Database (dbm-style) Abstraction Layer

Introduction

These functions build the foundation for accessing Berkeley DB style databases.

This is a general abstraction layer for several file-based databases. As such, functionality is limited to a common subset of features supported by modern databases such as » Sleepycat Software's DB2. (This is not to be confused with IBM's DB2 software, which is supported through the ODBC functions.)

Installing/Configuring

Requirements

The behaviour of various aspects depends on the implementation of the underlying database. Functions such as dba_optimize() and dba_sync() will do what they promise for one database and will do nothing for others. You have to download and install supported dba-Handlers.

List of DBA handlers

Handler	Notes
dbm	Dbm is the oldest (original) type of Berkeley DB style databases. You should avoid it, if possible. We do not support the compatibility functions built into DB2 and gdbm, because they are only compatible on the source code level, but cannot handle the original dbm format.
ndbm	Ndbm is a newer type and more flexible than dbm. It still has most of the arbitrary limits of dbm (therefore it is deprecated).
gdbm	Gdbm is the » GNU database manager.
db2	DB2 is » Sleepycat Software's DB2. It is described as "a programmatic toolkit that provides high-performance built-in database support for both standalone and client/server applications.
db3	DB3 is » Sleepycat Software's DB3.
db4	DB4 is » Sleepycat Software's DB4. This is available since PHP 4.3.2.
cdb	Cdb is "a fast, reliable, lightweight package for creating and reading constant databases." It is from the author of qmail and can be found at » http://cr.yp.to/cdb.html. Since it is constant, we support only reading operations. And since PHP 4.3.0 we support writing (not updating) through the internal cdb library.

cdb_make	Since PHP 4.3.0 we support creation (not updating) of cdb files when the bundled cdb library is used.
flatfile	This is available since PHP 4.3.0 for compatibility with the deprecated *dbm* extension only and should be avoided. However you may use this where files were created in this format. That happens when configure could not find any external library.
inifile	This is available since PHP 4.3.3 to be able to modify php.ini files from within PHP scripts. When working with ini files you can pass arrays of the form array(0=>group,1=>value_name) or strings of the form "[group]value_name" where group is optional. As the functions dba_firstkey() and dba_nextkey() return string representations of the key there is a new function dba_key_split() available since PHP 5 which allows to convert the string keys into array keys without loosing **FALSE**.
qdbm	This is available since PHP 5.0.0. The qdbm library can be loaded from » http://qdbm.sourceforge.net.

When invoking the dba_open() or dba_popen() functions, one of the handler names must be supplied as an argument. The actually available list of handlers is displayed by invoking phpinfo() or dba_handlers().

Installation

By using the *--enable-dba=shared* configuration option you can build a dynamic loadable module to enable PHP for basic support of dbm-style databases. You also have to add support for at least one of the following handlers by specifying the *--with-XXXX* configure switch to your PHP configure line.

Warning
After configuring and compiling PHP you must execute the following test from commandline: *php run-tests.php ext/dba*. This shows whether your combination of handlers works. Most problematic are *dbm* and *ndbm* which conflict with many installations. The reason for this is that on several systems these libraries are part of more than one other library. The configuration test only prevents you from configuring malfunctioning single handlers but not combinations.

Supported DBA handlers

Handler	Configure Switch
dbm	To enable support for dbm add *--with-dbm[=DIR]*. **Note** dbm normally is a wrapper which often results in failures. This means you should only use dbm if you are sure it works and if you really need this format.
ndbm	To enable support for ndbm add *--with-ndbm[=DIR]*. **Note** ndbm normally is a wrapper which often results in failures. This means you should only use ndbm if you are sure it works and if you really need this format.
gdbm	To enable support for gdbm add *--with-gdbm[=DIR]*.
db2	To enable support for db2 add *--with-db2[=DIR]*. **Note** db2 conflicts with db3 and db4.
db3	To enable support for db3 add *--with-db3[=DIR]*.

Note
db3 conflicts with db2 and db4.

db4	To enable support for db4 add *--with-db4[=DIR]*.

Note
db4 conflicts with db2 and db3.

