Paradox File Access

Introduction

This module allows to read and write Paradox databases, primary index files and blob files. Write support has been proven to be quite reliable, though due to lack of documentation the produced files may not in any case be readable by other applications. Encrypted databases can be read without specifying a password if pxlib >= 0.5.0 is used.

Note
This module is also in development and may change, though I don't expect major changes to the API.

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension?including the names of its functions and any other documentation surrounding this extension?may change without notice in a future release of PHP. This extension should be used at your own risk.

Installing/Configuring

Requirements

You need at least PHP 5.0.0 and pxlib >= 0.4.4 for the basic set of functions. Some newer functions are only available with pxlib >= 0.6.0. Reading and writing of encrypted databases requires at least pxlib >= 0.5.0. The paradox library (pxlib) is available at » http://pxlib.sourceforge.net.

Installation

Information for installing this PECL extension may be found in the manual chapter titled Installation of PECL extensions. Additional information such as new releases, downloads, source files, maintainer information, and a CHANGELOG, can be located here: » http://pecl.php.net/package/paradox

Make sure you have pxlib installed before. If you install pxlib from an rpm or debian package, do not forget to install the development package as well.

Runtime Configuration

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

Resource Types

px_new() creates a new Paradox object required by all Paradox functions.

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 two tables lists all constants defined by the paradox extension.

Contants for field types

Name	Meaning
PX_FIELD_ALPHA	Character data with fixed length
PX_FIELD_DATE	Date, number of days since 1.1.0000
PX_FIELD_SHORT	Short integer (2 Bytes)
PX_FIELD_LONG	Long integer (4 Bytes)
PX_FIELD_CURRENCY	same as PX_FIELD_NUMBER
PX_FIELD_NUMBER	Double
PX_FIELD_LOGICAL	Boolean
PX_FIELD_MEMOBLOB	Binary large object
PX_FIELD_BLOB	Binary large object (not supported)
PX_FIELD_FMTMEMOBLOB	Binary large object
PX_FIELD_OLE	OLE object (basically a blob, not supported)
PX_FIELD_GRAPHIC	Graphic (basically a blob, not supported)
PX_FIELD_TIME	time, number of milli seconds since midnight
PX_FIELD_TIMESTAMP	timestamp, number of milli seconds since 1.1.0000
PX_FIELD_AUTOINC	Auto incrementing interger (like PX_FIELD_LONG)
PX_FIELD_BCD	Decimal number stored in bcd format (not supported)
PX_FIELD_BYTES	Array of Bytes with not more than 255 bytes (not supported)
PX_KEYTOLOWER	Turn all field names into lower case

PX_KEYTOUPPER	Turn all field names into upper case

Contants for file types

Name	**Meaning**
PX_FILE_INDEX_DB	Indexed database
PX_FILE_PRIM_INDEX	Primary index
PX_FILE_NON_INDEX_DB	None indexed database
PX_FILE_NON_INC_SEC_INDEX	None incremental secondary index
PX_FILE_SEC_INDEX	Secondary index
PX_FILE_INC_SEC_INDEX	Incremental secondary index
PX_FILE_NON_INC_SEC_INDEX_G	Non incremental secondary index
PX_FILE_SEC_INDEX_G	Secondary index
PX_FILE_INC_SEC_INDEX_G	Non incremental secondary index

Paradox Functions

Object oriented API

The paradox extension provides also an object oriented API. It consists of only one class called paradox_db. Its methods only differ from the functions in its name and of course the missing first parameter. The following table will list all methods and its equivalent functions.

