Sybase

Introduction

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

To enable Sybase-DB support configure PHP *--with-sybase[=DIR]*. DIR is the Sybase home directory, defaults to */home/sybase*. To enable Sybase-CT support configure PHP *--with-sybase-ct[=DIR]*. DIR is the Sybase home directory, defaults to */home/sybase*.

Runtime Configuration

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

Sybase configuration options

Name	Default	Changeable	Changelog
sybase.allow_persistent	"1"	PHP_INI_ALL	PHP_INI_ALL in PHP <= 4.0.2. PHP_INI_SYSTEM in PHP <= 4.0.3.
sybase.max_persistent	"-1"	PHP_INI_ALL	PHP_INI_ALL in PHP <= 4.0.2. PHP_INI_SYSTEM in PHP <= 4.0.3.
sybase.max_links	"-1"	PHP_INI_ALL	PHP_INI_ALL in PHP <= 4.0.2. PHP_INI_SYSTEM in PHP <= 4.0.3.
sybase.interface_file	"/usr/sybase/interfaces"	PHP_INI_SYSTEM	
sybase.min_error_severity	"10"	PHP_INI_ALL	
sybase.min_message_severity	"10"	PHP_INI_ALL	

sybase.compatability_mode	"0"	PHP_INI_ALL	
magic_quotes_sybase	"0"	PHP_INI_ALL	Removed in PHP 6.0.0.

Here's a short explanation of the configuration directives.

sybase.allow_persistent boolean
Whether to allow persistent Sybase connections.

sybase.max_persistent integer
The maximum number of persistent Sybase connections per process. -1 means no limit.

sybase.max_links integer
The maximum number of Sybase connections per process, including persistent connections. -1 means no limit.

sybase.min_error_severity integer
Minimum error severity to display.

sybase.min_message_severity integer
Minimum message severity to display.

magic_quotes_sybase boolean
If *magic_quotes_sybase* is on, a single-quote is escaped with a single-quote instead of a backslash if magic_quotes_gpc or magic_quotes_runtime are enabled.

Note
Note that when *magic_quotes_sybase* is ON it completely overrides *magic_quotes_gpc*. In this case even when *magic_quotes_gpc* is enabled neither double quotes, backslashes or NUL's will be escaped.

Sybase-CT configuration options

Name	Default	Changeable	Changelog
sybct.allow_persistent	"1"	PHP_INI_SYSTEM	PHP_INI_ALL in PHP <= 4.0.2. Available since PHP 4.0.2. Removed in PHP 4.0.3.

sybct.max_persistent	"-1"	PHP_INI_SYSTEM	PHP_INI_ALL in PHP <= 4.0.2. Available since PHP 4.0.2. Removed in PHP 4.0.3.
sybct.max_links	"-1"	PHP_INI_SYSTEM	PHP_INI_ALL in PHP <= 4.0.2. Available since PHP 4.0.2. Removed in PHP 4.0.3.
sybct.min_server_severity	"10"	PHP_INI_ALL	Available since PHP 4.0.2. Removed in PHP 4.0.3.
sybct.min_client_severity	"10"	PHP_INI_ALL	Available since PHP 4.0.2. Removed in PHP 4.0.3.
sybct.hostname	NULL	PHP_INI_ALL	Available since PHP 4.0.2. Removed in PHP 4.0.3.
sybct.deadlock_retry_count	"0"	PHP_INI_ALL	Available since PHP 4.3.0.

Here's a short explanation of the configuration directives.

sybct.allow_persistent boolean
: Whether to allow persistent Sybase-CT connections. The default is on.

sybct.max_persistent integer
: The maximum number of persistent Sybase-CT connections per process. The default is -1 meaning unlimited.

sybct.max_links integer
: The maximum number of Sybase-CT connections per process, including persistent connections. The default is -1 meaning unlimited.

sybct.min_server_severity integer
: Server messages with severity greater than or equal to sybct.min_server_severity will be reported as warnings. This value can also be set from a script by calling sybase_min_server_severity(). The default is 10 which reports errors of information severity or greater.

sybct.min_client_severity integer
: Client library messages with severity greater than or equal to sybct.min_client_severity will be reported as warnings. This value can also be set from a script by calling

sybase_min_client_severity(). The default is 10 which effectively disables reporting.

sybct.login_timeout integer
: The maximum time in seconds to wait for a connection attempt to succeed before returning failure. Note that if max_execution_time has been exceeded when a connection attempt times out, your script will be terminated before it can take action on failure. The default is one minute.

sybct.timeout integer
: The maximum time in seconds to wait for a select_db or query operation to succeed before returning failure. Note that if max_execution_time has been exceeded when an operation times out, your script will be terminated before it can take action on failure. The default is no limit.

sybct.hostname string
: The name of the host you claim to be connecting from, for display by sp_who. The default is none.

sybct.deadlock_retry_count int
: Allows you to define how often deadlocks are to be retried. The default is -1, or "forever".