Note
This was added in PHP 4.3.2. In earlier versions of PHP you need to use *--with-db3=DIR* with DIR being the path to db4 library. It is not possible to use db versions starting from 4.1 with PHP prior to version 4.3.0. Also, the db libraries with versions 4.1 through 4.1.24 cannot be used in any PHP version.

cdb	To enable support for cdb add

	--with-cdb[=DIR]. **Note** Since PHP 4.3.0 you can omit DIR to use the bundled cdb library that adds the cdb_make handler which allows creation of cdb files and allows to access cdb files on the network using PHP's streams.
flatfile	To enable support for flatfile add *--with-flatfile*. **Note** This was added in PHP 4.3.0 to add compatibility with deprecated *dbm* extension. Use this handler only when you cannot install one of the libraries required by the other handlers and when you cannot use bundled cdb handler.
inifile	To enable support for inifile add *--with-inifile*. **Note** This was added in PHP 5.0.0 and allows to read and set microsoft style*.ini* files (like the *php.ini* file).
qdbm	To enable support for qdbm add *--with-qdbm[=DIR]*. **Note** qdbm conflicts with dbm and gdbm.

Note

This was added in PHP 5.0.0. The qdbm library can be loaded from » http://qdbm.sourceforge.net.

Note

Up to PHP 4.3.0 you are able to add both db2 and db3 handler but only one of them can be used internally. That means that you cannot have both file formats. Starting with PHP 5.0.0 there is a configuration check avoid such misconfigurations.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

The functions dba_open() and dba_popen() return a handle to the specified database file to access which is used by all other dba-function calls.

Predefined Constants

This extension has no constants defined.

Examples

Basic usage

Example #1 - DBA example

```
<?php

$id = dba_open("/tmp/test.db", "n", "db2");

if (!$id) {
   echo "dba_open failed\n";
   exit;
}

dba_replace("key", "This is an example!", $id);

if (dba_exists("key", $id)) {
   echo dba_fetch("key", $id);
   dba_delete("key", $id);
}

dba_close($id);
?>
```

DBA is binary safe and does not have any arbitrary limits. However, it inherits all limits set by the underlying database implementation.

All file-based databases must provide a way of setting the file mode of a new created database, if that is possible at all. The file mode is commonly passed as the fourth argument to dba_open() or dba_popen().

You can access all entries of a database in a linear way by using the dba_firstkey() and dba_nextkey() functions. You may not change the database while traversing it.

Example #2 - Traversing a database

```
<?php

// ...open database...

$key = dba_firstkey($id);

while ($key != false) {
   if (true) {          // remember the key to perform some action later
       $handle_later[] = $key;
```

```
   }
   $key = dba_nextkey($id);
}

foreach ($handle_later as $val) {
   dba_delete($val, $id);
}

?>
```

DBA Functions

dba_close

dba_close -- Close a DBA database

Description

void **dba_close** (resource `$handle`)

dba_close() closes the established database and frees all resources of the specified database handle.

Parameters

`handle`
The database handler, returned by dba_open() or dba_popen().

Return Values

No value is returned.

See Also

- dba_open()
- dba_popen()

dba_delete

dba_delete -- Delete DBA entry specified by key

Description

bool **dba_delete** (string `$key`, resource `$handle`)

dba_delete() deletes the specified entry from the database.

Parameters

`key`
The key of the entry which is deleted.

`handle`
The database handler, returned by dba_open() or dba_popen().

Return Values

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

See Also

- dba_exists()
- dba_fetch()
- dba_insert()
- dba_replace()

dba_exists

dba_exists -- Check whether key exists

Description

bool **dba_exists** (string `$key`, resource `$handle`)

dba_exists() checks whether the specified *`key`* exists in the database.

Parameters

`key`
: The key the check is performed for.

`handle`
: The database handler, returned by dba_open() or dba_popen().

Return Values

Returns **TRUE** if the key exists, **FALSE** otherwise.

See Also

- dba_delete()
- dba_fetch()
- dba_insert()
- dba_replace()

dba_fetch

dba_fetch -- Fetch data specified by key

Description

string **dba_fetch** (string `$key`, resource `$handle`)

string **dba_fetch** (string `$key`, int `$skip`, resource `$handle`)

dba_fetch() fetches the data specified by *`key`* from the database specified with *`handle`*.

Parameters

`key`
The key the data is specified by.

Note
When working with inifiles this function accepts arrays as keys where index 0 is the group and index 1 is the value name. See: dba_key_split().

