SYBASE®

An SAP® Company

Programmers Guide

Adaptive Server® Enterprise Extension Module for PHP 15.7 SP100

DOCUMENT ID: DC01693-01-1570100-01
LAST REVISED: May 2013

Contents

Adaptive Server Enterprise Extension Module for PHP

The Sybase® Adaptive Server® Enterprise extension module for the PHP scripting language allows PHP developers to execute queries against an Adaptive Server database.

The extension module for PHP is a vendor-specific database interface driver that executes queries against an Adaptive Server database. The open source programming language, PHP (Hypertext Preprocessor) provides the ability to retrieve information from common databases. The Adaptive Server extension module for the PHP scripting language includes the necessary PHP APIs to enable PHP developers to write standalone scripts to execute queries against an Adaptive Server database.

The extension module for PHP has been tested against the command line PHP implementation. Usage of the extension, such as a CGI module in a Web server, is beyond the scope of Sybase testing.

Required Components

Access to an Adaptive Server database using the PHP programming language requires the following components:

- PHP script – application script to connect to an Adaptive Server database server.
- Adaptive Server Enterprise extension module for PHP – is the vendor-specific driver supported by Sybase. The extension is called through the PHP script and connects to the API layer DBCAPI.
- DBCAPI – is a function library that acts as intermediate conversion layer between the PHP extension and the CT-Library.
- CT-Library – is a Sybase Open Client library that you can use to send commands to Adaptive Server, process the results and return the data.

Version Requirements

For information about platform support, see the *Software Developers Kit and Open Server Installation Guide* for your platform.

- Adaptive Server Enterprise – version 15.7 or later. The PHP driver has been developed and tested against version 15.7. However, the driver can connect to Adaptive Server version earlier than 15.7.
- PHP installation – version 5.3.6.
- CT-Library from Open Client SDK – version 15.7
- DBCAPI — Sybase recommends that you use the DBCAPI library and the PHP driver from the same SDK installation.

Installing the Extension Module for PHP

The extension module for PHP is a component you can install through the Sybase Installer.

The extension module for PHP is an optional installation component when you choose Custom as the installation type. The extension is installed by default if the installation type you choose is Typical or Full.

This is an overview of the extension module installation; for complete installation instructions, see the *Software Developers Kit and Open Server Installation Guide* for your platform.

- Pre-installation requirements:
  - PHP installation 5.3.6 – 64-bit only.
  - Open Client SDK is required. The DBCAPI library is included in the Open Client SDK and is installed together with the core Open Client (CT-Library).

  Do not install DBCAPI as part of the driver.
- The extension module for PHP is installed as a PHP extension in a dynamic library.

  The Sybase installer installs the extension module into the Open Client (CT-Library) directory as one shared library called `sybaseasephp.so`.
- These files are installed with the extension module:

  ```
  $SYBASE/$SYBASE_OCS/php/php536_64/lib/sybaseasephp.so
  $SYBASE/$SYBASE_OCS/php/php536_64/devlib/sybaseasephp.so
  $SYBASE/$SYBASE_OCS/sample/php/README
  $SYBASE/$SYBASE_OCS/sample/php/firstapp.php
  $SYBASE/$SYBASE_OCS/sample/php/dsnconnect.php
  ```

Configuration Overview

Configure your environment to locate the extension module and OCS installations.

Environment Variables

To successfully use the extension module, set Sybase environment variables in the environment in which you run the PHP executable. At a minimum, you must set the $SYBASE and $SYBASE_OCS environment variables. Set the *LD_LIBRARY_PATH* variable to point to the correct location under the `$SYBASE/$SYBASE_OCS/lib` directory. The extension module loads the DBCAPI library from this location.

