Firebird/InterBase

Introduction

Firebird/InterBase is a relational database offering many ANSI SQL-92 features that runs on Linux, Windows, and a variety of Unix platforms. Firebird/InterBase offers excellent concurrency, high performance, and powerful language support for stored procedures and triggers. It has been used in production systems, under a variety of names since 1981.

InterBase is the name of the closed-source variant of this RDBMS that was developed by Borland/Inprise. More information about InterBase is available at » http://www.borland.com/interbase/.

Firebird is a commercially independent project of C and C++ programmers, technical advisors and supporters developing and enhancing a multi-platform relational database management system based on the source code released by Inprise Corp (now known as Borland Software Corp) under the InterBase Public License v.1.0 on 25 July, 2000. More information about Firebird is available at » http://www.firebirdsql.org/.

Note

This extension supports InterBase versions 5 and up and all versions of Firebird. Support for InterBase version 5.x will be dropped in PHP 5.

This database uses a single quote (') character for escaping, a behavior similar to the Sybase database, add to your *php.ini* the following directive:

```
magic_quotes_sybase = On
```

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

To enable InterBase support configure PHP *--with-interbase[=DIR]*, where DIR is the InterBase base install directory, which defaults to */usr/interbase*.

Note
Note to Win32 Users In order for this extension to work, there are DLL files that must be available to the Windows system *PATH*. For information on how to do this, see the FAQ entitled " How do I add my PHP directory to the PATH on Windows ". Although copying DLL files from the PHP folder into the Windows system directory also works (because the system directory is by default in the system's *PATH*), this is not recommended. *This extension requires the following files to be in the PATH: gds32.dll* In case you installed the InterBase database server on the same machine PHP is running on, you will have this DLL already. Therefore you don't need to worry because *gds32.dll* will already be in the *PATH*.

Runtime Configuration

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

InterBase configuration options

Name	Default	Changeable	Changelog
ibase.allow_persistent	"1"	PHP_INI_SYSTEM	
ibase.max_persistent	"-1"	PHP_INI_SYSTEM	
ibase.max_links	"-1"	PHP_INI_SYSTEM	
ibase.default_db	NULL	PHP_INI_SYSTEM	Available since PHP

			5.0.0.
ibase.default_user	NULL	PHP_INI_ALL	
ibase.default_password	NULL	PHP_INI_ALL	
ibase.default_charset	NULL	PHP_INI_ALL	Available since PHP 5.0.0.
ibase.timestampformat	"%Y-%m-%d %H:%M:%S"	PHP_INI_ALL	
ibase.dateformat	"%Y-%m-%d"	PHP_INI_ALL	
ibase.timeformat	"%H:%M:%S"	PHP_INI_ALL	

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

Here's a short explanation of the configuration directives.

ibase.allow_persistent boolean
: Whether to allow persistent connections to Firebird/InterBase.

ibase.max_persistent integer
: The maximum number of persistent Firebird/InterBase connections per process. New connections created with ibase_pconnect() will be non-persistent if this number would be exceeded.

ibase.max_links integer
: The maximum number of Firebird/InterBase connections per process, including persistent connections.

ibase.default_db string
: The default database to connect to when ibase_[p]connect() is called without specifying a database name. If this value is set and SQL safe mode is enabled, no other connections than to this database will be allowed.

ibase.default_user string
: The user name to use when connecting to a database if no user name is specified.

ibase.default_password string
: The password to use when connecting to a database if no password is specified.

ibase.default_charset string
: The character set to use when connecting to a database if no character set is specified.

ibase.timestampformat string

ibase.dateformat string

ibase.timeformat string

These directives are used to set the date and time formats that are used when returning dates and times from a result set, or when binding arguments to date and time parameters.

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.

The following constants can be passed to ibase_trans() to specify transaction behaviour.

Firebird/InterBase transaction flags

Constant	Description
IBASE_DEFAULT	The default transaction settings are to be used. This default is determined by the client library, which defines it as IBASE_WRITE|IBASE_CONCURRENCY|IBASE_WAIT in most cases.
IBASE_READ	Starts a read-only transaction.
IBASE_WRITE	Starts a read-write transaction.
IBASE_CONSISTENCY	Starts a transaction with the isolation level set to 'consistency', which means the transaction cannot read from tables that are being modified by other concurrent transactions.
IBASE_CONCURRENCY	Starts a transaction with the isolation level set to 'concurrency' (or 'snapshot'), which means the transaction has access to all tables, but cannot see changes that were committed by other transactions after the transaction was started.
IBASE_COMMITTED	Starts a transaction with the isolation level set to 'read committed'. This flag should be combined with either **IBASE_REC_VERSION** or **IBASE_REC_NO_VERSION**. This isolation level allows access to changes that were committed after the transaction was started. If **IBASE_REC_NO_VERSION** was specified, only the latest version of a row can be read. If **IBASE_REC_VERSION** was specified, a row can even be read when a modification to it is pending in a concurrent transaction.
IBASE_WAIT	Indicated that a transaction should wait and retry when a conflict occurs.

IBASE_NOWAIT	Indicated that a transaction should fail immediately when a conflict occurs.

