's for select queries with TEXT columns

ifx_byteasvarchar(0): use blob id's for select queries with BYTE columns

ifx_textasvarchar(1): return TEXT columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

ifx_byteasvarchar(1): return BYTE columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

configuration variable: ifx.blobinfile

runtime function:

ifx_blobinfile_mode(0): return BYTE columns in memory, the blob id lets you get at the contents.

ifx_blobinfile_mode(1): return BYTE columns in a file, the blob id lets you get at the file name.

If you set ifx_text/byteasvarchar to 1, you can use TEXT and BYTE columns in select queries just like normal (but rather long) VARCHAR fields. Since all strings are "counted" in PHP, this remains "binary safe". It is up to you to handle this correctly. The returned data can contain anything, you are responsible for the contents.

If you set ifx_blobinfile to 1, use the file name returned by ifx_get_blob(..) to get at the blob contents. Note that in this case YOU ARE RESPONSIBLE FOR DELETING THE TEMPORARY FILES CREATED BY INFORMIX when fetching the row. Every new row fetched will create new temporary files for every BYTE column.

The location of the temporary files can be influenced by the environment variable "blobdir", default is "." (the current directory). Something like: putenv(blobdir=tmpblob");

will ease the cleaning up of temp files accidentally left behind (their names all start with "blb").

Note

Automatically trimming "char" (SQLCHAR and SQLNCHAR) data

This can be set with the configuration variable

ifx.charasvarchar: if set to 1 trailing spaces will be automatically trimmed, to save you some "chopping".

Note

NULL values

The configuration variable ifx.nullformat (and the runtime function ifx_nullformat()) when set to **TRUE** will return **NULL** columns as the string " **NULL** ", when set to **FALSE** they return the empty string. This allows you to discriminate between **NULL** columns and empty columns.

Informix configuration options

Name	Default	Changeable	Changelog
ifx.allow_persistent	"1"	PHP_INI_SYSTEM	Removed in PHP 5.2.1.
ifx.max_persistent	"-1"	PHP_INI_SYSTEM	Removed in PHP 5.2.1.
ifx.max_links	"-1"	PHP_INI_SYSTEM	Removed in PHP 5.2.1.
ifx.default_host	NULL	PHP_INI_SYSTEM	Removed in PHP 5.2.1.
ifx.default_user	NULL	PHP_INI_SYSTEM	Removed in PHP 5.2.1.
ifx.default_password	NULL	PHP_INI_SYSTEM	Removed in PHP 5.2.1.
ifx.blobinfile	"1"	PHP_INI_ALL	Removed in PHP 5.2.1.

ifx.textasvarchar	"0"	PHP_INI_ALL	Removed in PHP 5.2.1.
ifx.byteasvarchar	"0"	PHP_INI_ALL	Removed in PHP 5.2.1.
ifx.charasvarchar	"0"	PHP_INI_ALL	Removed in PHP 5.2.1.
ifx.nullformat	"0"	PHP_INI_ALL	Removed in PHP 5.2.1.

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

ifx.allow_persistent boolean
Whether to allow persistent Informix connections.

ifx.max_persistent integer
The maximum number of persistent Informix connections per process.

ifx.max_links integer
The maximum number of Informix connections per process, including persistent connections.

ifx.default_host string
The default host to connect to when no host is specified in ifx_connect() or ifx_pconnect(). Doesn't apply in safe mode.

ifx.default_user string
The default user id to use when none is specified in ifx_connect() or ifx_pconnect(). Doesn't apply in safe mode.

ifx.default_password string
The default password to use when none is specified in ifx_connect() or ifx_pconnect(). Doesn't apply in safe mode.

ifx.blobinfile boolean
Set to **TRUE** if you want to return blob columns in a file, **FALSE** if you want them in memory. You can override the setting at runtime with ifx_blobinfile_mode().

ifx.textasvarchar boolean
Set to **TRUE** if you want to return TEXT columns as normal strings in select statements, **FALSE** if you want to use blob id parameters. You can override the setting at runtime with ifx_textasvarchar().

