ElevateDB Version 2 PHP Extension Manual

Table Of Contents

This page intentionally left blank

Chapter 1
Getting Started

1.1 Installation and Configuration

System Requirements

The ElevateDB PHP Extension works on any version of Windows, and requires PHP 5.2 or higher. The extension was tested using PHP 5.2 and higher for Windows and Linux with Apache HTTP Server 2.2 and 2.4.

Installing the Extension

Running the setup application provided with the ElevateDB PHP Extension will install all relevant files into the destination installation directory. However, you will still have to copy the proper extension DLL into the PHP extension directory, and modify the php.ini file to make sure that the extension is registered properly. The following steps will guide you through this process:

1. There are many different versions of the ElevateDB PHP Extension found in the subdirectories under the \libs subdirectory under the main ElevateDB PHP Extension installation directory. Each version's directory is specified as follows:

```
php-<PHP Version>-<Windows Compiler>
```

Note
Only certain PHP versions require a Windows compiler designation.

Under each of these directories are subdirectories for the target platform, which are currently either "win32" or "linux-i386". The PHP extension is currently only available as a 32-bit binary, but will at some point in the future be ported to a 64-bit architecture.

For example, to copy the ElevateDB PHP Extension for PHP 5.3 from the default installation directory under Windows to the PHP extension directory for use with the Apache web server, one would use the following command-line command:

```
copy "C:\Program Files\ElevateDB 2
      PHP-STD\libs\php-5.30-vc6\win32\php_edb.dll" "C:\php\ext"
```

2. The next step is to make sure that the ElevateDB PHP Extension is registered in the php.ini file. To do so, add the following lines to the end of the php.ini file:

```
[PHP_EDB]
extension=php_edb.dll
```

3. You can test to see if the extension was installed correctly by running the following script using a local instance of your web server:

```
<?php phpinfo(); ?>
```

This script should result a section of the PHP configuration entitled "edb" with a table that includes the following information:

ElevateDB 2 PHP Extension - Enabled

If this section and table is present, then the ElevateDB PHP Extension is installed properly and is ready to be used in your PHP scripts.

Installing the Extension with RadPHP

RadPHP from Embarcadero Technologies currently uses PHP 5.2, so please use the PHP 5.2 version of the ElevateDB PHP Extension. Also, when using the ElevateDB PHP Extension with RadPHP you will need to edit the php.ini.template file instead of the php.ini file in the \php subdirectory under the main installation directory for RadPHP. This is because the php.ini.template file is copied to a user-specific php.ini file every time the IDE is started.

1.2 Connection Strings

Connection strings are used when the edb_connect function is called, and specify all of the keywords necessary to configure and access a given ElevateDB configuration and database. The keywords that can be used with connection strings and the ElevateDB PHP Extension are listed below. Here is an example connection string that connects to a remote ElevateDB Server and a database called "Accounting" (case-insensitive):

```
CHARSET=UNICODE;
TYPE=REMOTE;
ADDRESS=192.168.0.28;
DATABASE=Accounting
```

Note
The line breaks inserted above are only for readability and should not be used in an actual connection string.

Connection String Keywords

The following keywords are used with connection strings. All keywords marked with the (R) symbol next to their name are only applicable when the TYPE keyword is set to "REMOTE". Likewise, all keywords marked with the (L) symbol next to their name are only applicable when the TYPE keyword is set to "LOCAL".

Name	Description
CHARSET	This string value specifies which character set, "ANSI" or "UNICODE", to use for the connection. For remote connections to an ElevateDB Server, the value must match the character set being used by the ElevateDB Server. The default value is "UNICODE" for Windows and "ANSI" for Linux.
TYPE	This string value is set to either "LOCAL" if the connection is accessing the database directly, or "REMOTE" if the connection is accessing the database remotely via an ElevateDB Server.
NAME	This string value specifies the name of the connection.
DESCRIPTION	This string value specifies the description of the connection.
HOST (R)	This string value specifies the host name of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "".
ADDRESS (R)	This string value specifies the IP address of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "127.0.0.1".