The following constants can be passed to ibase_fetch_row(), ibase_fetch_assoc() or ibase_fetch_object() to specify fetch behaviour.

Firebird/InterBase fetch flags

Constant	Description
IBASE_FETCH_BLOBS	Also available as **IBASE_TEXT** for backward compatibility. Causes BLOB contents to be fetched inline, instead of being fetched as BLOB identifiers.
IBASE_FETCH_ARRAYS	Causes arrays to be fetched inline. Otherwise, array identifiers are returned. Array identifiers can only be used as arguments to INSERT operations, as no functions to handle array identifiers are currently available.
IBASE_UNIXTIME	Causes date and time fields not to be returned as strings, but as UNIX timestamps (the number of seconds since the epoch, which is 1-Jan-1970 0:00 UTC). Might be problematic if used with dates before 1970 on some systems.

The following constants are used to pass requests and options to the service API functions (ibase_server_info(), ibase_db_info(), ibase_backup(), ibase_restore() and ibase_maintain_db()). Please refer to the Firebird/InterBase manuals for the meaning of these options.

IBASE_BKP_IGNORE_CHECKSUMS

IBASE_BKP_IGNORE_LIMBO

IBASE_BKP_METADATA_ONLY

IBASE_BKP_NO_GARBAGE_COLLECT

IBASE_BKP_OLD_DESCRIPTIONS

IBASE_BKP_NON_TRANSPORTABLE

IBASE_BKP_CONVERT
Options to ibase_backup()

IBASE_RES_DEACTIVATE_IDX

IBASE_RES_NO_SHADOW

IBASE_RES_NO_VALIDITY

IBASE_RES_ONE_AT_A_TIME

IBASE_RES_REPLACE

IBASE_RES_CREATE

IBASE_RES_USE_ALL_SPACE
Options to ibase_restore()

IBASE_PRP_PAGE_BUFFERS

IBASE_PRP_SWEEP_INTERVAL

IBASE_PRP_SHUTDOWN_DB

IBASE_PRP_DENY_NEW_TRANSACTIONS

IBASE_PRP_DENY_NEW_ATTACHMENTS

IBASE_PRP_RESERVE_SPACE

IBASE_PRP_RES_USE_FULL

IBASE_PRP_RES

IBASE_PRP_WRITE_MODE

IBASE_PRP_WM_ASYNC

IBASE_PRP_WM_SYNC

IBASE_PRP_ACCESS_MODE

IBASE_PRP_AM_READONLY

IBASE_PRP_AM_READWRITE

IBASE_PRP_SET_SQL_DIALECT

IBASE_PRP_ACTIVATE

IBASE_PRP_DB_ONLINE

IBASE_RPR_CHECK_DB

IBASE_RPR_IGNORE_CHECKSUM

IBASE_RPR_KILL_SHADOWS

IBASE_RPR_MEND_DB

IBASE_RPR_VALIDATE_DB

IBASE_RPR_FULL

IBASE_RPR_SWEEP_DB
Options to ibase_maintain_db()

IBASE_STS_DATA_PAGES

IBASE_STS_DB_LOG

IBASE_STS_HDR_PAGES

IBASE_STS_IDX_PAGES

IBASE_STS_SYS_RELATIONS
Options to ibase_db_info()

IBASE_SVC_SERVER_VERSION

IBASE_SVC_IMPLEMENTATION

IBASE_SVC_GET_ENV

IBASE_SVC_GET_ENV_LOCK

IBASE_SVC_GET_ENV_MSG

IBASE_SVC_USER_DBPATH

IBASE_SVC_SVR_DB_INFO

IBASE_SVC_GET_USERS
Options to ibase_server_info()

Firebird/InterBase Functions

ibase_add_user

ibase_add_user -- Add a user to a security database (only for IB6 or later)

Description

bool **ibase_add_user** (resource `$service_handle`, string `$user_name`, string `$password` [, string `$first_name` [, string `$middle_name` [, string `$last_name`]]])

PHP 4 uses *`server`*, *`dba_user_name`* and *`dba_user_password`* instead of *`service_handle`* parameter.

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

See Also

- ibase_modify_user()
- ibase_delete_user()

ibase_affected_rows

ibase_affected_rows -- Return the number of rows that were affected by the previous query

Description

int **ibase_affected_rows** ([resource `$link_identifier`])

This function returns the number of rows that were affected by the previous query (INSERT, UPDATE or DELETE) that was executed from within the specified transaction context.

Parameters

`link_identifier`
: A transaction context. If *`link_identifier`* is a connection resource, its default transaction is used.

Return Values

Returns the number of rows as an integer.

See Also

- ibase_query()
- ibase_execute()

ibase_backup

ibase_backup -- Initiates a backup task in the service manager and returns immediately

Description

mixed **ibase_backup** (resource `$service_handle`, string `$source_db`, string `$dest_file` [, int `$options` [, bool `$verbose`]])

Warning
This function is currently not documented; only its argument list is available.

ibase_blob_add

ibase_blob_add -- Add data into a newly created blob

Description

void **ibase_blob_add** (resource `$blob_handle`, string `$data`)

ibase_blob_add() adds data into a blob created with ibase_blob_create().

Parameters

`blob_handle`
A blob handle opened with ibase_blob_create().