Methods of class paradox_db

Name of method	Equivalent function
Constructor	px_new()
Destructor	px_delete()
open_fp()	px_open_fp()
create_fp()	px_create_fp()
close()	px_close()
numrecords()	px_numrecords()
numfields()	px_numfields()
get_record()	px_get_record()
put_record()	px_put_record()
retrieve_record()	px_retrieve_record()
delete_record()	px_delete_record()
insert_record()	px_insert_record()
update_record()	px_update_record()
get_field()	px_get_field()
get_schema()	px_get_schema()
get_info()	px_get_info()
set_parameter()	px_set_parameter()

get_parameter()	px_get_parameter()
set_value()	px_set_value()
get_value()	px_get_value()
get_info()	px_get_info()
set_targetencoding()	px_set_targetencoding()
set_tablename()	px_set_tablename()
set_blob_file()	px_set_blob_file()
date2string()	px_date2string()
timestamp2string()	px_timestamp2string()

px_close

px_close -- Closes a paradox database

Description

bool **px_close** (resource `$pxdoc`)

Closes the paradox database. This function will not close the file. You will have to call fclose() afterwards.

Parameters

pxdoc
Resource identifier of the paradox database as returned by px_new().

Return Values

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

See Also

- px_open_fp()
- The example at px_new()

px_create_fp

px_create_fp -- Create a new paradox database

Description

bool **px_create_fp** (resource `$pxdoc`, resource `$file`, array `$fielddesc`)

Create a new paradox database file. The actual file has to be opened before with fopen(). Make sure the file is writable.

Note
Calling this functions issues a warning about an empty tablename which can be safely ignored. Just set the tablename afterwards with px_set_parameter().

Note
This function is highly experimental, due to insufficient documentation of the paradox file format. Database files created with this function can be opened by px_open_fp() and has been successfully opened by the Paradox software, but your milage may vary.

Parameters

`pxdoc`
: Resource identifier of the paradox database as returned by px_new().

`file`
: File handle as returned by fopen().

`fielddesc`
: *fielddesc* is an array containing one element for each field specification. A field specification is an array itself with either two or three elements.The first element is always a string value used as the name of the field. It may not be larger than ten characters. The second element contains the field type which is one of the constants listed in the table Constants for field types. In the case of a character field or bcd field, you will have to provide a third element specifying the length respectively the precesion of the field. If your field specification contains blob fields, you will have to make sure to either make the field large enough for all field values to fit or specify a blob file with px_set_blob_file() for storing the blobs. If this is not done the field data is truncated.

Return Values

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

Examples

Example #1 - Creating a Paradox database with two fields

```
<?php
if(!$pxdoc = px_new()) {
 /* Error handling */
}
$fp = fopen("test.db", "w+");
$fields = array(array("col1", "S"), array("col2", "I"));
if(!px_create_fp($pxdoc, $fp, $fields)) {
 /* Error handling */
}
px_set_parameter($pxdoc, "tablename", "testtable");
for($i=-50; $i<50; $i++) {
 $rec = array($i, -$i);
 px_put_record($pxdoc, $rec);
}
px_close($pxdoc);
px_delete($pxdoc);
fclose($fp);
?>
```

See Also

- px_new()
- px_put_record()
- fopen()

px_date2string

px_date2string -- Converts a date into a string.

Description

string **px_date2string** (resource `$pxdoc`, int `$value`, string `$format`)

Turns a date as it stored in the paradox file into human readable format. Paradox dates are the number of days since 1.1.0000. This function is just for convenience. It can be easily replaced by some math and the calendar functions as demonstrated in the example below.

Parameters

pxdoc
: Resource identifier of the paradox database as returned by px_new().

value
: Value as stored in paradox database field of type PX_FIELD_DATE.

format
: String format similar to the format used by date(). The placeholders support by this function is a subset of those supported by date() (Y, y, m, n, d, j, L).

Return Values

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

Examples

Example #2 - Turn a paradox date into a human readable form

```
<?php
$px = px_new();

/* make up a date as it could be stored in */
/* a date field of a paradox db. */
/* 700000 days since 1.1.0000. */
$days = 700000;

/* Use the calendar functions to print a */
/* human readable format of the date */
echo jdtogregorian($days+1721425)."\n";
/* px_date2string() outputs the same */
echo px_date2string($px, $days, "n/d/Y")."\n";
```

```
px_delete($px);
?>
```

The above example will output:

```
7/15/1917
7/15/1917
```

See Also

- px_timestamp2string()
- jdtogregorian()

px_delete_record

px_delete_record -- Deletes record from paradox database

Description

bool **px_delete_record** (resource `$pxdoc`, int `$num`)

This function deletes a record from the database. It does not free the space in the database file but just marks it as deleted. Inserting a new record afterwards will reuse the space.