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

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Sybase Functions

sybase_affected_rows

sybase_affected_rows -- Gets number of affected rows in last query

Description

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

sybase_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query on the server associated with the specified link identifier.

This command is not effective for SELECT statements, only on statements which modify records. To retrieve the number of rows returned from a SELECT, use sybase_num_rows().

Parameters

`link_identifier`
If the link identifier isn't specified, the last opened link is assumed.

Return Values

Returns the number of affected rows, as an integer.

Examples

Example #1 - Delete-Query

```
<?php
/* connect to database */
sybase_connect('SYBASE', '', '') or
die("Could not connect");
sybase_select_db("db");

sybase_query("DELETE FROM sometable WHERE id < 10");
printf("Records deleted: %d\n", sybase_affected_rows());
?>
```

The above example will output:

```
Records deleted: 10
```

See Also

- sybase_num_rows()

sybase_close

sybase_close -- Closes a Sybase connection

Description

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

sybase_close() closes the link to a Sybase database that's associated with the specified link *`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.

sybase_close() will not close persistent links generated by sybase_pconnect().

Parameters

`link_identifier`
: If the link identifier isn't specified, the last opened link is assumed.

Return Values

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

See Also

- sybase_connect()
- sybase_pconnect()

sybase_connect

sybase_connect -- Opens a Sybase server connection

Description

resource **sybase_connect** ([string `$servername` [, string `$username` [, string `$password` [, string `$charset` [, string `$appname`]]]]])

sybase_connect() establishes a connection to a Sybase server.

In case a second call is made to sybase_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 sybase_close().

Parameters

`servername`
The servername argument has to be a valid servername that is defined in the 'interfaces' file.

`username`
Sybase user name

`password`
Password associated with *`username`*.

`charset`
Specifies the charset for the connection

`appname`

Return Values

Returns a positive Sybase link identifier on success, or **FALSE** on failure.

ChangeLog

Version	Description

4.0.2	The *charset* parameter was added.

Examples

Example #2 - sybase_connect() example

```
<?php
$link = sybase_connect('SYBASE', '', '')
       or die("Could not connect !");
echo "Connected successfully";
sybase_close($link);
?>
```

See Also

- sybase_pconnect()
- sybase_close()

sybase_data_seek

sybase_data_seek -- Moves internal row pointer

Description

bool **sybase_data_seek** (resource `$result_identifier`, int `$row_number`)

sybase_data_seek() moves the internal row pointer of the Sybase result associated with the specified result identifier to pointer to the specified row number. The next call to sybase_fetch_row() would return that row.

Parameters

`result_identifier`

`row_number`

Return Values

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

See Also

- sybase_fetch_row()

sybase_deadlock_retry_count

sybase_deadlock_retry_count -- Sets the deadlock retry count

Description

void **sybase_deadlock_retry_count** (int `$retry_count`)

Using sybase_deadlock_retry_count(), the number of retries can be defined in cases of deadlocks. By default, every deadlock is retried an infinite number of times or until the process is killed by Sybase, the executing script is killed (for instance, by set_time_limit()) or the query succeeds.

Parameters

`retry_count`

Values for retry_count

-1	Retry forever (default)
0	Do not retry
n	Retry n times

Return Values

No value is returned.

Notes

Note
This function is only available when using the CT library interface to Sybase, and not with the DB library.

sybase_fetch_array

sybase_fetch_array -- Fetch row as array

Description

array **sybase_fetch_array** (resource `$result`)

sybase_fetch_array() is an extended version of sybase_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using sybase_fetch_array() is NOT significantly slower than using sybase_fetch_row(), while it provides a significant added value.

Parameters

result

Return Values

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

Note
When selecting fields with identical names (for instance, in a join), the associative indices will have a sequential number prepended. See the example for details.