SERVICE (R)	This string value specifies the service name of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "".
PORT (R)	This string value specifies the port number of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "12010".
CONNECTTIMEOUT (R)	This string value specifies the maximum amount of time, in seconds, that ElevateDB will wait for a successful connection before aborting the connection attempt. The default value is "15" (seconds).
PING (R)	This string value specifies whether pinging will be enabled for the connection to the ElevateDB Server. When pinging is enabled, the driver will send a keep-alive request to the ElevateDB Server in the interval specified by the PINGINTERVAL value. This prevents the ElevateDB Server from disconnecting and/or removing the connection, even if the connection has been idle for a very long period of time. Please see the Server Session Timeout configuration item in the Starting and Configuring the ElevateDB Server topic for more information on how idle connections are handled. Specify "TRUE" to enable pinging, or "FALSE" (the default) to disable pinging.
PINGINTERVAL (R)	This string value specifies how often the connection will ping the ElevateDB Server when pinging is enabled via the PING value. The default value is "60" (seconds).
TIMEOUT (R)	This string value specifies how long the connection will wait on a response from the ElevateDB Server before disconnecting and issuing an error. The default value is "180" (seconds).
ENCRYPTED (R)	This string value specifies whether the connection to the ElevateDB Server should be encrypted or no. Specify "TRUE" to enable encryption for the connection, or "FALSE" (the default) to disable encryption for the connection.
ENCRYPTSRVPWD (R)	This string value specifies the password to use for encrypted connections to the remote ElevateDB Server, as well as for encrypting logins to the remote ElevateDB Server. This password must match the configured encryption password for the remote ElevateDB Server, and you should only specify this value if you know exactly what you are doing and need to use a different encryption password than the default of "elevatesoft".
COMPRESSION (R)	This string value specifies the amount of compression to use when communicating with the ElevateDB Server. The default value is "0", or no compression. A value of "1" to "10" specifies the amount of compression from fast, but not very thorough, to very thorough, but not as fast. A value of "6" specifies a balance between size and speed, and represents

	the ideal level for most applications. **Note** The ElevateDB PHP Extension will automatically adjust this value for situations where the existing compression level is not ideal, such as in cases where the amount of data being sent is so small that it is of no benefit to compress the data.
CONFIGMEMORY (L)	This string value specifies whether the configuration file used by the connection will be located in the process memory ("virtual") or on disk. Specify "TRUE" to use a virtual configuration file, or "FALSE" (the default) to use a disk-based configuration file in the location specified by the CONFIGPATH string value (see below). The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on configuration files.
CONFIGPATH (L)	This string value specifies the configuration path to use for the connection. The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on the configuration path.
TEMPPATH (L)	This string value specifies the temporary tables path to use for the connection. The temporary tables path is used to store any temporary tables generated during query execution. The default value is the operating system setting for the storing temporary files for the current user. Please see the Architecture for more information on the temporary tables path.
KEEPTABLESOPEN	This string value specifies whether tables should be kept open for the duration of the connection once they have been opened at least once. Specify "TRUE" to keep tables open, or "FALSE" (the default) to have tables opened and closed on demand. Setting this value to "TRUE" can result in improved performance, especially with applications that execute many singleton SQL statements in a row.
CONFIGNAME (L)	This string value specifies the root name (without extension) used by the connection for the configuration file. The default value is "EDBConfig". The extension used for the configuration file is determined by the CONFIGEXT value. The location of the configuration file is determined by the CONFIGPATH value.
CONFIGEXT (L)	This string value specifies the extension used by the connection for the configuration file. The default value is ".EDBCfg". The root name (without extension) used for the configuration file is determined by the CONFIGNAME value. The location of the configuration file is determined by the CONFIGPATH value.
LOCKEXT (L)	This string value specifies the extension used by the connection for both the configuration and database catalog

	lock files. The default value is ".EDBLck". The root name (without extension) used for the configuration lock file is determined by the CONFIGNAME value. The root name (without extension) used for a database catalog lock file is determined by the CATALOGNAME value. The location of the configuration lock file is determined by the CONFIGPATH value, and the configuration lock file is hidden, by default. The location of a database catalog lock file is determined by the path designated for the applicable database when the database was created, and a database catalog lock file is hidden, by default.
LOGEXT (L)	This string value specifies the extension used by the connection for the configuration log file. The default value is ".EDBLog". The root name (without extension) used for the configuration log file is determined by the CONFIGNAME value. The location of the configuration log file is determined by the CONFIGPATH value.
MAXLOGSIZE (L)	This string value specifies the maximum size of the log file (in bytes) that the log file can grow to. Log entries are added to the log in a circular fashion, meaning that once the maximum log file size ia reached, ElevateDB will start re-using the oldest log entries for new log entries. The default value is 1048576 bytes. Which types of logged events are recorded in the log can be controlled by the LOGCATS value. By default, all categories of events are logged (INFO, WARN, and ERROR). **Warning** It is very important that all data sources and/or applications accessing the same configuration file use the same maximum log file size for the configuration log file. Using different values can result in log entries being prematurely overwritten or appearing "out-of-order" when viewing the log entries via the LogEvents Table.
LOGCATS (L)	This string value specifies the types of events that should be logged in the configuration log file for the current connection, with each value separated by a comma (,). The default value is "INFO,WARN,ERROR", or all categories of events.
CATALOGNAME (L)	This string value specifies the root name (without extension) used by the connection for all database catalog files. The default value is "EDBDatabase". The extension used for the catalog files is determined by the CATALOGEXT value. The location of the catalog file is determined by the path designated for the applicable database when the database was created.
CATALOGEXT (L)	This string value specifies the extension used by the connection for database catalog files. The default value is ".EDBCat". The root name (without extension) used for all database catalog files is determined by the CATALOGNAME value. The location of the catalog file is determined by the path designated for the applicable database when the