Note
This function is only available if pxlib >= 0.6.0 is used.

Parameters

`pxdoc`
Resource identifier of the paradox database as returned by px_new().

`num`
The record number is an artificial number counting records in the order as they are stored in the database. The first record has number 0.

px_delete

px_delete -- Deletes resource of paradox database

Description

bool **px_delete** (resource `$pxdoc`)

Deletes the resource of the paradox file and frees all memory.

Parameters

pxdoc
: Resource identifier of the paradox database as returned by px_new().

Return Values

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

px_get_field

px_get_field -- Returns the specification of a single field

Description

array **px_get_field** (resource `$pxdoc`, int `$fieldno`)

Parameters

`pxdoc`
Resource identifier of the paradox database as returned by px_new().

`fieldno`
Number of the field. The first field has number 0. Specifying a field number less than 0 and greater or equal the number of fields will trigger an error.

Return Values

Returns the specification of the *fieldno* 'th database field as an associated array. The array contains three fields named *name*, *type*, and *size*.

px_get_info

px_get_info -- Return lots of information about a paradox file

Description

array **px_get_info** (resource `$pxdoc`)

Parameters

pxdoc
: Resource identifier of the paradox database as returned by px_new().

Return Values

Returns an associated array with lots of information about a paradox file. This array is likely to be extended in the future.

fileversion
: Version of file multiplied by 10, e.g. 70.

tablename
: Name of table as stored in the file. If the database was created by pxlib, then this will be the name of the file without the extension.

numrecords
: Number of records in this table.

numfields
: Number of fields in this table.

headersize
: Number of bytes used for the header. This is usually 0x800.

recordsize
: Number of bytes used for each record. This is the sum of all field sizes (available since version 1.4.2).

maxtablesize
: This value multiplied by 0x400 is the size of a data block in bytes. The maximum number of records in a datablock is the integer part of (maxtablesize * 0x400 - 8) / recordsize.

numdatablocks
: The number of data blocks in the file. Each data block contains a certain number of records which depends on the record size and the data block size (maxtablesize). Data blocks may not necessarily be completely filled.

numindexfields
: Number of fields used for the primary index. The fields do always start with field number 1.

codepage
: The DOS codepage which was used for encoding fields with character data. If the target encoding is not set with px_set_targetencoding() this will be the encoding for character fields when records are being accessed with px_get_record() or px_retrieve_record().

See Also

- px_numfields()
- px_numrecords()

px_get_parameter

px_get_parameter -- Gets a parameter

Description

string **px_get_parameter** (resource `$pxdoc`, string `$name`)

Gets various parameters.

Parameters

`pxdoc`
Resource identifier of the paradox database as returned by px_new().

`name`
The *`name`* can be one of the following:
tablename
The name of the table as it will be stored in the database header.

targetencoding
The encoding for the output. Data which is being read from character fields with px_get_record() or px_retrieve_record() is recoded into the targetencoding. If it is not set, then the data will be delivered as stored in the database file.

inputencoding
The encoding of the input data which is to be stored into the database. When storing data of character fields in the database, the data is expected to be delivered in this encoding.