Examples

Example #3 - Identical fieldnames

```
<?php

$dbh = sybase_connect('SYBASE', '', '');
$q = sybase_query('SELECT * FROM p, a WHERE p.person_id= a.person_id');
var_dump(sybase_fetch_array($q));
sybase_close($dbh);
?>
```

The above example would produce the following output (assuming the two tables only have each one column called "person_id"):

```
array(4) {
 [0]=>
 int(1)
 ["person_id"]=>
 int(1)
 [1]=>
 int(1)
 ["person_id1"]=>
 int(1)
}
```

See Also

- sybase_fetch_row()
- sybase_fetch_assoc()
- sybase_fetch_object()

sybase_fetch_assoc

sybase_fetch_assoc -- Fetch a result row as an associative array

Description

array **sybase_fetch_assoc** (resource `$result`)

sybase_fetch_assoc() is a version of sybase_fetch_row() that uses column names instead of integers for indices in the result array. Columns from different tables with the same names are returned as name, name1, name2, ..., nameN.

An important thing to note is that using sybase_fetch_assoc() is NOT significantly slower than using sybase_fetch_row(), while it provides a significant added value.

Parameters

`result`

Return Values

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

Notes

Note
This function is only available when using the CT library interface to Sybase, and not with the DB library.

See Also

- sybase_fetch_row()
- sybase_fetch_array()
- sybase_fetch_object()

sybase_fetch_field

sybase_fetch_field -- Get field information from a result

Description

object **sybase_fetch_field** (resource `$result` [, int `$field_offset`])

sybase_fetch_field() can be used in order to obtain information about fields in a certain query result.

Parameters

`result`

`field_offset`
If the field offset isn't specified, the next field that wasn't yet retrieved by sybase_fetch_field() is retrieved.

Return Values

Returns an object containing field information.

The properties of the object are:

- name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.
- column_source - the table from which the column was taken
- max_length - maximum length of the column
- numeric - 1 if the column is numeric
- type - datatype of the column

See Also

- sybase_field_seek()

sybase_fetch_object

sybase_fetch_object -- Fetch a row as an object

Description

object **sybase_fetch_object** (resource `$result` [, mixed `$object`])

sybase_fetch_object() is similar to sybase_fetch_assoc(), with one difference - an object is returned, instead of an array.

Speed-wise, the function is identical to sybase_fetch_array(), and almost as quick as sybase_fetch_row() (the difference is insignificant).

Parameters

`result`

`object`
Use the second *`object`* to specify the type of object you want to return. If this parameter is omitted, the object will be of type stdClass.

Return Values

Returns an object with properties that correspond to the fetched row, or **FALSE** if there are no more rows.

ChangeLog

Version	Description

4.3.0	This function will no longer return numeric object members. Old behaviour:
	`object(stdclass)(3) {`
	`[0]=>`
	`string(3) "foo"`
	`["foo"]=>`
	`string(3) "foo"`
	`[1]=>`
	`string(3) "bar"`
	`["bar"]=>`
	`string(3) "bar"`
	`}`
	New behaviour:

```
object(stdclass)(3) {
 ["foo"]=>
 string(3) "foo"
 ["bar"]=>
 string(3) "bar"
}
```

Examples

Example #4 - sybase_fetch_object() return as Foo

```
<?php
   class Foo {
       var $foo, $bar, $baz;
   }

   // {...]
   $qrh= sybase_query('SELECT foo, bar, baz FROM example');
   $foo= sybase_fetch_object($qrh, 'Foo');
   $bar= sybase_fetch_object($qrh, new Foo());
   // {...]
?>
```

See Also

- sybase_fetch_array()
- sybase_fetch_row()

sybase_fetch_row

sybase_fetch_row -- Get a result row as an enumerated array

Description

array **sybase_fetch_row** (resource `$result`)

sybase_fetch_row() fetches one row of data from the result associated with the specified result identifier.

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

Parameters

`result`

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.

Data types

PHP	Sybase
string	VARCHAR, TEXT, CHAR, IMAGE, BINARY, VARBINARY, DATETIME
int	NUMERIC (w/o precision), DECIMAL (w/o precision), INT, BIT, TINYINT, SMALLINT
float	NUMERIC (w/ precision), DECIMAL (w/ precision), REAL, FLOAT, MONEY
NULL	NULL

See Also

- sybase_fetch_array()

- sybase_fetch_assoc()
- sybase_fetch_object()
- sybase_data_seek()
- sybase_result()

sybase_field_seek

sybase_field_seek -- Sets field offset

Description

bool **sybase_field_seek** (resource `$result`, int `$field_offset`)

Seeks to the specified field offset. If the next call to sybase_fetch_field() won't include a field offset, this field would be returned.