PHP Configuration

A system administrator can copy the extension module into a different extension specific directory and load the extension module from the directory by modifying the `php.ini` file, for example:

```
; Set the default extension directory.
extension_dir = "/usr/local/lib/php/extensions"
```

```
;Load the Sybase ASE PHP driver: sybaseasephp
extension="sybaseasephp.so"
```

Once the file is modified, you can copy the `$SYBASE/$SYBASE_OCS/php/php536_64/lib/sybaseasephp.so` or `$SYBASE/$SYBASE_OCS/php/php536_64/devlib/sybaseasephp.so` library to `/usr/local/lib/php/extensions`.

Extension Libraries
When executing `php -m` you should see the `sybaseasephp` extension listed among the extensions active in the PHP installation.

Note: You need not place all extension libraries of a particular PHP installation in the same directory.

Sample PHP Scripts

Sample PHP scripts are installed in `$SYBASE/$SYBASE_OCS/sample/php/`

For information about running the sample scripts, see the README file located in the same directory. Also see the *Software Developers Kit and Open Server Installation Guide* for platform installation information.

The PHP sample scripts are:

- **firstapp.php** – a simple example program that connects to a server, sends a select query, and prints the rows.
- **dsnconnect.php** – a simple example program that connects to a server using a DSN connection string. It optionally prints out the servername, account, and the current database.

Support for DSN-style Connection Properties

sybase_connect() and **sybase_pconnect()** APIs support DSN-style connection properties.

When you call either **sybase_connect()** or **sybase_pconnect()** APIs using only the **servername** parameter, **servername** must contain a valid DSN (data source name) string. The data source name is a semicolon (;) separated string of name=value parts as explained as follows:

1. Name – a case-insensitive value that can be delimited by an equal sign (=) or semicolon (;). An attribute can have multiple synonyms. For example, **server** and **servername** refer to the same attribute.
2. Equals sign (=) – indicates the start of the value to be assigned to the Name. If there is no equals sign, the Name is assumed to be of boolean type with a value of true.

3. Value – a string that is terminated by a semicolon (;). Use a backslash (\) if a semicolon or another back slash is present in the value. Values can be of type boolean, integer, or string. Valid values for Boolean types are true, false, on, off, 1, and 0.

Note: If a boolean name is present without a value, the Boolean type must be set to true.

For example:

```
Username=name;Password=pwd;Timeout=10
```

Valid Attribute Names and Values

The table lists the valid attribute names and values for the **dsn** keyword argument.

Name	Description	Value
ANSINull	Determines whether evaluation of NULL-valued operands in SQL equality (=) or inequality (!=) comparisons is ANSI-compliant. If the value is true, Adaptive Server enforces the ANSI behavior that *= NULL* and *is NULL* are not equivalent. In standard Transact-SQL, *= NULL* and *is NULL* are considered to be equivalent. This option affects *<> NULL* and *is not NULL* behavior in a similar fashion.	Boolean value. The default is false.
BulkLogin	Determines whether a connection is enabled to perform a bulk-copy operation.	Boolean value. The default is false.
ChainXacts	If true, Adaptive Server uses chained transaction behavior, that is, each server command is considered to be a distinct transaction. Adaptive Server implicitly executes a begin transaction before any of these statements: **delete**, **fetch**, **insert**, **open**, **select**, and **update**. You must still explicitly end or roll back the transaction. If false, an application must specify explicit begin transaction statements paired with commit or rollback statements.	Boolean value. The default is false.
Charset	Specifies the **charset** to be used on this connection.	String value.
Confidentiality	Whether data encryption service is performed on the connection.	Boolean value. The default is false.

Name	Description	Value
CredentialDelegation	Determines whether to allow the server to connect to a second server with the user's delegated credentials.	Boolean value. The default is false.
DetectReplay	Determines whether the connection's security mechanism detects replayed transmissions.	Boolean value. The default is false.
DetectOutOfSequence	Determines whether the connection's security mechanism detects transmissions that arrive out of sequence.	Boolean value. The default is false.
Integrity	Determines whether the connection's security mechanism performs data integrity checking.	Boolean value. The default is false.
Interfaces	The path and name of the interfaces file.	String value.
Keytab	The name and path to the file from which a connection's security mechanism reads the security key to go with the *username* value.	String value. The default is NULL, that is, the user must have established credentials before connecting.
Locale	Determines which language and character set to use for messages, datatype conversions, and datetime formats.	String value.
Language	Determines which language set to use for messages, datatype conversions, and datetime formats.	String value.
LoginTimeout	Specifies the login timeout value.	Integer value.
MaxConnect	Specifies the maximum number of simultaneously open connections that a context may have.	Integer value. Default value is 25. Negative and zero values are not allowed.
MutualAuthentication	Determines whether the server is required to authenticate itself to the client.	Boolean value. The default is false.