Return Values

Returns the value of the parameter or **FALSE** on failure.

px_get_record

px_get_record -- Returns record of paradox database

Description

array **px_get_record** (resource `$pxdoc`, int `$num` [, int `$mode`])

Parameters

pxdoc
: Resource identifier of the paradox database as returned by px_new().

num
: The record number is an artificial number counting records in the order as they are stored in the database. The first record has number 0.

mode
: The optional *mode* can be **PX_KEYTOLOWER** or **PX_KEYTOUPPER** in order to convert the keys of the returned array into lower or upper case. If *mode* is not passed or is 0, then the key will be exactly like the field name. The element values will contain the field values. NULL values will be retained and are different from 0.0, 0 or the empty string. Fields of type **PX_FIELD_TIME** will be returned as an integer counting the number of milliseconds starting at midnight. A timestamp (**PX_FIELD_TIMESTAMP**) and date (**PX_FIELD_DATE**) are floating point respectively int values counting milliseconds respectively days starting at the beginning of julian calendar. Use the functions px-timestamp2string() and px-date2string() to convert them into a character representation.

Return Values

Returns the *num* 'th record from the paradox database. The record is returned as an associated array with its keys being the field names.

See Also

- px_retrieve_record()

px_get_schema

px_get_schema -- Returns the database schema

Description

array **px_get_schema** (resource `$pxdoc` [, int `$mode`])

px_get_schema() returns the database schema.

Parameters

pxdoc
: Resource identifier of the paradox database as returned by px_new().

mode
: If the optional *mode* is **PX_KEYTOLOWER** or **PX_KEYTOUPPER** the keys of the returned array will be converted to lower or upper case. If *mode* is 0 or not passed at all, then the key name will be identical to the field name.

Return Values

Returns the schema of a database file as an associated array. The key name is equal to the field name. Each array element is itself an associated array containing the two fields *type* and *size*. *type* is one of the constants in table Constants for field types. *size* is the number of bytes this field consumes in the record. The total of all field sizes is equal to the record size as it can be retrieved with px-get-info().

px_get_value

px_get_value -- Gets a value

Description

float **px_get_value** (resource `$pxdoc`, string `$name`)

Gets various values.

Parameters

`pxdoc`
Resource identifier of the paradox database as returned by px_new().

`name`
`name` can be one of the following.
numprimkeys
The number of primary keys. Paradox databases always use the first *numprimkeys* fields for the primary index.

Return Values

Returns the value of the parameter or **FALSE** on failure.

px_insert_record

px_insert_record -- Inserts record into paradox database

Description

int **px_insert_record** (resource `$pxdoc`, array `$data`)

Inserts a new record into the database. The record is not necessarily inserted at the end of the database, but may be inserted at any position depending on where the first free slot is found.

The record data is passed as an array of field values. The elements in the array must correspond to the fields in the database. If the array has less elements than fields in the database, the remaining fields will be set to null.

Most field values can be passed as its equivalent php type e.g. a long value is used for fields of type PX_FIELD_LONG, PX_FIELD_SHORT and PX_FIELD_AUTOINC, a double values is used for fields of type PX_FIELD_CURRENCY and PX_FIELD_NUMBER. Field values for blob and alpha fields are passed as strings.

Fields of type PX_FIELD_TIME and PX_FIELD_DATE both require a long value. In the first case this is the number of milliseconds since midnight. In the second case this is the number of days since 1.1.0000. Below there are two examples to convert the current date or timestamp into a value suitable for one of paradox's date/time fields.

Note
This function is only available if pxlib >= 0.6.0 is used.

Parameters

`pxdoc`
Resource identifier of the paradox database as returned by px_new().

`data`
Associated or indexed array containing the field values as e.g. returned by px_retrieve_record().

Return Values

Returns **FALSE** on failure or the record number in case of success.