ifx.byteasvarchar boolean
Set to **TRUE** if you want to return BYTE columns as normal strings in select queries, **FALSE** if you want to use blob id parameters. You can override the setting at runtime with ifx_textasvarchar().

ifx.charasvarchar boolean
: Set to **TRUE** if you want to trim trailing spaces from CHAR columns when fetching them.

ifx.nullformat boolean
: Set to **TRUE** if you want to return **NULL** columns as the literal string " **NULL** ", **FALSE** if you want them returned as the empty string "". You can override this setting at runtime with ifx_nullformat().

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

IFX_SCROLL (integer)

IFX_HOLD (integer)

IFX_LO_RDONLY (integer)

IFX_LO_WRONLY (integer)

IFX_LO_APPEND (integer)

IFX_LO_RDWR (integer)

IFX_LO_BUFFER (integer)

IFX_LO_NOBUFFER (integer)

Informix Functions

ifx_affected_rows

ifx_affected_rows -- Get number of rows affected by a query

Description

int **ifx_affected_rows** (resource `$result_id`)

Returns the number of rows affected by a query associated with *`result_id`*.

For inserts, updates and deletes the number is the real number (sqlerrd[2]) of affected rows. For selects it is an estimate (sqlerrd[0]). Don't rely on it. The database server can never return the actual number of rows that will be returned by a SELECT because it has not even begun fetching them at this stage (just after the "PREPARE" when the optimizer has determined the query plan).

Useful after ifx_prepare() to limit queries to reasonable result sets.

Parameters

`result_id`
A valid result id returned by ifx_query() or ifx_prepare().

Return Values

Returns the number of rows as an integer.

Examples

Example #1 - Informix affected rows

```
<?php
$rid = ifx_prepare("select * from emp
                    where name like " . $name, $connid);
if (! $rid) {
   /* ... error ... */
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
   printf ("Too many rows in result set (%d)\n<br />", $rowcount);
   die ("Please restrict your query<br />\n");
}
?>
```

See Also

- ifx_num_rows()

ifx_blobinfile_mode

ifx_blobinfile_mode -- Set the default blob mode for all select queries

Description

bool **ifx_blobinfile_mode** (int `$mode`)

Set the default blob mode for all select queries.

Parameters

`mode`
: Mode "0" means save Byte-Blobs in memory, and mode "1" means save Byte-Blobs in a file.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ifx_byteasvarchar

ifx_byteasvarchar -- Set the default byte mode

Description

bool **ifx_byteasvarchar** (int `$mode`)

Sets the default byte mode for all select-queries.

Parameters

`mode`
: Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ifx_textasvarchar()

ifx_close

ifx_close -- Close Informix connection

Description

bool **ifx_close** ([resource `$link_identifier`])

ifx_close() closes the link to an Informix database that's associated with the specified link identifier.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

ifx_close() will not close persistent links generated by ifx_pconnect().

Parameters

`link_identifier`
The link identifier. If not specified, the last opened link is assumed.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #2 - Closing a Informix connection

```
<?php
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");
/* ... some queries and stuff ... */
ifx_close($conn_id);
?>
```

See Also

- ifx_connect()
- ifx_pconnect()

ifx_connect

ifx_connect -- Open Informix server connection

Description

resource **ifx_connect** ([string `$database` [, string `$userid` [, string `$password`]]])

ifx_connect() establishes a connection to an Informix server.

In case a second call is made to ifx_connect() with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling ifx_close().

Parameters

All of the arguments are optional, and if they're missing, defaults are taken from values supplied in *php.ini* (ifx.default_host for the host (Informix libraries will use INFORMIXSERVER environment value if not defined), ifx.default_user for user, ifx.default_password for the password (none if not defined).

`database`
The database name, as a string.

`userid`
The username, as a string.

`password`
The password, as a string.

Return Values

Returns a connection identifier on success, or **FALSE** on error.