Name	Description	Value
NetworkAuthentication	Determines whether the connection's security mechanism performs network-based user authentication.	Boolean value. The default is false.
PacketSize	Specifies the TDS packet size.	Integer value.
Password	Specifies the password used to log in to the server.	String value.
PasswordEncryption	Determines whether the connection uses asymmetrical password encryption.	Boolean value. The default is false.
SecurityMechanism	Specifies the name of the network security mechanism that performs security services for the connection.	String value. The default value depends on security driver configuration.
Server **Servername**	Specifies the name of the server to which you are connected.	String value.
ServerPrincipalName	Specifies the network security principal name for the server to which a connection is opened.	String value. The default is NULL, which means that the connection assumes the server principal name is the same as its *ServerName* value.
Keepalive	Determines whether to use the KEEPALIVE option.	Boolean value. The default is true.
Timeout	Specifies the connection timeout value.	Integer value.
UID **User** **Username**	Specifies the name used to log in to the server.	String value.

dsnconnect.php Sample Program

The `dsnconnect.php` sample program connects to a server using a DSN connection string. It optionally prints the server name, the user account, and the current database.

Extension Module for PHP API Reference

The extension module interface API.

sybase_affected_rows

Returns the number of rows affected by the last **INSERT**, **UPDATE**, or **DELETE** query on the connection referred to by $conn.

This function is typically used for **INSERT**, **UPDATE**, or **DELETE** statements. For **SELECT** statements, use the **sybase_num_rows()** function instead.

Syntax

```
int sybase_affected_rows([resource $conn])
```

Parameters

- **$conn –** the connection resource returned by a connection opening function. If the connection resource isn't specified, the most recently opened connection is used.

Returns

The number of rows affected by the last **INSERT**, **UPDATE**, or **DELETE** query.

sybase_close

Closes a connection.

Syntax

```
bool sybase_close([resource $conn])
```

Parameters

- **$conn –** the connection resource returned by a connection opening function. If the connection resource is not specified, the most recently opened connection is closed.

Returns

TRUE on success.

FALSE on failure.

sybase_connect

Opens a connection to Adaptive Server.

sybase_connect() supports DSN style connection properties. When it is called with the parameter *servername* only, *servername* is expected to contain a valid DSN string.

Syntax

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

Parameters

- **$servername –** when **sybase_connect()** is called with more than this parameter, then **$servername** contains a server defined in the relevant Sybase directory service. When **sybase_connect()** is called with only this parameter, then **$servername** contains a valid DSN connection string of syntax: `param1=val1;param2=val2;...`
- **$username –** the user account used to log in to Adaptive Server.
- **$password –** the password of the user account with which to log in to Adaptive Server.

Returns

A positive connection identifier on success.

FALSE on failure.

sybase_data_seek

Moves the internal row pointer on the result set associated with the result identifier to point to the specified row number.

Syntax

```
bool sybase_data_seek(resource $result, int $row)
```

Parameters

- **$result –** the result resource that comes from a call to **sybase_query().**
- **$row –** the row number (starting with 0) to set the internal pointer to.

Returns

TRUE – internal pointer has been successfully positioned.

FALSE – failed to set the internal pointer correctly.

sybase_fetch_array

Fetches a result row as an associative array, a numeric array, or both.