Examples

Example #3 - Set the date/time fields in a paradox database to the current date/time

```
<?php
$px = px_new();
$fp = fopen("test.db", "w+");
px_create_fp($px, $fp, array(array("timestamp", "@"), array("time", "T"),
array("date", "D")));

$curdate = getdate();
$jd = gregoriantojd($curdate["mon"], $curdate["mday"], $curdate["year"]);
$days = $jd - 1721425; /* Number of days between 1.1.4714 b.c. and 1.1.0000
*/
$secs = $curdate["hours"]*3600 + $curdate["minutes"]*60 +
$curdate["seconds"];
px_insert_record($px, array($days*86400000.0 + $secs*1000.0, $secs*1000.0,
$days));

$curtimestamp = microtime(true);
$days = (int) ($curtimestamp/86400);
$secs = $curtimestamp - ($days * 86400.0);
$days += 2440588; /* Number of days between 1.1.4714 b.c. and 1.1.1970 */
$days -= 1721425; /* Number of days between 1.1.4714 b.c. and 1.1.0000 */
px_insert_record($px, array($days*86400000.0 + $secs*1000.0, $secs*1000.0,
$days));
for($i=0; $i<2; $i++) {
   $rec = px_retrieve_record($px, $i);
   echo px_timestamp2string($px, $rec["timestamp"], "n/d/Y H:i:s")."\n";
   echo px_date2string($px, $rec["date"], "n/d/Y")."\n";
}
px_close($px);
px_delete($px);
?>
```

The above example will output:

```
2/21/2006 21:42:30
2/21/2006
2/21/2006 20:42:30
2/21/2006
```

The Julian day count as passed to jdtogregorian() has a different base of 1.1.4714 b.c. and must therefore be calculated by adding 1721425 to the day count used in the paradox file. Turning the day count into a timestamp is easily done by multiplying with 86400000.0 to obtain milli seconds.

See Also

px_update_record()

px_new

px_new -- Create a new paradox object

Description

resource **px_new** (void)

Create a new paradox object. You will have to call this function before any further functions. px_new() does not create any file on the disk, it just creates an instance of a paradox object. This function must not be called if the object oriented interface is used. Use *new paradox_db()* instead.

Return Values

Returns **FALSE** on failure.

Examples

Example #4 - Opening a Paradox database

```
<?php
if(!$pxdoc = px_new()) {
 /* Error handling */
}
$fp = fopen("test.db", "r");
if(!px_open_fp($pxdoc, $fp)) {
 /* Error handling */
}
// ...
px_close($pxdoc);
px_delete($pxdoc);
fclose($fp);
?>
```

If you prefer the object oriented API, then the above example will look like the following.

Example #5 - Opening a Paradox database

```
<?php
$fp = fopen("test.db", "r");
$pxdoc = new paradox_db();
if(!$pxdoc->open_fp($fp)) {
 /* Error handling */
}
// ...
$pxdoc->close();
fclose($fp);
```

```
?>
```

See Also

- px_delete()
- px_open_fp()

px_numfields

px_numfields -- Returns number of fields in a database

Description

int **px_numfields** (resource `$pxdoc`)

Get the number of fields in a database file.

Parameters

pxdoc
: Resource identifier of the paradox database as returned by px_new().

Return Values

Returns the number of fields in a database file. The return value of this function is identical to the element *numfields* in the array returned by px_get_info().

px_numrecords

px_numrecords -- Returns number of records in a database

Description

int **px_numrecords** (resource `$pxdoc`)

Get the number of records in a database file.

Parameters

pxdoc
: Resource identifier of the paradox database as returned by px_new().

Return Values

Returns the number of records in a database file. The return value of this function is identical to the element *numrecords* in the array returned by px_get_info().

px_open_fp

px_open_fp -- Open paradox database

Description

bool **px_open_fp** (resource `$pxdoc`, resource `$file`)

Open an existing paradox database file. The actual file has to be opened before with fopen(). This function can also be used to open primary index files and tread them like a paradox database. This is supported for those who would like to investigate a primary index. It cannot be used to accelerate access to a database file.

Parameters

`pxdoc`
: Resource identifier of the paradox database as returned by px_new().

`file`
: *`file`* is the return value from fopen() with the actual database file as parameter. Make sure the database is writable if you plan to update or insert records.

Return Values

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

See Also

- fopen()
- The example at px_new()

px_put_record

px_put_record -- Stores record into paradox database

Description

bool **px_put_record** (resource `$pxdoc`, array `$record` [, int `$recpos`])

Stores a record into a paradox database. The record is always added at the end of the database, regardless of any free slots. Use px_insert_record() to add a new record into the first free slot found in the database.