Examples

Example #3 - Connect to a Informix database

```
<?php
$conn_id = ifx_connect ("mydb@ol_srv1", "imyself", "mypassword");
?>
```

See Also

- ifx_pconnect()
- ifx_close()

ifx_copy_blob

ifx_copy_blob -- Duplicates the given blob object

Description

int **ifx_copy_blob** (int `$bid`)

Duplicates the given blob object.

Parameters

`bid`
A BLOB identifier.

Return Values

Returns the new blob object-id, or **FALSE** on errors.

See Also

- ifx_create_blob()
- ifx_free_blob()

ifx_create_blob

ifx_create_blob -- Creates an blob object

Description

int **ifx_create_blob** (int `$type`, int `$mode`, string `$param`)

Creates a blob object.

Parameters

`type`
1 = TEXT, 0 = BYTE

`mode`
0 = blob-object holds the content in memory, 1 = blob-object holds the content in file.

`param`
if mode = 0: pointer to the content, if mode = 1: pointer to the filestring.

Return Values

Returns the new BLOB object-id, or **FALSE** on errors.

See Also

- ifx_copy_blob()
- ifx_free_blob()

ifx_create_char

ifx_create_char -- Creates an char object

Description

int **ifx_create_char** (string `$param`)

Creates an char object.

Parameters

`param`
The char content.

Return Values

Returns the new char object id, or **FALSE** on errors.

See Also

- ifx_free_char()

ifx_do

ifx_do -- Execute a previously prepared SQL-statement

Description

bool **ifx_do** (resource `$result_id`)

Executes a previously prepared query or opens a cursor for it.

Does NOT free *`result_id`* on error.

Also sets the real number of ifx_affected_rows() for non-select statements for retrieval by ifx_affected_rows().

Parameters

`result_id`

`result_id` is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

Example #4 - ifx_do() Example

```
<?php
$conn = fx_connect( "db", "user", "password" );
$result = ifx_prepare("SELECT customer_num, company FROM customer", $conn);
ifx_do($result);
?>
```

See Also

- ifx_prepare()

ifx_error

ifx_error -- Returns error code of last Informix call

Description

string **ifx_error** ([resource `$link_identifier`])

Returns in a string one character describing the general results of a statement and both SQLSTATE and SQLCODE associated with the most recent SQL statement executed.

Parameters

`link_identifier`
The link identifier.

Return Values

The Informix error codes (SQLSTATE & SQLCODE) formatted as *x [SQLSTATE = aa bbb SQLCODE=cccc]*.

where x = space : no error

E : error

N : no more data

W : warning

? : undefined

If the "x" character is anything other than space, SQLSTATE and SQLCODE describe the error in more detail.

See the Informix manual for the description of SQLSTATE and SQLCODE

See Also

- ifx_errormsg()

ifx_errormsg

ifx_errormsg -- Returns error message of last Informix call

Description

string **ifx_errormsg** ([int `$errorcode`])

Returns the Informix error message associated with the most recent Informix error.

Parameters

`errorcode`
: If specified, the function will return the message corresponding to the specified code.

Return Values

Return the error message, as a string.

Examples

Example #5 - ifx_errormsg() **example**

```
printf("%s\n&lt;br>", ifx_errormsg(-201));
```

See Also

- ifx_error()

ifx_fetch_row

ifx_fetch_row -- Get row as an associative array

Description

array **ifx_fetch_row** (resource `$result_id` [, mixed `$position`])

Fetches one row of data from the result associated with the specified result identifier.

Subsequent calls to ifx_fetch_row() would return the next row in the result set, or **FALSE** if there are no more rows.

Parameters

`result_id`
: *`result_id`* is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

`position`
: An optional parameter for a "fetch" operation on "scroll" cursors: *NEXT*, *PREVIOUS*, *CURRENT*, *FIRST*, *LAST* or a number. If you specify a number, an "absolute" row fetch is executed. This parameter is optional, and only valid for SCROLL cursors.