`data`
The data to be added.

Return Values

No value is returned.

See Also

- ibase_blob_cancel()
- ibase_blob_close()
- ibase_blob_create()
- ibase_blob_import()

ibase_blob_cancel

ibase_blob_cancel -- Cancel creating blob

Description

bool **ibase_blob_cancel** (resource `$blob_handle`)

This function will discard a BLOB if it has not yet been closed by ibase_blob_close().

Parameters

`blob_handle`
A BLOB handle opened with **ibase_create_blob()**.

Return Values

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

See Also

- ibase_blob_close()
- ibase_blob_create()
- ibase_blob_import()

ibase_blob_close

ibase_blob_close -- Close blob

Description

mixed **ibase_blob_close** (resource `$blob_handle`)

This function closes a BLOB that has either been opened for reading by **ibase_open_blob()** or has been opened for writing by **ibase_create_blob()**.

Parameters

`blob_handle`
A BLOB handle opened with **ibase_create_blob()** or **ibase_open_blob()**.

Return Values

If the BLOB was being read, this function returns **TRUE** on success, if the BLOB was being written to, this function returns a string containing the BLOB id that has been assigned to it by the database. On failure, this function returns **FALSE**.

See Also

- ibase_blob_cancel()
- ibase_blob_open()

ibase_blob_create

ibase_blob_create -- Create a new blob for adding data

Description

resource **ibase_blob_create** ([resource `$link_identifier`])

ibase_blob_create() creates a new BLOB for filling with data.

Parameters

`link_identifier`
An InterBase link identifier. If omitted, the last opened link is assumed.

Return Values

Returns a BLOB handle for later use with ibase_blob_add() or **FALSE** on failure.

See Also

- ibase_blob_add()
- ibase_blob_cancel()
- ibase_blob_close()
- ibase_blob_import()

ibase_blob_echo

ibase_blob_echo -- Output blob contents to browser

Description

bool **ibase_blob_echo** ([resource `$link_identifier`], string `$blob_id`)

This function opens a BLOB for reading and sends its contents directly to standard output (the browser, in most cases).

Parameters

`link_identifier`
An InterBase link identifier. If omitted, the last opened link is assumed.

`blob_id`

Return Values

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

See Also

- ibase_blob_open()
- ibase_blob_close()
- ibase_blob_get()

ibase_blob_get

ibase_blob_get -- Get len bytes data from open blob

Description

string **ibase_blob_get** (resource `$blob_handle`, int `$len`)

This function returns at most *`len`* bytes from a BLOB that has been opened for reading by ibase_blob_open().

Note
It is not possible to read from a BLOB that has been opened for writing by ibase_blob_create().

Parameters

`blob_handle`
A BLOB handle opened with ibase_blob_open().

`len`
Size of returned data.

Return Values

Returns at most *`len`* bytes from the BLOB, or **FALSE** on failure.

Examples

Example #1 - ibase_blob_get() example

```
<?php
$result    = ibase_query("SELECT blob_value FROM table");
$data      = ibase_fetch_object($result);
$blob_data = ibase_blob_info($data->BLOB_VALUE);
$blob_hndl = ibase_blob_open($data->BLOB_VALUE);
echo         ibase_blob_get($blob_hndl, $blob_data[0]);
?>
```

Whilst this example doesn't do much more than a 'ibase_blob_echo($data->BLOB_VALUE)' would do, it does show you how to get information into a $variable to manipulate as you please.

See Also

- ibase_blob_open()
- ibase_blob_close()
- ibase_blob_echo()

ibase_blob_import

ibase_blob_import -- Create blob, copy file in it, and close it

Description

string **ibase_blob_import** (resource `$link_identifier`, resource `$file_handle`)

string **ibase_blob_import** (resource `$file_handle`)

This function creates a BLOB, reads an entire file into it, closes it and returns the assigned BLOB id.

Parameters

`link_identifier`
An InterBase link identifier. If omitted, the last opened link is assumed.

`file_handle`
The file handle is a handle returned by fopen().

Return Values

Returns the BLOB id on success, or **FALSE** on error.

Examples

Example #2 - ibase_blob_import() example

```
<?php
$dbh = ibase_connect($host, $username, $password);
$filename = '/tmp/bar';

$fd = fopen($filename, 'r');
if ($fd) {

   $blob = ibase_blob_import($dbh, $fd);
   fclose($fd);

   if (!is_string($blob)) {
       // import failed
   } else {
       $query = "INSERT INTO foo (name, data) VALUES ('$filename', ?)";
       $prepared = ibase_prepare($dbh, $query);
       if (!ibase_execute($prepared, $blob)) {
           // record insertion failed
       }
```

```
    }
} else {
    // unable to open the data file
}
?>
```

See Also

- ibase_blob_add()
- ibase_blob_cancel()
- ibase_blob_close()
- ibase_blob_create()

ibase_blob_info

ibase_blob_info -- Return blob length and other useful info

Description

array **ibase_blob_info** (resource `$link_identifier`, string `$blob_id`)

array **ibase_blob_info** (string `$blob_id`)

Returns the BLOB length and other useful information.

Parameters

`link_identifier`
An InterBase link identifier. If omitted, the last opened link is assumed.