	database was created.
BACKUPEXT (L)	This string value specifies the extension used by the connection for database backup files. The default value is ".EDBBkp". The root name (without extension) used for a database backup file is determined by the name given when the BACKUP DATABASE statement is executed.
UPDATEEXT (L)	This string value specifies the extension used by the connection for database update files. The default value is ".EDBUpd". The root name (without extension) used for a database update file is determined by the name given when the SAVE UPDATES statement is executed.
TABLEEXT (L)	This string value specifies the extension used by the connection for database table files. The default value is ".EDBTbl". The root name (without extension) used for a table file is determined by the name given when the CREATE TABLE statement is executed. The location of the table files is determined by the path designated for the applicable database when the database was created.
INDEXEXT (L)	This string value specifies the extension used by the connection for database table index files. The default value is ".EDBIdx". The root name (without extension) used for a table's index file is determined by the name given when the CREATE TABLE statement is executed. The location of the table index files is determined by the path designated for the applicable database when the database was created.
BLOBEXT (L)	This string value specifies the extension used by the connection for database table BLOB files. The default value is ".EDBBlb". The root name (without extension) used for a table's BLOB file is determined by the name given when the CREATE TABLE statement is executed. The location of the table BLOB files is determined by the path designated for the applicable database when the database was created.
PUBLISHEXT (L)	This string value specifies the extension used by the connection for database table publish files. The default value is ".EDBPbl". The root name (without extension) used for a table's publish file is determined by the name given when the CREATE TABLE statement is executed. The location of the table publish files is determined by the path designated for the applicable database when the database was created.
UID	This string value specifies the user ID to use when connecting to the connection. If this value is left blank, the user will be prompted for the user ID, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.
PWD	This string value specifies the password to use when connecting to the connection. If this value is left blank, the user will be prompted for the password, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.

DATABASE	This string value specifies the name of the database being accessed with the connection. Please see the Architecture for more information on databases.
READONLY	This string value specifies whether the connection is read-only or read-write. Set this value to "TRUE" to make the connection read-only, and "FALSE" to make the connection read-write.
ROWLOCKPROTOCOL	This string value specifies which row locking protocol to use. Set this value to "PESSIMISTIC" to specify that all UPDATE row locks should be acquired when the rows are read during the process of the UPDATEs. Set this value to "OPTIMISTIC" to specify that all UPDATE row locks should only be acquired when the rows are actually being updated. Please see the Locking and Concurrency topic for more information.
ROWLOCKRETRIES	This string value specifies the number of row lock attempts that the driver should make before issuing an error. The default is "15".
ROWLOCKWAIT	This string value specifies the amount of time (in milliseconds) to wait between each row lock attempt. The default is "100".
DETECTROWCHANGES	This string value specifies whether the driver should issue an error when a row is updated and the row has changed since it was last cached. Specify "TRUE" to enable row change detection, or "FALSE" (the default) to disable row change detection. Please see the Change Detection topic for more information.
CATALOGINFO (L)	This string value specifies whether database catalog character set and version information should appear in the Databases system information table. The default value is "TRUE". **Note** Setting the value to "FALSE" can significantly improve the performance of the loading of the Databases system information table when there are a lot of databases in a configuration. This is because ElevateDB has to open the database catalog for each database in order to read the character set and version number.
CACHEMODULES (L)	This string value specifies whether module DLLs should be cached in memory for the duration of the connection. The default value is "FALSE".
STMTCACHESIZE	This string value specifies how many SQL statements can be cached in memory for the duration of the connection. Caching SQL statements improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is "0", which means that SQL statements will not be cached for the connection.