Syntax

```
mixed sybase_fetch_array(resource $result
[, int $result_type ] )
```

Parameters

- **$result –** the result resource that comes from a call to **sybase_query()**..
- **$result_type –** the type of array to be returned. It accepts the following values to return arrays with:

  SYB_FETCH_ASSOC – associative array

  SYB_FETCH_NUM – numeric array

  SYB_FETCH_BOTH – (Default) both associative & number indices

Returns

Associative array (identical to **sybase_fetch_assoc()**) when $result_type is SYB_FETCH_ASSOC or SYB_FETCH_BOTH.

Numeric array (identical to **sybase_fetch_row()**) when $result_type is SYB_FETCH_NUM or SYB_FETCH_BOTH.

FALSE – there were no more rows to be fetched.

sybase_fetch_assoc

Fetches one row of data from the result set associated with the specified result identifier in an associative array.

Positions the internal pointer one row farther in the result set. A subsequent call to **sybase_fetch_assoc()** returns the next row in the result set, or FALSE if there are no more rows.

Syntax

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

Parameters

- **$result –** the result resource that comes from a call to **sybase_query()**.

Returns

On success, returns the next row in the result set.

FALSE – there were no more rows to be fetched.

sybase_fetch_field

Returns an object containing field information.

This function can be used to obtain information about fields in the provided query result.

Syntax

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

Parameters

- **$result –** the result resource that comes from a call to **sybase_query()**.
- **$field_offset –** the field number (starting from 0) to retrieve information from. If the field offset is not specified, the next field that is not yet retrieved by this function is used.

Returns

Returns an object with the following field information as properties:

Field name	Field type	Field description
name	string	Column name.
table	string	The table from which the column was taken.
max_length	int	Maximum length of the column.
type	string	Data type of the column, as defined in `cspublic.h`.

sybase_fetch_object

Fetches a row as an object.

This function is similar to **sybase_fetch_assoc()**, but it returns an object rather than an array. It positions the internal pointer one row farther in the result set.

Syntax

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

Parameters

- **$result –** the result resource that comes from a call to **sybase_query()**.

- **$object –** optionally, specify the type of object to be returned. The default object type is stdClass.

Returns

Returns an object with properties that correspond to the fetched row's field names.

FALSE – there were no more rows to be fetched.

sybase_fetch_row

Fetches one row of data from the result set associated with the specified result identifier in a numerical array.

It positions the internal pointer one row farther in the result set.

A subsequent call to **sybase_fetch_row()** returns the next row in the result set, or FALSE if there are no more rows.

Syntax

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

Parameters

- **$result –** the result resource that comes from a call to **sybase_query()**.

Returns

Numerical array (starting at 0) – identical to **sybase_fetch_array($result, SYB_FETCH_NUM)**

FALSE – there are no more rows to be fetched.

sybase_field_seek

Sets the internal pointer to the requested field offset.

If the next call to **sybase_fetch_field()** does not specify a field offset, the field internal pointer is set to, can be used.

Syntax

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

Parameters

- **$result –** the result resource that comes from a call to **sybase_query()**.
- **$field_offset –** the field offset (starting at 0).

Returns

TRUE – internal pointer set correctly.

FALSE – failed to set internal pointer correctly.

sybase_free_result

Frees all memory associated with this result set.

Although result memory is automatically freed when the PHP script ends. Sybase recommends, as a good programming practice, that you free memory that is no longer needed.

Syntax

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

Parameters

- **$result –** the result resource that comes from a call to **sybase_query()**.

Returns

TRUE – memory freed successfully.

FALSE – failed to free the memory.

sybase_get_last_message

The last message returned by the Server.

Syntax

```
string sybase_get_last_message(void)
```

Parameters

None

Returns

The last message returned by the server.

FALSE – failed to retrieve the last server message.

sybase_get_last_status

Returns the last status result that was sent on the connection $conn.

Syntax

```
int sybase_get_last_status(resource $conn)
```

Parameters

- **$conn –** the connection resource returned by a connection opening function.

Returns

The last status result that was sent on the connection $conn.

sybase_next_result

Returns a result set identifier pointing to the next result set on connection $conn.