`blob_id`
A BLOB id.

Return Values

Returns an array containing information about a BLOB. The information returned consists of the length of the BLOB, the number of segments it contains, the size of the largest segment, and whether it is a stream BLOB or a segmented BLOB.

ibase_blob_open

ibase_blob_open -- Open blob for retrieving data parts

Description

resource **ibase_blob_open** (resource `$link_identifier`, string `$blob_id`)

resource **ibase_blob_open** (string `$blob_id`)

Opens an existing BLOB for reading.

Parameters

`link_identifier`
An InterBase link identifier. If omitted, the last opened link is assumed.

`blob_id`
A BLOB id.

Return Values

Returns a BLOB handle for later use with ibase_blob_get() or **FALSE** on failure.

See Also

- ibase_blob_close()
- ibase_blob_echo()
- ibase_blob_get()

ibase_close

ibase_close -- Close a connection to an InterBase database

Description

bool **ibase_close** ([resource `$connection_id`])

Closes the link to an InterBase database that's associated with a connection id returned from ibase_connect(). Default transaction on link is committed, other transactions are rolled back.

Parameters

`connection_id`
: An InterBase link identifier returned from ibase_connect(). If omitted, the last opened link is assumed.

Return Values

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

See Also

- ibase_connect()
- ibase_pconnect()

ibase_commit_ret

ibase_commit_ret -- Commit a transaction without closing it

Description

bool **ibase_commit_ret** ([resource `$link_or_trans_identifier`])

Commits a transaction without closing it.

Parameters

`link_or_trans_identifier`
: If called without an argument, this function commits the default transaction of the default link. If the argument is a connection identifier, the default transaction of the corresponding connection will be committed. If the argument is a transaction identifier, the corresponding transaction will be committed. The transaction context will be retained, so statements executed from within this transaction will not be invalidated.

Return Values

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

ibase_commit

ibase_commit -- Commit a transaction

Description

bool **ibase_commit** ([resource `$link_or_trans_identifier`])

Commits a transaction.

Parameters

`link_or_trans_identifier`
: If called without an argument, this function commits the default transaction of the default link. If the argument is a connection identifier, the default transaction of the corresponding connection will be committed. If the argument is a transaction identifier, the corresponding transaction will be committed.

Return Values

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

ibase_connect

ibase_connect -- Open a connection to an InterBase database

Description

resource **ibase_connect** ([string `$database` [, string `$username` [, string `$password` [, string `$charset` [, int `$buffers` [, int `$dialect` [, string `$role` [, int `$sync`]]]]]]])

Establishes a connection to an InterBase server.

In case a second call is made to ibase_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 ibase_close().

Parameters

`database`
The *`database`* argument has to be a valid path to database file on the server it resides on. If the server is not local, it must be prefixed with either 'hostname:' (TCP/IP), '//hostname/' (NetBEUI) or 'hostname@' (IPX/SPX), depending on the connection protocol used.

`username`
The user name. Can be set with the *ibase.default_user php.ini* directive.

`password`
The password for *`username`*. Can be set with the *ibase.default_password php.ini* directive.

`charset`
`charset` is the default character set for a database.

`buffers`
`buffers` is the number of database buffers to allocate for the server-side cache. If 0 or omitted, server chooses its own default.

`dialect`
`dialect` selects the default SQL dialect for any statement executed within a connection, and it defaults to the highest one supported by client libraries. Functional only with InterBase 6 and up.

`role`
Functional only with InterBase 5 and up.

`sync`

Return Values

Returns an InterBase link identifier on success, or **FALSE** on error.

Errors/Exceptions

If you get some error like "arithmetic exception, numeric overflow, or string truncation. Cannot transliterate character between character sets" (this occurs when you try use some character with accents) when using this and after ibase_query() you must set the character set (i.e. ISO8859_1 or your current character set).

ChangeLog

Version	Description
4.0.0	The *buffers*, *dialect* and *role* parameters were added

Examples

Example #3 - ibase_connect() example

```
<?php
$host = 'localhost:/path/to/your.gdb';

$dbh = ibase_connect($host, $username, $password);
$stmt = 'SELECT * FROM tblname';
$sth = ibase_query($dbh, $stmt);
while ($row = ibase_fetch_object($sth)) {
   echo $row->email, "\n";
}
ibase_free_result($sth);
ibase_close($dbh);
?>
```

See Also

- ibase_pconnect()
- ibase_close()

ibase_db_info

ibase_db_info -- Request statistics about a database

Description

string **ibase_db_info** (resource `$service_handle`, string `$db`, int `$action` [, int `$argument`])

Warning
This function is currently not documented; only its argument list is available.

ibase_delete_user

ibase_delete_user -- Delete a user from a security database (only for IB6 or later)

Description

bool **ibase_delete_user** (resource `$service_handle`, string `$user_name`)

PHP 4 uses *`server`*, *`dba_user_name`* and *`dba_user_password`* instead of *`service_handle`* parameter.

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

See Also

- ibase_add_user()
- ibase_modify_user()

ibase_drop_db

ibase_drop_db -- Drops a database

Description

bool **ibase_drop_db** ([resource `$connection`])

This functions drops a database that was opened by either ibase_connect() or ibase_pconnect(). The database is closed and deleted from the server.