	Note The maximum number of open SQL statements per connection is 2048, so you should not set the statement cache size that high. Also, the SQL statement cache size is a **per-open-database** setting.
PROCCACHESIZE	This string value specifies how many functions/procedures can be cached in memory for the duration of the connection. Caching functions/procedures improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is "0", which means that functions/procedures will not be cached for the connection. **Note** The maximum number of open functions/procedures per connection is 2048, so you should not set the procedure cache size that high. Also, the function/procedure cache size is a **per-open-database** setting.
FLUSHWRITES	This string value controls whether the driver forces the operating system to flush any buffered writes to disk immediately after the data is written to the operating system. If the value is "FALSE" (the default), then ElevateDB will leave the flushing up to the operating system. If it is "TRUE", then ElevateDB will force a buffer flush after every write. Please see the Buffering and Caching topic for more information.
SIGNATURE	This string value specifies the signature to use for the connection. A signature is used to "sign" all configuration and database files created by ElevateDB so that they are only accessible using that signature, as well as "signing" all communications with a remote ElevateDB Server. You should only specify this value if you know exactly what you are doing and need to use a different signature than the default of "edb_signature".
ENCRYPTPWD (L)	This string value specifies the password to use for encrypting any local engine files. You should only specify this value if you know exactly what you are doing and need to use a different encryption password than the default of "elevatesoft".

1.3 Character Sets

When using the ElevateDB PHP Extension with Western European languages, it is normally possible to code your PHP scripts without specifying a content type or character set in the scripts. This is regardless of whether you are using ANSI or Unicode connections with the ElevateDB PHP Extension. However, if you intend to use a Unicode connection with the ElevateDB PHP Extension with characters sets other than the ISO-8859-1 (Latin1) character set, then it is required that you include a header in your PHP scripts to specify that the content coming from ElevateDB, and ultimately from the script, is using the UTF-8 character set. This will ensure that the content is interpreted/displayed correctly by the client web browser.

> **Note**
> The ElevateDB PHP Extension always encodes/decodes all strings coming into and out of the extension as UTF-8.

The following example shows how you would use the PHP header function to specify the content type and character set at the beginning of your script:

```
<?php

header('Content-type: text/html; charset=utf-8');
```

1.4 Sample Script

The following is a sample script that shows how to connect to a local configuration and database, open a table directly, and then display all of the rows in the table in an HTML table.

```
<?php

// First connect to the database

$con = edb_connect("type=local;charset=Ansi;configpath=C:\\Tutorial;"+
                   "uid=Administrator;pwd=EDBDefault;database=Tutorial");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

// Create a direct table open command

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);

// Execute the command to get the table cursor

$cursor = edb_execute($cmd);

echo "<table border=\"1\" cellpadding=\"3\">";

echo "<tr>";

// Dump out the column headers

for ($i = 0; $i < edb_colcount($cursor); $i++)
   {
   $colinfo = edb_colinfo($cursor,$i);
   echo "<td>" . $colinfo["Name"] . "</td>";
   }

echo "</tr>";

// Now dump out the rows

while (!edb_eof($cursor)):
   echo "<tr>";
   for ($i = 0; $i < edb_colcount($cursor); $i++)
      {
      $col = edb_getcol($cursor,$i);
      if (is_null($col))
         echo "<td>NULL</td>";
      else
         echo "<td>" . $col . "</td>";
      }
   echo "</tr>";
   edb_next($cursor);
endwhile;

echo "</table>";
```

```
?>
```

Chapter 2
Connection Functions

2.1 Introduction

The ElevateDB PHP Extension includes several connection-related functions that provide functionality for connecting to a given database using a connection string, transaction handling, ad-hoc SQL execution, and retrieving server information.

Notation

The notation used in the syntax section for each function is as follows:

Notation	Description
<Element>	Specifies an element of the statement that may be expanded upon further on in the syntax section
<Element> =	Describes an element specified earlier in the syntax section
[Optional Element]	Describes an optional element by enclosing it in square brackets []
Element|Element	Describes multiple elements, of which one and only one may be used in the syntax

2.2 edb_connect

Connects to an ElevateDB configuration/server and database using a connection string.

Syntax

```
edb_connect(<ConnectionString>)

<ConnectionString> =

Semi-colon-separated set of connection string keywords
```

Returns

```
Connection handle if successful, or FALSE if any errors
```

Usage

The edb_connect function uses a connection string to establish a connection to a local ElevateDB configuration or ElevateDB Server, as well as a specific database. Please see the Connection Strings topic for a list of all of the connection string keywords that can be specified. If the connection fails and the edb_connect function returns False, you can use the edb_errcode and edb_errmsg functions to retrieve more information about the cause of the connection failure.