`skip`
The number of key-value pairs to ignore when using cdb databases. This value is ignored for all other databases which do not support multiple keys with the same name.

`handle`
The database handler, returned by dba_open() or dba_popen().

Return Values

Returns the associated string if the key/data pair is found, **FALSE** otherwise.

ChangeLog

Version	Description
4.3	The *`skip`* parameter is available to support cdb's capability of multiple keys having the same name.

See Also

- dba_exists()
- dba_delete()
- dba_insert()
- dba_replace()
- dba_key_split()

dba_firstkey

dba_firstkey -- Fetch first key

Description

string **dba_firstkey** (resource `$handle`)

dba_firstkey() returns the first key of the database and resets the internal key pointer. This permits a linear search through the whole database.

Parameters

`handle`
The database handler, returned by dba_open() or dba_popen().

Return Values

Returns the key on success, or **FALSE** on failure.

See Also

- dba_nextkey()
- dba_key_split()
- Example 2 in the DBA examples

dba_handlers

dba_handlers -- List all the handlers available

Description

array **dba_handlers** ([bool `$full_info`])

dba_handlers() list all the handlers supported by this extension.

Parameters

`full_info`
Turns on/off full information display in the result. The default is **FALSE**.

Return Values

Returns an array of database handlers. If *`full_info`* is set to **TRUE**, the array will be associative with the handlers names as keys, and their version information as value. Otherwise, the result will be an indexed array of handlers names.

Note
When the internal cdb library is used you will see *cdb* and *cdb_make*.

Examples

Example #3 - dba_handlers() Example

```
<?php

echo "Available DBA handlers:\n";
foreach (dba_handlers(true) as $handler_name => $handler_version) {
 // clean the versions
 $handler_version = str_replace('$', '', $handler_version);
 echo " - $handler_name: $handler_version\n";
}

?>
```

The above example will output something similar to:

```
Available DBA handlers:
- cdb: 0.75, Revision: 1.3.2.3
```

```
- cdb_make: 0.75, Revision: 1.2.2.4
- db2: Sleepycat Software: Berkeley DB 2.7.7: (08/20/99)
- inifile: 1.0, Revision: 1.6.2.3
- flatfile: 1.0, Revision: 1.5.2.4
```

dba_insert

dba_insert -- Insert entry

Description

bool **dba_insert** (string `$key`, string `$value`, resource `$handle`)

dba_insert() inserts the entry described with *`key`* and *`value`* into the database.

Parameters

`key`
: The key of the entry to be inserted. If this key already exist in the database, this function will fail. Use dba_replace() if you need to replace an existent key.

`value`
: The value to be inserted.

`handle`
: The database handler, returned by dba_open() or dba_popen().

Return Values

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

See Also

- dba_exists()
- dba_delete()
- dba_fetch()
- dba_replace()

dba_key_split

dba_key_split -- Splits a key in string representation into array representation

Description

mixed **dba_key_split** (mixed `$key`)

dba_key_split() splits a key (string representation) into an array representation.

Parameters

key
The key in string representation.

Return Values

Returns an array of the form *array(0 => group, 1 => value_name)*. This function will return **FALSE** if *key* is **NULL** or **FALSE**.

See Also

- dba_firstkey()
- dba_nextkey()
- dba_fetch()

dba_list

dba_list -- List all open database files

Description

array **dba_list** (void)

dba_list() list all open database files.

Return Values

An associative array, in the form *resourceid* => *filename*.

dba_nextkey

dba_nextkey -- Fetch next key

Description

string **dba_nextkey** (resource `$handle`)

dba_nextkey() returns the next key of the database and advances the internal key pointer.

Parameters

`handle`
The database handler, returned by dba_open() or dba_popen().

Return Values

Returns the key on success, or **FALSE** on failure.

See Also

- dba_firstkey()
- dba_key_split()
- Example 2 in the DBA examples

dba_open

dba_open -- Open database

Description

resource **dba_open** (string `$path`, string `$mode` [, string `$handler` [, mixed `$...`]])

dba_open() establishes a database instance for *`path`* with *`mode`* using *`handler`*.

Parameters

`path`
: Commonly a regular path in your filesystem.