Parameters

`connection`
An InterBase link identifier. If omitted, the last opened link is assumed.

Return Values

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

See Also

- ibase_connect()
- ibase_pconnect()

ibase_errcode

ibase_errcode -- Return an error code

Description

int **ibase_errcode** (void)

Returns the error code that resulted from the most recent InterBase function call.

Return Values

Returns the error code as an integer, or **FALSE** if no error occured.

See Also

- ibase_errmsg()

ibase_errmsg

ibase_errmsg -- Return error messages

Description

string **ibase_errmsg** (void)

Returns the error message that resulted from the most recent InterBase function call.

Return Values

Returns the error message as a string, or **FALSE** if no error occured.

See Also

- ibase_errcode()

ibase_execute

ibase_execute -- Execute a previously prepared query

Description

resource **ibase_execute** (resource `$query` [, mixed `$bind_arg` [, mixed `$...`]])

Execute a query prepared by ibase_prepare().

This is a lot more effective than using ibase_query() if you are repeating a same kind of query several times with only some parameters changing.

Parameters

`query`
An InterBase query prepared by ibase_prepare().

`bind_arg`

`...`

Return Values

If the query raises an error, returns **FALSE**. If it is successful and there is a (possibly empty) result set (such as with a SELECT query), returns a result identifier. If the query was successful and there were no results, returns **TRUE**.

Note
In PHP 5.0.0 and up, this function returns the number of rows affected by the query (if > 0 and applicable to the statement type). A query that succeeded, but did not affect any rows (e.g. an UPDATE of a non-existent record) will return **TRUE**.

Examples

Example #4 - ibase_execute() example

```
<?php

$dbh = ibase_connect($host, $username, $password);
```

```
$updates = array(
   1 => 'Eric',
   5 => 'Filip',
   7 => 'Larry'
);

$query = ibase_prepare($dbh, "UPDATE FOO SET BAR = ? WHERE BAZ = ?");

foreach ($updates as $baz => $bar) {
   ibase_execute($query, $bar, $baz);
}

?>
```

See Also

- ibase_query()

ibase_fetch_assoc

ibase_fetch_assoc -- Fetch a result row from a query as an associative array

Description

array **ibase_fetch_assoc** (resource `$result` [, int `$fetch_flag`])

Fetch a result row from a query as an associative array.

ibase_fetch_assoc() fetches one row of data from the *`result`*. If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you either need to access the result with numeric indices by using ibase_fetch_row() or use alias names in your query.

Parameters

`result`
: The result handle.

`fetch_flag`
: *`fetch_flag`* is a combination of the constants **IBASE_TEXT** and **IBASE_UNIXTIME** ORed together. Passing **IBASE_TEXT** will cause this function to return BLOB contents instead of BLOB ids. Passing **IBASE_UNIXTIME** will cause this function to return date/time values as Unix timestamps instead of as formatted strings.

Return Values

Returns an associative array that corresponds to the fetched row. Subsequent calls will return the next row in the result set, or **FALSE** if there are no more rows.

See Also

- ibase_fetch_row()
- ibase_fetch_object()

ibase_fetch_object

ibase_fetch_object -- Get an object from a InterBase database

Description

object **ibase_fetch_object** (resource `$result_id` [, int `$fetch_flag`])

Fetches a row as a pseudo-object from a given result identifier.

Subsequent calls to ibase_fetch_object() return the next row in the result set.

Parameters

`result_id`
An InterBase result identifier obtained either by ibase_query() or ibase_execute().

`fetch_flag`
`fetch_flag` is a combination of the constants **IBASE_TEXT** and **IBASE_UNIXTIME** ORed together. Passing **IBASE_TEXT** will cause this function to return BLOB contents instead of BLOB ids. Passing **IBASE_UNIXTIME** will cause this function to return date/time values as Unix timestamps instead of as formatted strings.

Return Values

Returns an object with the next row information, or **FALSE** if there are no more rows.

Examples

Example #5 - ibase_fetch_object() example

```
<?php
$dbh = ibase_connect($host, $username, $password);
$stmt = 'SELECT * FROM tblname';
$sth = ibase_query($dbh, $stmt);
while ($row = ibase_fetch_object($sth)) {
   echo $row->email . "\n";
}
ibase_close($dbh);
?>
```

See Also

- ibase_fetch_row()
- ibase_fetch_assoc()

ibase_fetch_row

ibase_fetch_row -- Fetch a row from an InterBase database

Description

array **ibase_fetch_row** (resource `$result_identifier` [, int `$fetch_flag`])

ibase_fetch_row() fetches one row of data from the given result set.

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

Parameters

`result_identifier`
An InterBase result identifier.

`fetch_flag`
`fetch_flag` is a combination of the constants **IBASE_TEXT** and **IBASE_UNIXTIME** ORed together. Passing **IBASE_TEXT** will cause this function to return BLOB contents instead of BLOB ids. Passing **IBASE_UNIXTIME** will cause this function to return date/time values as Unix timestamps instead of as formatted strings.

Return Values

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows. Each result column is stored in an array offset, starting at offset 0.