Return Values

Returns an associative array that corresponds to the fetched row, or **FALSE** if there are no more rows.

Blob columns are returned as integer blob id values for use in ifx_get_blob() unless you have used ifx_textasvarchar(1) or ifx_byteasvarchar(1), in which case blobs are returned as string values.

Examples

Example #6 - Informix fetch rows

```
<?php
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
   /* ... error ... */
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
   printf ("Too many rows in result set (%d)\n<br />", $rowcount);
```

```
   die ("Please restrict your query<br />\n");
}
if (! ifx_do ($rid)) {
  /* ... error ... */
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
   for (reset($row); $fieldname=key($row); next($row)) {
       $fieldvalue = $row[$fieldname];
       printf ("%s = %s,", $fieldname, $fieldvalue);
   }
   printf("\n<br />");
   $row = ifx_fetch_row($rid, "NEXT");
}
ifx_free_result ($rid);
?>
```

ifx_fieldproperties

ifx_fieldproperties -- List of SQL fieldproperties

Description

array **ifx_fieldproperties** (resource `$result_id`)

Returns the Informix SQL fieldproperties of every field in the query as an associative array. Properties are encoded as: "SQLTYPE;length;precision;scale;ISNULLABLE" where SQLTYPE = the Informix type like "SQLVCHAR" etc. and ISNULLABLE = "Y" or "N".

Parameters

`result_id`

`result_id` is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

Return Values

Returns an associative array with fieldnames as key and the SQL fieldproperties as data for a query with *`result_id`*. Returns **FALSE** on errors.

Examples

Example #7 - Informix SQL fieldproperties

```
<?php
$properties = ifx_fieldproperties($resultid);
if (!isset($properties)) {
   /* ... error ... */
}
foreach ($properties as $fname => $val) {
   echo "$fname:\t property = $val\n";
}
?>
```

See Also

- ifx_fieldtypes()

ifx_fieldtypes

ifx_fieldtypes -- List of Informix SQL fields

Description

array **ifx_fieldtypes** (resource `$result_id`)

Returns an associative array with fieldnames as key and the SQL fieldtypes as data for the query associated with *`result_id`*.

Parameters

`result_id`
`result_id` is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

Return Values

Returns an associative array with fieldnames as key and the SQL fieldtypes as data for query with *`result_id`*. Returns **FALSE** on error.

Examples

Example #8 - Fieldnames and SQL fieldtypes

```
<?php
$types = ifx_fieldtypes($resultid);
if (is_array($types)) {
   foreach ($types as $fname => $val) {
       echo "$fname:\t type = $val\n";
   }
}
?>
```

See Also

- ifx_fieldproperties()

ifx_free_blob

ifx_free_blob -- Deletes the blob object

Description

bool **ifx_free_blob** (int `$bid`)

Deletes the blobobject for the given blob object-id.

Parameters

`bid`
The BLOB object id.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ifx_create_blob()

ifx_free_char

ifx_free_char -- Deletes the char object

Description

bool **ifx_free_char** (int `$bid`)

Deletes the charobject for the given char object-id.

Parameters

`bid`
The char object id.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ifx_create_char()

ifx_free_result

ifx_free_result -- Releases resources for the query

Description

bool **ifx_free_result** (resource `$result_id`)

Releases resources for the query associated with *`result_id`*.

Parameters

`result_id`
: *`result_id`* is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ifx_do()

ifx_get_blob

ifx_get_blob -- Return the content of a blob object

Description

string **ifx_get_blob** (int `$bid`)

Returns the content of the blob object.

Parameters

`bid`
: The BLOB object id.

Return Values

The contents of the BLOB as a string, or **FALSE** on errors.

See Also

- ifx_get_char()

ifx_get_char

ifx_get_char -- Return the content of the char object

Description

string **ifx_get_char** (int `$bid`)

Returns the content of the char object.

Parameters

`bid`
: The char object-id.

Return Values

Returns the contents as a string, or **FALSE** on errors.