`mode`
: It is *r* for read access, *w* for read/write access to an already existing database, *c* for read/write access and database creation if it doesn't currently exist, and *n* for create, truncate and read/write access. Additionally you can set the database lock method with the next char. Use *l* to lock the database with a *.lck* file or *d* to lock the databasefile itself. It is important that all of your applications do this consistently. If you want to test the access and do not want to wait for the lock you can add *t* as third character. When you are absolutely sure that you do not require database locking you can do so by using - instead of *l* or *d*. When none of *d*, *l* or - is used, dba will lock on the database file as it would with *d*.

Note

There can only be one writer for one database file. When you use dba on a web server and more than one request requires write operations they can only be done one after another. Also read during write is not allowed. The dba extension uses locks to prevent this. See the following table:

DBA locking

already open	*mode* = **"rl"**	*mode* = **"rlt"**	*mode* = **"wl"**	*mode* = **"wlt"**	*mode* = **"rd"**	*mode* = **"rdt"**	*mode* = **"wd"**	*mode* = **"wdt"**
not open	ok	ok	ok	ok	ok	ok	ok	ok
mode = "rl"	ok	ok	wait	false	illegal	illegal	illegal	illegal
mode = "wl"	wait	false	wait	false	illegal	illegal	illegal	illegal
mode = "rd"	illegal	illegal	illegal	illegal	ok	ok	wait	false

mode = "wd"	illegal	illegal	illegal	illegal	wait	false	wait	false

- ok: the second call will be successfull.
- wait: the second call waits until dba_close() is called for the first.
- false: the second call returns false.
- illegal: you must not mix "l" and "d" modifiers for *mode* parameter.

handler
The name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to dba_open() and can act on behalf of them.

Return Values

Returns a positive handle on success, or **FALSE** on failure.

ChangeLog

Version	Description
4.3.0	It's possible to open database files over network connection. However in cases a socket connection will be used (as with http or ftp) the connection will be locked instead of the resource itself. This is important to know since in such cases locking is simply ignored on the resource and other solutions have to be found.
4.3.0	Locking and the *mode* modifiers "l", "d", "-" and "t" were added. In previous PHP versions, you must use semaphores to guard against simultaneous database access for any database handler with the exception of GDBM. See System V semaphore support.
before 4.3.5	open mode 'c' is broken for several internal handlers and truncates the database instead of appending data to an existent database. Also dbm and ndbm fail on mode 'c' in typical configurations (this cannot be fixed).

See Also

- dba_popen()
- dba_close()

dba_optimize

dba_optimize -- Optimize database

Description

bool **dba_optimize** (resource `$handle`)

dba_optimize() optimizes the underlying database.

Parameters

`handle`
The database handler, returned by dba_open() or dba_popen().

Return Values

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

See Also

- dba_sync()

dba_popen

dba_popen -- Open database persistently

Description

resource **dba_popen** (string `$path`, string `$mode` [, string `$handler` [, mixed `$...`]])

dba_popen() establishes a persistent database instance for *`path`* with *`mode`* using *`handler`*
.

Parameters

`path`
: Commonly a regular path in your filesystem.

`mode`
: It is *r* for read access, *w* for read/write access to an already existing database, *c* for read/write access and database creation if it doesn't currently exist, and *n* for create, truncate and read/write access.

`handler`
: The name of the handler which shall be used for accessing *`path`*. It is passed all optional parameters given to dba_popen() and can act on behalf of them.

Return Values

Returns a positive handle on success, or **FALSE** on failure.

See Also

- dba_open()
- dba_close()

dba_replace

dba_replace -- Replace or insert entry

Description

bool **dba_replace** (string `$key`, string `$value`, resource `$handle`)

dba_replace() replaces or inserts the entry described with *`key`* and *`value`* into the database specified by *`handle`*.

Parameters

`key`
: The key of the entry to be replaced.

`value`
: The value to be replaced.

`handle`
: The database handler, returned by dba_open() or dba_popen().

Return Values

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

See Also

- dba_exists()
- dba_delete()
- dba_fetch()
- dba_insert()

dba_sync

dba_sync -- Synchronize database

Description

bool **dba_sync** (resource `$handle`)

dba_sync() synchronizes the database. This will probably trigger a physical write to the disk, if supported.

Parameters

`handle`
The database handler, returned by dba_open() or dba_popen().

Return Values

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

See Also

- dba_optimize()