See Also

- ibase_fetch_assoc()
- ibase_fetch_object()

ibase_field_info

ibase_field_info -- Get information about a field

Description

array **ibase_field_info** (resource `$result`, int `$field_number`)

Returns an array with information about a field after a select query has been run.

Parameters

`result`
An InterBase result identifier.

`field_number`
Field offset.

Return Values

Returns an array with the following keys: *name*, *alias*, *relation*, *length* and *type*.

Examples

Example #6 - ibase_field_info() example

```
<?php
$rs = ibase_query("SELECT * FROM tablename");
$coln = ibase_num_fields($rs);
for ($i = 0; $i < $coln; $i++) {
   $col_info = ibase_field_info($rs, $i);
   echo "name: ". $col_info['name']. "\n";
   echo "alias: ". $col_info['alias']. "\n";
   echo "relation: ". $col_info['relation']. "\n";
   echo "length: ". $col_info['length']. "\n";
   echo "type: ". $col_info['type']. "\n";
}
?>
```

See Also

- ibase_num_fields()

ibase_free_event_handler

ibase_free_event_handler -- Cancels a registered event handler

Description

bool **ibase_free_event_handler** (resource `$event`)

This function causes the registered event handler specified by *`event`* to be cancelled. The callback function will no longer be called for the events it was registered to handle.

Parameters

`event`
An event resource, created by ibase_set_event_handler().

Return Values

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

See Also

- ibase_set_event_handler()

ibase_free_query

ibase_free_query -- Free memory allocated by a prepared query

Description

bool **ibase_free_query** (resource `$query`)

Frees a prepared query.

Parameters

`query`
A query prepared with ibase_prepare().

Return Values

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

ibase_free_result

ibase_free_result -- Free a result set

Description

bool **ibase_free_result** (resource `$result_identifier`)

Frees a result set.

Parameters

`result_identifier`
A result set created by ibase_query() or ibase_execute().

Return Values

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

ibase_gen_id

ibase_gen_id -- Increments the named generator and returns its new value

Description

mixed **ibase_gen_id** (string `$generator` [, int `$increment` [, resource `$link_identifier`]])

Warning
This function is currently not documented; only its argument list is available.

Return Values

Returns new generator value as integer, or as string if the value is too big.

ibase_maintain_db

ibase_maintain_db -- Execute a maintenance command on the database server

Description

bool **ibase_maintain_db** (resource `$service_handle`, string `$db`, int `$action` [, int `$argument`])

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

ibase_modify_user

ibase_modify_user -- Modify a user to a security database (only for IB6 or later)

Description

bool **ibase_modify_user** (resource `$service_handle`, string `$user_name`, string `$password` [, string `$first_name` [, string `$middle_name` [, string `$last_name`]]])

Warning
This function is currently not documented; only its argument list is available.

PHP 4 uses *`server`*, *`dba_user_name`* and *`dba_user_password`* instead of *`service_handle`* parameter.

Return Values

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

See Also

- ibase_add_user()
- ibase_delete_user()

ibase_name_result

ibase_name_result -- Assigns a name to a result set

Description

bool **ibase_name_result** (resource `$result`, string `$name`)

This function assigns a name to a result set. This name can be used later in UPDATE|DELETE ... WHERE CURRENT OF *`name`* statements.

Parameters

`result`
An InterBase result set.

`name`
The name to be assigned.

Return Values

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

Examples

Example #7 - ibase_name_result() **example**

```
<?php
$result = ibase_query("SELECT field1,field2 FROM table FOR UPDATE");
ibase_name_result($result, "my_cursor");

$updateqry = ibase_prepare("UPDATE table SET field2 = ? WHERE CURRENT OF
my_cursor");

for ($i = 0; ibase_fetch_row($result); ++$i) {
   ibase_execute($updateqry, $i);
}
?>
```

See Also

- ibase_prepare()

- ibase_execute()

ibase_num_fields

ibase_num_fields -- Get the number of fields in a result set

Description

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

Get the number of fields in a result set.

Parameters

`result_id`
An InterBase result identifier.

Return Values

Returns the number of fields as an integer.

Examples

Example #8 - ibase_num_fields() **example**

```
<?php
$rs = ibase_query("SELECT * FROM tablename");
$coln = ibase_num_fields($rs);
for ($i = 0; $i < $coln; $i++) {
   $col_info = ibase_field_info($rs, $i);
   echo "name: " . $col_info['name'] . "\n";
   echo "alias: " . $col_info['alias'] . "\n";
   echo "relation: " . $col_info['relation'] . "\n";
   echo "length: " . $col_info['length'] . "\n";
   echo "type: " . $col_info['type'] . "\n";
}
?>
```

See Also

- ibase_field_info()

ibase_num_params

ibase_num_params -- Return the number of parameters in a prepared query

Description

int **ibase_num_params** (resource `$query`)

This function returns the number of parameters in the prepared query specified by *`query`*. This is the number of binding arguments that must be present when calling ibase_execute().

Parameters

`query`
The prepared query handle.

Return Values

Returns the number of parameters as an integer.

See Also

- ibase_prepare()
- ibase_param_info()

ibase_param_info

ibase_param_info -- Return information about a parameter in a prepared query

Description

array **ibase_param_info** (resource `$query`, int `$param_number`)

Returns an array with information about a parameter after a query has been prepared.