See Also

- ifx_get_blob()

ifx_getsqlca

ifx_getsqlca -- Get the contents of sqlca.sqlerrd[0..5] after a query

Description

array **ifx_getsqlca** (resource `$result_id`)

Returns a pseudo-row with sqlca.sqlerrd[0] ... sqlca.sqlerrd[5] after the query associated with *`result_id`*.

For inserts, updates and deletes the values returned are those as set by the server after executing the query. This gives access to the number of affected rows and the serial insert value. For *SELECT* s the values are those saved after the *PREPARE* statement. This gives access to the *estimated* number of affected rows. The use of this function saves the overhead of executing a *SELECT dbinfo('sqlca.sqlerrdx')* query, as it retrieves the values that were saved by the ifx driver at the appropriate moment.

Parameters

`result_id`

`result_id` is a valid result id returned by ifx_query() or ifx_prepare() (select type queries only!).

Return Values

Returns an associative array with the following entries: *sqlerrd0*, *sqlerrd1*, *sqlerrd2*, *sqlerrd3*, *sqlerrd4* and *sqlerrd5*.

Examples

Example #9 - Retrieve Informix sqlca.sqlerrd[x] values

```
<?php
/* assume the first column of 'sometable' is a serial */
$qid = ifx_query("insert into sometable
                  values (0, '2nd column', 'another column') ", $connid);
if (!$qid) {
   /* ... error ... */
}
$sqlca = ifx_getsqlca($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : $serial_value<br />\n";
?>
```

ifx_htmltbl_result

ifx_htmltbl_result -- Formats all rows of a query into a HTML table

Description

int **ifx_htmltbl_result** (resource `$result_id` [, string `$html_table_options`])

Formats and prints all rows of the *`result_id`* query into a HTML table.

Parameters

`result_id`
: *`result_id`* is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

`html_table_options`
: This optional argument is a string of <table> tag options.

Return Values

Returns the number of fetched rows, or **FALSE** on errors.

Examples

Example #10 - Informix results as HTML table

```
<?php
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
   /* ... error ... */
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
   printf ("Too many rows in result set (%d)\n<br />", $rowcount);
   die ("Please restrict your query<br />\n");
}
if (! ifx_do($rid)) {
   /* ... error ... */
}

ifx_htmltbl_result ($rid, "border=\"2\"");

ifx_free_result($rid);
?>
```

ifx_nullformat

ifx_nullformat -- Sets the default return value on a fetch row

Description

bool **ifx_nullformat** (int `$mode`)

Sets the default return value of a NULL-value on a fetch row.

Parameters

`mode`
Mode "0" returns "", and mode "1" returns " **NULL** ".

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ifx_num_fields

ifx_num_fields -- Returns the number of columns in the query

Description

int **ifx_num_fields** (resource `$result_id`)

After preparing or executing a query, this call gives you the number of columns in the query.

Parameters

`result_id`
: *`result_id`* is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

Return Values

Returns the number of columns in query for *`result_id`*, or **FALSE** on errors.

Examples

Example #11 - ifx_num_fields() Example

```
<?php
$conn_id = ifx_connect("db", "user", "password");
$res_id = ifx_query("select * from systables", $conn_id);
echo ifx_num_fields($res_id);
?>
```

See Also

- ifx_num_rows()

ifx_num_rows

ifx_num_rows -- Count the rows already fetched from a query

Description

int **ifx_num_rows** (resource `$result_id`)

Gives the number of rows fetched so far for a query with *`result_id`* after a ifx_query() or ifx_do() query.

Parameters

`result_id`
: *`result_id`* is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

Return Values

Returns the number of fetched rows or **FALSE** on errors.

See Also

- ifx_num_fields()

ifx_pconnect

ifx_pconnect -- Open persistent Informix connection

Description

resource **ifx_pconnect** ([string `$database` [, string `$userid` [, string `$password`]]])

ifx_pconnect() acts very much like ifx_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (ifx_close() will not close links established by ifx_pconnect()).