Syntax

```
mixed sybase_next_result(resource $conn)
```

Parameters

- **$conn –** the connection resource returned by a connection opening function.

Returns

A positive Sybase result set identifier on success.

FALSE – there is no further result set on the connection.

sybase_num_fields

Returns the number of fields in the result set.

Syntax

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

Parameters

- **$result –** the result resource that comes from a call to **sybase_query()**.

Returns

The number of fields in the result set.

FALSE – failed to retrieve the number of fields.

sybase_num_rows

Returns the number of rows in the result set of a SELECT statement.

sybase_num_rows() returns the correct number of rows when the complete result set has been read.

Syntax

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

Parameters

- **$result –** the result resource that comes from a call to **sybase_query()**.

Returns

The number of rows in the result set.

FALSE – failed to retrieve the number of rows.

sybase_pconnect

Opens a persistent connection to Adaptive Server.

sybase_pconnect() supports DSN style connection properties. When it is called with the parameter *servername* only, *servername* is expected to contain a valid DSN string.

Note: If a persistent connection has been previously opened with the same arguments as this call, an identifier for the existing connection gets returned instead of opening a new connection.

Syntax

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

Parameters

- **$servername –** when **sybase_connect()** is called with more than this parameter, then **$servername** contains a server defined in the relevant Sybase directory service. When **sybase_connect()** is called with only this parameter, then **$servername** contains a valid DSN connection string of syntax: `param1=val1;param2=val2;...`
- **$username –** the name of the user account used to login to Adaptive Server.
- **$password –** the password of the user account used to login to Adaptive Server.

Returns

A positive connection identifier on success.

FALSE – on failure.

sybase_query

Sends a query to the connection.

Syntax

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

Parameters

- **$query –** a string containing the query to be sent to the Adaptive Server.
- **$conn –** the connection resource returned by a connection opening function. If the connection resource is not specified, the most recently opened connection is used.

Returns

A positive Sybase result set identifier on success.

TRUE – query was successful, but no result set was returned.

FALSE – query failed.

sybase_rpc_bind_param_ex

Binds a PHP variable to a remote procedure parameter.

Syntax

```
bool sybase_rpc_bind_param_ex(resource $stmt,
int $param_id,
mixed $var,
string $type
[, bool $is_null
[, int $direction]] )
```

Parameters

- **$stmt –** a statement identifier resource returned by a **sybase_rpc_init()** call.
- **$param_id –** positional index of the stored procedure parameter to bind with. It starts with 0 for the first parameter.
- **$var –** reference to (address of) the PHP variable to be bound.
- **$type –** the data type of the PHP variable that will be bound. One of:

  'd' - double

  'i' - integer

  'b' - binary

  's' - string

- **$is_null –** an optional boolean indicating whether the variable contains a NULL or not.
- **$direction –** optionally, one of:

  SASE_D_INPUT – for an input parameter (default).

  SASE_D_OUTPUT – for an output parameter.

Returns

TRUE – binding the PHP variable succeeded.

FALSE – binding the PHP variable failed

sybase_rpc_execute

Executes the remote procedure call that was initialized with **sybase_rpc_init()** in $stmt.

Syntax

```
mixed sybase_rpc_execute(resource $stmt)
```

Parameters

- **$stmt –** a statement identifier resource returned by a **sybase_rpc_init()** call.

Returns

A positive Sybase result set identifier on success.

FALSE – the RPC execution failed.

sybase_rpc_init

Returns a statement identifier pointing to the statement initialized for $procedure on connection $conn.

Syntax

```
mixed sybase_rpc_init(resource $conn, string $procedure)
```

Parameters

- **$conn –** the connection resource returned by a connection opening function.
- **$procedure –** the name of the remote (stored) procedure to be executed with **sybase_rpc_execute()**.

Returns

A positive Sybase statement identifier on success.

FALSE – initialization of the RPC statement failed.

sybase_select_db

Sets the current active database on the server referred to by the connection resource.

Every subsequent call to **sybase_query()** is made on the active database of the current connection resource.