Parameters

`query`
An InterBase prepared query handle.

`param_number`
Parameter offset.

Return Values

Returns an array with the following keys: *name*, *alias*, *relation*, *length* and *type*.

See Also

- ibase_field_info()
- ibase_num_params()

ibase_pconnect

ibase_pconnect -- Open a persistent connection to an InterBase database

Description

resource **ibase_pconnect** ([string `$database` [, string `$username` [, string `$password` [, string `$charset` [, int `$buffers` [, int `$dialect` [, string `$role` [, int `$sync`]]]]]]])

Opens a persistent connection to an InterBase database.

ibase_pconnect() acts very much like ibase_connect() with two major differences.

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

Second, the connection to the InterBase server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (ibase_close() will not close links established by ibase_pconnect()). This type of link is therefore called 'persistent'.

Parameters

`database`
: The *`database`* argument has to be a valid path to database file on the server it resides on. If the server is not local, it must be prefixed with either 'hostname:' (TCP/IP), '//hostname/' (NetBEUI) or 'hostname@' (IPX/SPX), depending on the connection protocol used.

`username`
: The user name. Can be set with the *ibase.default_user php.ini* directive.

`password`
: The password for *`username`*. Can be set with the *ibase.default_password php.ini* directive.

`charset`
: *`charset`* is the default character set for a database.

`buffers`
: *`buffers`* is the number of database buffers to allocate for the server-side cache. If 0 or omitted, server chooses its own default.

`dialect`
: *`dialect`* selects the default SQL dialect for any statement executed within a connection, and it defaults to the highest one supported by client libraries. Functional only with InterBase 6 and up.

role
: Functional only with InterBase 5 and up.

sync

Return Values

Returns an InterBase link identifier on success, or **FALSE** on error.

ChangeLog

Version	Description
4.0.0	The *buffers*, *dialect* and *role* parameters were added

See Also

- ibase_close()
- ibase_connect()

ibase_prepare

ibase_prepare -- Prepare a query for later binding of parameter placeholders and execution

Description

resource **ibase_prepare** (string `$query`)

resource **ibase_prepare** (resource `$link_identifier`, string `$query`)

resource **ibase_prepare** (resource `$link_identifier`, string `$trans`, string `$query`)

Prepare a query for later binding of parameter placeholders and execution (via ibase_execute()).

Parameters

`query`
An InterBase query.

Return Values

Returns a prepared query handle, or **FALSE** on error.

ibase_query

ibase_query -- Execute a query on an InterBase database

Description

resource **ibase_query** ([resource `$link_identifier`], string `$query` [, int `$bind_args`])

Performs a query on an InterBase database.

Parameters

`link_identifier`
An InterBase link identifier. If omitted, the last opened link is assumed.

`query`
An InterBase query.

`bind_args`

Return Values

If the query raises an error, returns **FALSE**. If it is successful and there is a (possibly empty) result set (such as with a SELECT query), returns a result identifier. If the query was successful and there were no results, returns **TRUE**.

Note
In PHP 5.0.0 and up, this function will return the number of rows affected by the query for INSERT, UPDATE and DELETE statements. In order to retain backward compatibility, it will return **TRUE** for these statements if the query succeeded without affecting any rows.

Errors/Exceptions

If you get some error like "arithmetic exception, numeric overflow, or string truncation. Cannot transliterate character between character sets" (this occurs when you try use some character with accents) when using this and after ibase_query() you must set the character set (i.e. ISO8859_1 or your current character set).

Examples

Example #9 - ibase_query() **example**

```
<?php

$host = 'localhost:/path/to/your.gdb';

$dbh = ibase_connect($host, $username, $password);
$stmt = 'SELECT * FROM tblname';

$sth = ibase_query($dbh, $stmt) or die(ibase_errmsg());

?>
```

See Also

- ibase_errmsg()
- ibase_fetch_row()
- ibase_fetch_object()
- ibase_free_result()

ibase_restore

ibase_restore -- Initiates a restore task in the service manager and returns immediately

Description

mixed **ibase_restore** (resource `$service_handle`, string `$source_file`, string `$dest_db` [, int `$options` [, bool `$verbose`]])

Warning
This function is currently not documented; only its argument list is available.

ibase_rollback_ret

ibase_rollback_ret -- Roll back a transaction without closing it

Description

bool **ibase_rollback_ret** ([resource `$link_or_trans_identifier`])

Rolls back a transaction without closing it.

Parameters

`link_or_trans_identifier`
: If called without an argument, this function rolls back the default transaction of the default link. If the argument is a connection identifier, the default transaction of the corresponding connection will be rolled back. If the argument is a transaction identifier, the corresponding transaction will be rolled back. The transaction context will be retained, so statements executed from within this transaction will not be invalidated.

Return Values

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

ibase_rollback

ibase_rollback -- Roll back a transaction

Description

bool **ibase_rollback** ([resource `$link_or_trans_identifier`])

Rolls back a transaction.

Parameters

`link_or_trans_identifier`
: If called without an argument, this function rolls back the default transaction of the default link. If the argument is a connection identifier, the default transaction of the corresponding connection will be rolled back. If the argument is a transaction identifier, the corresponding transaction will be rolled back.