This type of links is therefore called 'persistent'.

Parameters

All of the arguments are optional, and if they're missing, defaults are taken from values supplied in *php.ini* (ifx.default_host for the host (Informix libraries will use INFORMIXSERVER environment value if not defined), ifx.default_user for user, ifx.default_password for the password (none if not defined).

`database`
The database name, as a string.

`userid`
The username, as a string.

`password`
The password, as a string.

Return Values

Returns: valid Informix persistent link identifier on success, or **FALSE** on errors.

See Also

- ifx_connect()

ifx_prepare

ifx_prepare -- Prepare an SQL-statement for execution

Description

resource **ifx_prepare** (string `$query`, resource `$link_identifier` [, int `$cursor_def`], mixed `$blobidarray`)

Prepares a *`query`* for later use with ifx_do().

For "select-type" queries a cursor is declared and opened. Non-select queries are "execute immediate".

For either query type the number of (estimated or real) affected rows is saved for retrieval by ifx_affected_rows().

If the contents of the TEXT (or BYTE) column allow it, you can also use *ifx_textasvarchar(1)* and *ifx_byteasvarchar(1)*. This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With *ifx_textasvarchar(0)* or *ifx_byteasvarchar(0)* (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

Parameters

`query`
The query string.

`link_identifier`
The link identifier.

`cursor_def`
This optional parameter allows you to make this a *scroll* and/or *hold* cursor. It's a bitmask and can be either **IFX_SCROLL**, **IFX_HOLD**, or both or'ed together.

`blobidarray`
If you have BLOB (BYTE or TEXT) columns in the query, you can add a *`blobidarray`* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

Return Values

Returns a valid result identifier for use by ifx_do(), or **FALSE** on errors.

See Also

- ifx_do()

ifx_query

ifx_query -- Send Informix query

Description

resource **ifx_query** (string `$query`, resource `$link_identifier` [, int `$cursor_type` [, mixed `$blobidarray`]])

Sends a *`query`* to the currently active database on the server that's associated with the specified link identifier.

For "select-type" queries a cursor is declared and opened. Non-select queries are "execute immediate".

For either query type the number of (estimated or real) affected rows is saved for retrieval by ifx_affected_rows().

If the contents of the TEXT (or BYTE) column allow it, you can also use *ifx_textasvarchar(1)* and *ifx_byteasvarchar(1)*. This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With *ifx_textasvarchar(0)* or *ifx_byteasvarchar(0)* (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

Parameters

`query`
The query string.

`link_identifier`
The link identifier.

`cursor_def`
This optional parameter allows you to make this a *scroll* and/or *hold* cursor. It's a bitmask and can be either **IFX_SCROLL**, **IFX_HOLD**, or both or'ed together. I you omit this parameter the cursor is a normal sequential cursor.

`blobidarray`
If you have BLOB (BYTE or TEXT) columns in the query, you can add a *`blobidarray`* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

Return Values

Returns valid Informix result identifier on success, or **FALSE** on errors.

Examples

Example #12 - Show all rows of the "orders" table as a HTML table

```
<?php
ifx_textasvarchar(1);      // use "text mode" for blobs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
   printf("Can't select orders : %s\n<br />%s<br />\n", ifx_error(),
ifx_errormsg());
   die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);
?>
```

Example #13 - Insert some values into the "catalog" table

```
<?php

// create blob id's for a byte and text column
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");

// store blob id's in a blobid array
$blobidarray[] = $textid;
$blobidarray[] = $byteid;

// launch query
$query = "insert into catalog (stock_num, manu_code, " .
        "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
   /* ... error ... */
}

// free result id
ifx_free_result($res_id);
?>
```

See Also

- ifx_connect()

ifx_textasvarchar

ifx_textasvarchar -- Set the default text mode

Description

bool **ifx_textasvarchar** (int `$mode`)

Sets the default text mode for all select-queries.