Parameters

`pxdoc`
Resource identifier of the paradox database as returned by px_new().

`record`
Associated or indexed array containing the field values as e.g. returned by px_retrieve_record().

`recpos`
This optional parameter may be used to specify a record number greater than the current number of records in the database. The function will add as many empty records as needed. There is hardly any need for this parameter.

Return Values

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

px_retrieve_record

px_retrieve_record -- Returns record of paradox database

Description

array **px_retrieve_record** (resource `$pxdoc`, int `$num` [, int `$mode`])

This function is very similar to px_get_record() but uses internally a different approach to retrieve the data. It relies on pxlib for reading each single field value, which usually results in support for more field types.

Note
This function is only available if pxlib >= 0.6.0 is used.

Parameters

pxdoc
: Resource identifier of the paradox database as returned by px_new().

num
: The record number is an artificial number counting records in the order as they are stored in the database. The first record has number 0.

mode
: The optional *mode* can be **PX_KEYTOLOWER** or **PX_KEYTOUPPER** in order to convert the keys into lower or upper case. If *mode* is not passed or is 0, then the key will be exactly like the field name. The element values will contain the field values. NULL values will be retained and are different from 0.0, 0 or the empty string. Fields of type **PX_FIELD_TIME** will be returned as an integer counting the number of milliseconds starting at midnight. A timestamp is a floating point value also counting milliseconds starting at the beginning of julian calendar.

Return Values

Returns the *num* 'th record from the paradox database. The record is returned as an associated array with its keys being the field names.

See Also

- px_get_record()

px_set_blob_file

px_set_blob_file -- Sets the file where blobs are read from

Description

bool **px_set_blob_file** (resource `$pxdoc`, string `$filename`)

Sets the name of the file where blobs are going to be read from or written into. Without calling this function, px_get_record() or px_retrieve_record() will only return data in blob fields if the data is part of the record and not stored in the blob file. Blob data is stored in the record if it is small enough to fit in the size of the blob field.

Calling px_put_record(), px_insert_record(), or px_update_record() without calling px_set_blob_file() will result in truncated blob fields unless the data fits into the database file.

Calling this function twice will close the first blob file and open the new one.

Parameters

`pxdoc`
Resource identifier of the paradox database as returned by px_new().

`filename`
The name of the file. Its extension should be*.MB*.

Return Values

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

px_set_parameter

px_set_parameter -- Sets a parameter

Description

bool **px_set_parameter** (resource `$pxdoc`, string `$name`, string `$value`)

Sets various parameters.

Parameters

`pxdoc`
: Resource identifier of the paradox database as returned by px_new().

`name`
: Depending on the parameter you want to set, *`name`* can be one of the following.
  tablename
  : The name of the table as it will be stored in the database header.

  targetencoding
  : The encoding for the output. Data which is being read from character fields is recoded into the targetencoding.

  inputencoding
  : The encoding of the input data which is to be stored into the database.

`value`
: The value of parameter to set. For inputencoding and targetencoding this must be the name of the encoding as understood by iconv or recode, e.g. iso-8859-1, utf-8, cp850.

Return Values

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

See Also

- px_get_info() to determine the DOS code page.

px_set_tablename

px_set_tablename -- Sets the name of a table (deprecated)

Description

void **px_set_tablename** (resource `$pxdoc`, string `$name`)

Sets the table name of a paradox database, which was created with px_create_fp(). This function is deprecated use px_set_parameter() instead.

Parameters

`pxdoc`
: Resource identifier of the paradox database as returned by px_new().

`tablename`
: The name of the table. If it is not set explicitly it will be set to the name of the database file.