Return Values

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

ibase_server_info

ibase_server_info -- Request information about a database server

Description

string **ibase_server_info** (resource `$service_handle`, int `$action`)

Warning
This function is currently not documented; only its argument list is available.

ibase_service_attach

ibase_service_attach -- Connect to the service manager

Description

resource **ibase_service_attach** (string `$host`, string `$dba_username`, string `$dba_password`)

Warning
This function is currently not documented; only its argument list is available.

ibase_service_detach

ibase_service_detach -- Disconnect from the service manager

Description

bool **ibase_service_detach** (resource `$service_handle`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

ibase_set_event_handler

ibase_set_event_handler -- Register a callback function to be called when events are posted

Description

resource **ibase_set_event_handler** (callback `$event_handler`, string `$event_name1` [, string `$event_name2` [, string `$...`]])

resource **ibase_set_event_handler** (resource `$connection`, callback `$event_handler`, string `$event_name1` [, string `$event_name2` [, string `$...`]])

This function registers a PHP user function as event handler for the specified events.

Parameters

`event_handler`
: The callback is called with the event name and the link resource as arguments whenever one of the specified events is posted by the database. The callback must return **FALSE** if the event handler should be canceled. Any other return value is ignored. This function accepts up to 15 event arguments.

`event_name1`
: An event name.

`event_name2`

`...`

Return Values

The return value is an event resource. This resource can be used to free the event handler using ibase_free_event_handler().

Examples

Example #10 - ibase_set_event_handler() example

```
<?php

function event_handler($event_name, $link)
{
```

```
    if ($event_name == "NEW ORDER") {
        // process new order
        ibase_query($link, "UPDATE orders SET status='handled'");
    } else if ($event_name == "DB_SHUTDOWN") {
        // free event handler
        return false;
    }
}

ibase_set_event_handler($link, "event_handler", "NEW_ORDER", "DB_SHUTDOWN");
?>
```

See Also

- ibase_free_event_handler()
- ibase_wait_event()

ibase_timefmt

ibase_timefmt -- Sets the format of timestamp, date and time type columns returned from queries

Description

bool **ibase_timefmt** (string `$format` [, int `$columntype`])

Sets the format of timestamp, date or time type columns returned from queries.

You can set defaults for these formats with the PHP configuration directives *ibase.timestampformat*, *ibase.dateformat* and *ibase.timeformat*.

Note
This function has been removed from PHP 5, use ini_set() instead.

Parameters

`format`
: Internally, the columns are formatted by c-function strftime(), so refer to its documentation regarding to the format of the string.

`columntype`
: *`columntype`* is one of the constants **IBASE_TIMESTAMP**, **IBASE_DATE** and **IBASE_TIME**. If omitted, defaults to **IBASE_TIMESTAMP** for backwards compatibility.

Return Values

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

Examples

Example #11 - ibase_timefmt() example

```
<?php
/* InterBase 6 TIME-type columns will be returned in
* the form '05 hours 37 minutes'. */
ibase_timefmt("%H hours %M minutes", IBASE_TIME);
?>
```

ibase_trans

ibase_trans -- Begin a transaction

Description

resource **ibase_trans** ([int `$trans_args` [, resource `$link_identifier`]])

Begins a transaction.

Note
The behaviour of this function has been changed in PHP 5.0.0. The first call to ibase_trans() will not return the default transaction of a connection. All transactions started by ibase_trans() will be rolled back at the end of the script if they were not committed or rolled back by either ibase_commit() or ibase_rollback().

Note
In PHP 5.0.0. and up, this function will accept multiple *`trans_args`* and *`link_identifier`* arguments. This allows transactions over multiple database connections, which are committed using a 2-phase commit algorithm. This means you can rely on the updates to either succeed in every database, or fail in every database. It does NOT mean you can use tables from different databases in the same query! If you use transactions over multiple databases, you will have to specify both the *`link_id`* and *`transaction_id`* in calls to ibase_query() and ibase_prepare().

Parameters

`trans_args`
: *`trans_args`* can be a combination of **IBASE_READ**, **IBASE_WRITE**, **IBASE_COMMITTED**, **IBASE_CONSISTENCY**, **IBASE_CONCURRENCY**, **IBASE_REC_VERSION**, **IBASE_REC_NO_VERSION**, **IBASE_WAIT** and **IBASE_NOWAIT**.

`link_identifier`
: An InterBase link identifier. If omitted, the last opened link is assumed.

Return Values

Returns a transaction handle, or **FALSE** on error.

ibase_wait_event

ibase_wait_event -- Wait for an event to be posted by the database

Description

string **ibase_wait_event** (string `$event_name1` [, string `$event_name2` [, string `$...`]])

string **ibase_wait_event** (resource `$connection`, string `$event_name1` [, string `$event_name2` [, string `$...`]])

This function suspends execution of the script until one of the specified events is posted by the database. The name of the event that was posted is returned. This function accepts up to 15 event arguments.

Parameters

`event_name1`
The event name.

`event_name2`

`...`

Return Values

Returns the name of the event that was posted.

See Also

- ibase_set_event_handler()
- ibase_free_event_handler()