Parameters

`result`

`field_offset`

Return Values

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

See Also

- sybase_fetch_field()

sybase_free_result

sybase_free_result -- Frees result memory

Description

bool **sybase_free_result** (resource `$result`)

sybase_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script ends. You may call sybase_free_result() with the result identifier as an argument and the associated result memory will be freed.

Parameters

`result`

Return Values

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

sybase_get_last_message

sybase_get_last_message -- Returns the last message from the server

Description

string **sybase_get_last_message** (void)

sybase_get_last_message() returns the last message reported by the server.

Return Values

Returns the message as a string.

See Also

- sybase_min_message_severity()

sybase_min_client_severity

sybase_min_client_severity -- Sets minimum client severity

Description

void **sybase_min_client_severity** (int `$severity`)

sybase_min_client_severity() sets the minimum client severity level.

Parameters

`severity`

Return Values

No value is returned.

Notes

Note
This function is only available when using the CT library interface to Sybase, and not with the DB library.

See Also

- sybase_min_server_severity()

sybase_min_error_severity

sybase_min_error_severity -- Sets minimum error severity

Description

void **sybase_min_error_severity** (int `$severity`)

sybase_min_error_severity() sets the minimum error severity level.

Parameters

`severity`

Return Values

No value is returned.

Notes

Note
This function is only available when using the CT library interface to Sybase, and not with the DB library.

See Also

- sybase_min_message_severity()

sybase_min_message_severity

sybase_min_message_severity -- Sets minimum message severity

Description

void **sybase_min_message_severity** (int `$severity`)

sybase_min_message_severity() sets the minimum message severity level.

Parameters

`severity`

Return Values

No value is returned.

Notes

Note
This function is only available when using the DB library interface to Sybase, and not with the CT library.

See Also

- sybase_min_error_severity()

sybase_min_server_severity

sybase_min_server_severity -- Sets minimum server severity

Description

void **sybase_min_server_severity** (int `$severity`)

sybase_min_server_severity() sets the minimum server severity level.

Parameters

`severity`

Return Values

No value is returned.

Notes

Note
This function is only available when using the CT library interface to Sybase, and not with the DB library.

See Also

- sybase_min_client_severity()

sybase_num_fields

sybase_num_fields -- Gets the number of fields in a result set

Description

int **sybase_num_fields** (resource `$result`)

sybase_num_fields() returns the number of fields in a result set.

Parameters

`result`

Return Values

Returns the number of fields as an integer.

See Also

- sybase_query()
- sybase_fetch_field()
- sybase_num_rows()

sybase_num_rows

sybase_num_rows -- Get number of rows in a result set

Description

int **sybase_num_rows** (resource `$result`)

sybase_num_rows() returns the number of rows in a result set.

Parameters

`result`

Return Values

Returns the number of rows as an integer.

See Also

- sybase_num_fields()
- sybase_query()
- sybase_fetch_row()

sybase_pconnect

sybase_pconnect -- Open persistent Sybase connection

Description

resource **sybase_pconnect** ([string `$servername` [, string `$username` [, string `$password` [, string `$charset` [, string `$appname`]]]]])

sybase_pconnect() acts very much like sybase_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 (sybase_close() will not close links established by sybase_pconnect()).

This type of links is therefore called 'persistent'.

Parameters

`servername`
The servername argument has to be a valid servername that is defined in the 'interfaces' file.

`username`
Sybase user name

`password`
Password associated with *`username`*.

`charset`
Specifies the charset for the connection

`appname`

Return Values

Returns a positive Sybase persistent link identifier on success, or **FALSE** on error.

ChangeLog

Version	Description
4.0.2	The *charset* parameter was added.

See Also

- sybase_connect()

sybase_query

sybase_query -- Sends a Sybase query

Description

mixed **sybase_query** (string `$query` [, resource `$link_identifier`])

sybase_query() sends a query to the currently active database on the server that's associated with the specified link identifier.

Parameters

`query`

`link_identifier`
If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if sybase_connect() was called, and use it.

Return Values

Returns a positive Sybase result identifier on success, **FALSE** on error, or **TRUE** if the query was successful but didn't return any columns.