Syntax

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

Parameters

- **$database_name –** the name of the database to use.
- **$conn –** the connection resource returned by a connection opening function.
  If the connection resource is not specified, the most recently opened connection is used.

Returns

TRUE – current database set successfully.

FALSE – failed to set current database.

sybase_set_message_handler

Sets a user defined callback function that is called when either a client or a server message is received.

Syntax

```
bool sybase_set_message_handler(callback $handler, int $msg_type
[, resource $conn])
```

Parameters

- **$handler –** the callback handler takes the following arguments:

  int – message_number

  int – severity

  int – state

  int – line_number

  string – description
- **$msg_type –** one of:

  SYB_CLIENTMSG_CB – a client message callback

  SYB_SERVERMSG_CB – a server message callback

Note: Although **$msg_type** is mandatory, currently it is ignored as the installed message handler is called for both client and server messages.

- **$conn –** the connection resource returned by a connection opening function.
  If the connection resource is not specified, the most recently opened connection is used.

Returns

TRUE – callback function installed successfully.

FALSE – failed to install callback function.

sybase_unbuffered_query

Sends a query to the connection referred to by $conn.

The complete result set is not automatically fetched and buffered as with **sybase_query()**. This yields better performance, especially with large result sets.

Use **sybase_fetch_array()** and similar functions to read more rows as needed and use **sybase_data_seek()** to jump to the target row.

Use **sybase_num_rows()** to return the correct number of rows when the complete result set has been read.

Syntax

```
mixed sybase_unbuffered_query(string $query [, resource $conn])
```

Parameters

- **$query –** a string containing the query to be sent to Adaptive Server.
- **$conn –** the connection resource returned by a connection opening function.

  If the connection resource is not specified, the most recently opened connection is used.

Returns

A positive Sybase result set identifier on success.

TRUE – query has been successful but no result set was returned.

FALSE – query failed.

sybase_use_result

Stores the result set of the last unbuffered query on connection $conn and returns a result set identifier pointing to this stored result set.

Syntax

```
mixed sybase_use_result(resource $conn)
```

Parameters

- **$conn –** the connection resource returned by a connection opening function.

Returns

A positive Sybase result set identifier on success.

FALSE – there is no further result set on the connection to store.

Security and Directory Services

Configure security options using the `ocs.cfg` and `libtcl.cfg` files.

The extension module options enabling security are currently not supported.

For a connection, use `ocs.cfg` to set directory and security properties. Edit `libtcl.cfg` to load security and directory service drivers.

For more information, see *Configuration Files* in the *Open Client and Open Server Configuration Guide for UNIX.*

Additional Resources

Additional information for installing and configuring the extension module.

- Open Client and Open Server documentation for configuration information:
  Open Client and Open Server Configuration Guide for UNIX > Configuration Files
- Platform related issues for all the Open Client and Open Server products:
  Open Client and Open Server Programmers Supplement for UNIX
- Using the Open Client and Open Server runtime configuration file:
  Open Client Client-Library/C Reference Manual > Using the runtime configuration file > Open Client and Open Server runtime configuration file syntax
- Platform support:
  Software Developer Kit and Open Server Installation Guide for your platform.

Glossary

Glossary of term specific to scripting languages.

- **Client-Library –** part of Open Client, a collection of routines for use in writing client applications. Client-Library is designed to accommodate cursors and other advanced features in the Sybase product line.
- **CS-Library –** included with both the Open Client and Open Server products, a collection of utility routines that are useful to both Client-Library and Server-Library applications.
- **CT-Library –** (CT-Lib API) is part of the Open Client suite and is required to let an scripting application connect to Adaptive Server.
- **PHP –** self-referential acronym for Hypertext Preprocessor.
- **thread –** a path of execution through Open Server application and library code and the path's associated stack space, state information, and event handlers.
- **Transact-SQL –** an enhanced version of the database language SQL. Applications can use Transact-SQL to communicate with Adaptive Server Enterprise.

Index