Parameters

`mode`
: Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- **ifx_bytesasvarchar()**

ifx_update_blob

ifx_update_blob -- Updates the content of the blob object

Description

bool **ifx_update_blob** (int `$bid`, string `$content`)

Updates the content of the blob object for the given blob object *`bid`*.

Parameters

`bid`
: A BLOB object identifier.

`content`
: The new data, as a string.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ifx_update_char()

ifx_update_char

ifx_update_char -- Updates the content of the char object

Description

bool **ifx_update_char** (int `$bid`, string `$content`)

Updates the content of the char object for the given char object *`bid`*.

Parameters

`bid`
: A char object identifier.

`content`
: The new data, as a string.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ifx_update_blob()

ifxus_close_slob

ifxus_close_slob -- Deletes the slob object

Description

bool **ifxus_close_slob** (int `$bid`)

Deletes the slob object on the given slob object-id *`bid`*.

Parameters

`bid`
: An existing slob id.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ifxus_open_slob()

ifxus_create_slob

ifxus_create_slob -- Creates an slob object and opens it

Description

int **ifxus_create_slob** (int `$mode`)

Creates an slob object and opens it.

Parameters

`mode`
: A combination of **IFX_LO_RDONLY**, **IFX_LO_WRONLY**, **IFX_LO_APPEND** **IFX_LO_RDWR**, **IFX_LO_BUFFER**, **IFX_LO_NOBUFFER**.

Return Values

Return the new slob object-id, or **FALSE** on errors.

See Also

- ifxus_close_slob()
- ifxus_free_slob()

ifxus_free_slob

ifxus_free_slob -- Deletes the slob object

Description

bool **ifxus_free_slob** (int `$bid`)

Deletes the slob object.

Parameters

`bid`
: An existing slob id.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ifxus_close_slob()

ifxus_open_slob

ifxus_open_slob -- Opens an slob object

Description

int **ifxus_open_slob** (int `$bid`, int `$mode`)

Opens an slob object. *`bid`* should be an existing slob id.

Parameters

`bid`
: An existing slob id.

`mode`
: A combination of **IFX_LO_RDONLY**, **IFX_LO_WRONLY**, **IFX_LO_APPEND** **IFX_LO_RDWR**, **IFX_LO_BUFFER**, **IFX_LO_NOBUFFER**.

Return Values

Returns the new slob object-id, or **FALSE** on errors.

See Also

- ifxus_close_slob()
- ifxus_free_slob()

ifxus_read_slob

ifxus_read_slob -- Reads nbytes of the slob object

Description

string **ifxus_read_slob** (int `$bid`, int `$nbytes`)

Reads *`nbytes`* of the slob object.

Parameters

`bid`
: An existing slob id.

`nbytes`
: The number of bytes to read.

Return Values

Returns the slob contents as a string, or **FALSE** on errors.

See Also

- ifxus_write_slob()

ifxus_seek_slob

ifxus_seek_slob -- Sets the current file or seek position

Description

int **ifxus_seek_slob** (int `$bid`, int `$mode`, int `$offset`)

Sets the current file or seek position of an open slob object.

Parameters

`bid`
: An existing slob id.

`mode`
: 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END.

`offset`
: A byte offset.

Return Values

Returns the seek position as an integer, or **FALSE** on errors.

See Also

- ifxus_tell_slob()

ifxus_tell_slob

ifxus_tell_slob -- Returns the current file or seek position

Description

int **ifxus_tell_slob** (int `$bid`)

Returns the current file or seek position of an open slob object

Parameters

`bid`
: An existing slob id.

Return Values

Returns the seek position as an integer, or **FALSE** on errors.

See Also

- ifxus_seek_slob()

ifxus_write_slob

ifxus_write_slob -- Writes a string into the slob object

Description

int **ifxus_write_slob** (int `$bid`, string `$content`)

Writes a string into the slob object.

Parameters

`bid`
: An existing slob id.

`content`
: The content to write, as a string.

Return Values

Returns the bytes written as an integer, or **FALSE** on errors.

See Also

- ifxus_read_slob()