See Also

- sybase_select_db()
- sybase_connect()

sybase_result

sybase_result -- Get result data

Description

string **sybase_result** (resource `$result`, int `$row`, mixed `$field`)

Returns the contents of the cell at the row and offset in the specified Sybase result set.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than sybase_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: sybase_fetch_row(), sybase_fetch_array() and sybase_fetch_object().

Parameters

result

row

field
The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

Return Values

sybase_result() returns the contents of one cell from a Sybase result set.

sybase_select_db

sybase_select_db -- Selects a Sybase database

Description

bool **sybase_select_db** (string `$database_name` [, resource `$link_identifier`])

sybase_select_db() sets the current active database on the server that's associated with the specified link identifier.

Every subsequent call to sybase_query() will be made on the active database.

Parameters

`database_name`

`link_identifier`
If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if sybase_connect() was called, and use it.

Return Values

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

See Also

- sybase_connect()
- sybase_pconnect()
- sybase_query()

sybase_set_message_handler

sybase_set_message_handler -- Sets the handler called when a server message is raised

Description

bool **sybase_set_message_handler** (callback `$handler` [, resource `$connection`])

sybase_set_message_handler() sets a user function to handle messages generated by the server. You may specify the name of a global function, or use an array to specify an object reference and a method name.

Parameters

`handler`
The handler expects five arguments in the following order: message number, severity, state, line number and description. The first four are integers. The last is a string. If the function returns **FALSE**, PHP generates an ordinary error message.

`connection`

Return Values

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

ChangeLog

Version	Description
4.3.5	The *`connection`* parameter was added.

Examples

Example #5 - sybase_set_message_handler() callback function

```
<?php
   function msg_handler($msgnumber, $severity, $state, $line, $text)
   {
       var_dump($msgnumber, $severity, $state, $line, $text);
   }
```

```
   sybase_set_message_handler('msg_handler');
?>
```

Example #6 - sybase_set_message_handler() callback to a class

```
<?php
   class Sybase {
       function handler($msgnumber, $severity, $state, $line, $text)
       {
           var_dump($msgnumber, $severity, $state, $line, $text);
       }
   }

   $sybase= new Sybase();
   sybase_set_message_handler(array($sybase, 'handler'));
?>
```

Example #7 - sybase_set_message_handler() unhandled messages

```
<?php
   // Return FALSE from this function to indicate you can't handle
   // this. The error is printed out as a warning, the way you're used
   // to it if there is no handler installed.
   function msg_handler($msgnumber, $severity, $state, $line, $text)
   {
       if (257 == $msgnumber) {
           return false;
       }
       var_dump($msgnumber, $severity, $state, $line, $text);
   }

   sybase_set_message_handler('msg_handler');
?>
```

Notes

Note

This function is only available when using the CT library interface to Sybase, and not with the DB library.

sybase_unbuffered_query

sybase_unbuffered_query -- Send a Sybase query and do not block

Description

resource **sybase_unbuffered_query** (string `$query`, resource `$link_identifier` [, bool `$store_result`])

sybase_unbuffered_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if sybase_connect() was called, and use it.

Unlike sybase_query(), sybase_unbuffered_query() reads only the first row of the result set. sybase_fetch_array() and similar function read more rows as needed. sybase_data_seek() reads up to the target row. The behavior may produce better performance for large result sets.

sybase_num_rows() will only return the correct number of rows if all result sets have been read. To Sybase, the number of rows is not known and is therefore computed by the client implementation.

Note
If you don't read all of the resultsets prior to executing the next query, PHP will raise a warning and cancel all of the pending results. To get rid of this, use sybase_free_result() which will cancel pending results of an unbuffered query.

Parameters

`query`

`link_identifier`

`store_result`
The optional *`store_result`* can be **FALSE** to indicate the resultsets shouldn't be fetched into memory, thus minimizing memory usage which is particularly interesting with very large resultsets.

Return Values

Returns a positive Sybase result identifier on success, or **FALSE** on error.

Examples

Example #8 - sybase_unbuffered_query() example

```
<?php

$dbh = sybase_connect('SYBASE', '', '');
$q = sybase_unbuffered_query('select firstname, lastname from huge_table',
$dbh, false);
sybase_data_seek($q, 10000);
$i = 0;

while ($row = sybase_fetch_row($q)) {
   echo $row[0], ' ', $row[1], '<br />';
   if ($i++ > 40000) {
       break;
   }
}

sybase_free_result($q);
sybase_close($dbh);

?>
```

Notes

Note

This function is only available when using the CT library interface to Sybase, and not with the DB library.

See Also

- sybase_query()