Return Values

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

See Also

px_set_parameter()

px_set_targetencoding

px_set_targetencoding -- Sets the encoding for character fields (deprecated)

Description

bool **px_set_targetencoding** (resource `$pxdoc`, string `$encoding`)

Set the encoding for data retrieved from a character field. All character fields will be recoded to the encoding set by this function. If the encoding is not set, the character data will be returned in the DOS code page encoding as specified in the database file. The *`encoding`* can be any string identifier known to iconv or recode. On Unix systems run *iconv -l* for a list of available encodings.

This function is deprecated and should be replaced by calling px_set_parameter().

See also px_get_info() to determine the DOS code page as stored in the database file.

Parameters

`pxdoc`
: Resource identifier of the paradox database as returned by px_new().

`encoding`
: The encoding for the output. Data which is being read from character fields is recoded into the targetencoding.

Return Values

Returns **FALSE** if the encoding could not be set, e.g. the encoding is unknown, or pxlib does not support recoding at all. In the second case a warning will be issued.

See Also

px_set_parameter()

px_set_value

px_set_value -- Sets a value

Description

bool **px_set_value** (resource `$pxdoc`, string `$name`, float `$value`)

Sets various values.

Parameters

`pxdoc`
: Resource identifier of the paradox database as returned by px_new().

`name`
: *`name`* can be one of the following.
: numprimkeys
: : The number of primary keys. Paradox databases always use the first *numprimkeys* fields for the primary index.

`value`

Return Values

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

See Also

px_set_parameter()

px_timestamp2string

px_timestamp2string -- Converts the timestamp into a string.

Description

string **px_timestamp2string** (resource `$pxdoc`, float `$value`, string `$format`)

Turns a timestamp as it stored in the paradox file into human readable format. Paradox timestamps are the number of milliseconds since 1.1.0000. This function is just for convenience. It can be easily replaced by some math and the calendar functions as demonstrated in the following example.

Parameters

`pxdoc`
Resource identifier of the paradox database.

`value`
Value as stored in paradox database field of type PX_FIELD_TIME, or PX_FIELD_TIMESTAMP.

`format`
String format similar to the format used by date(). The placeholders support by this function is a subset of those supported by date() (Y, y, m, n, d, j, H, h, G, g, i, s, A, a, L).

Return Values

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

Examples

Example #6 - Turn a paradox timestamp into a human readable form

```
<?php
$px = px_new();

/* make up a date as it could be stored in */
/* a date field of a paradox db. */
/* 700000 days since 1.1.0000. */
$days = 700000;

/* Use the calendar functions to print a */
/* human readable format of the date */
echo jdtogregorian($days+1721425)."\n";
```

```
/* Turn it into a timestamp as it stored in a paradox database */
/* Timestamps are stored in milli seconds since 1.1.0000 */
$stamp = $days * 86400.0 * 1000.0;
/* Add one hour */
$stamp += 3600000.0;
/* The following will output '7/15/1917 01:00:00'. */
echo px_timestamp2string($px, $stamp, "n/d/Y H:i:s")."\n";

px_delete($px);
?>
```

The above example will output:

```
7/15/1917
7/15/1917 01:00:00
```

The Julian day count as passed to jdtogregorian() has a different base of 1.1.4714 b.c. and must therefore be calculated by adding 1721425 to the day count used in the paradox file. Turning the day count into a timestamp is easily done by multiplying with 86400000.0 to obtain milli seconds.

See Also

- px_date2string()
- jdtogregorian()

px_update_record

px_update_record -- Updates record in paradox database

Description

bool **px_update_record** (resource `$pxdoc`, array `$data`, int `$num`)

Updates an exiting record in the database. The record starts at 0.

The record data is passed as an array of field values. The elements in the array must correspond to the fields in the database. If the array has less elements than fields in the database, the remaining fields will be set to null.

Note
This function is only available if pxlib >= 0.6.0 is used.

Parameters

`pxdoc`
Resource identifier of the paradox database as returned by px_new().

`data`
Associated array containing the field values as returned by px_retrieve_record().

`num`
The record number is an artificial number counting records in the order as they are stored in the database. The first record has number 0.

Return Values

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

See Also

px_insert_record()