Examples

```
<?php

// The following script connects to a local
// ElevateDB configuration and database, displays
// the configuration path using the edb_connset()
// function, and then disconnects using the edb_disconnect()
// function

$con = edb_connect("type=local;charset=Ansi;configpath=C:\\Tutorial;"+
                   "uid=Administrator;pwd=EDBDefault;database=Tutorial");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_connset($con,"configpath");
edb_disconnect($con);
?>
```

2.3 edb_disconnect

Disconnects a connection, freeing all associated resources.

Syntax

```
edb_disconnect(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
TRUE if successful, or FALSE if any errors
```

Usage

The edb_disconnect function disconnects a connection, freeing the handle and all associate resources. Any attempts to use the connection handle after calling edb_disconnect will result in an error.

Note
Any connections, and their associated resources, are automatically disconnected and freed when a PHP script finishes executing, so use of this function is optional.

Examples

```
<?php

// The following script connects to a local
// ElevateDB configuration and database, displays
// the configuration path using the edb_connset()
// function, and then disconnects using the edb_disconnect()
// function

$con = edb_connect("type=local;charset=Ansi;configpath=C:\\Tutorial;"+
                   "uid=Administrator;pwd=EDBDefault;database=Tutorial");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_connset($con,"configpath");
edb_disconnect($con);
?>
```

2.4 edb_connset

Returns the value of a specific connection string setting for a connection.

Syntax

```
edb_connset(<ConnectionHandle>,<Keyword>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function

<Keyword> = Connection string keyword
```

Returns

```
The value of the connection string setting (STRING) if
successful, or FALSE if any errors
```

Usage

The edb_connset function returns the value of a given connection setting used in the connection string and the edb_connect function.

Examples

```
<?php

// The following script connects to a local
// ElevateDB configuration and database, displays
// the configuration path using the edb_connset()
// function, and then disconnects using the edb_disconnect()
// function

$con = edb_connect("type=local;charset=Ansi;configpath=C:\\Tutorial;"+
                   "uid=Administrator;pwd=EDBDefault;database=Tutorial");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_connset($con,"configpath");
edb_disconnect($con);
?>
```

2.5 edb_user

Returns the current user name for a connection.

Syntax

```
edb_user(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
The user name (STRING) if successful, or
FALSE if any errors
```

Usage

The edb_user function returns the user name for a connection.

Examples

```
<?php

// The following script connects to a local
// ElevateDB configuration and database, displays
// the user name using the edb_user()
// function, and then disconnects using the edb_disconnect()
// function

$con = edb_connect("type=local;charset=Ansi;configpath=C:\\Tutorial;"+
                   "uid=Administrator;pwd=EDBDefault;database=Tutorial");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_user($con);
edb_disconnect($con);
?>
```

2.6 edb_sessid

Returns the current session ID for a connection.

Syntax

```
edb_sessid(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
The session ID (INTEGER) if successful, or FALSE if
any errors
```

Usage

The edb_sessid function returns the session ID for a connection. The session ID is useful for linking the current session to various system information tables that are used for remote management of an ElevateDB Server:

ServerSessions Table
ServerSessionStatistics Table
ServerSessionLocks Table

The session ID is not particularly useful for local connections, but can still be used for logging purposes and for linking events in the logged events system information table:

LogEvents Table

back to logs for a particular session.

Examples

```
<?php

// The following script connects to a local
// ElevateDB configuration and database, displays
// the session ID using the edb_sessid()
// function, and then disconnects using the edb_disconnect()
// function

$con = edb_connect("type=local;charset=Ansi;configpath=C:\\Tutorial;"+
                   "uid=Administrator;pwd=EDBDefault;database=Tutorial");
if (!$con)
  {
  die("Could not connect: " . edb errmsg());
```

```
  }
echo edb_sessid($con);
edb_disconnect($con);
?>
```

2.7 edb_srvver

Returns the current ElevateDB Server version number and build number for a remote connection.

Syntax

```
edb_srvver(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
The ElevateDB Server version/build (STRING) if successful,
or FALSE if there are any errors
```

Usage

The edb_srvver function returns the ElevateDB Server version and build number for a remote connection, and returns an empty string for a local connection.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, displays the server version
// number using the edb_srvver() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_srvver($con);
edb_disconnect($con);
?>
```

2.8 edb_srvdatetime

Returns the current ElevateDB Server date and time for a remote connection.

Syntax

```
edb_srvdatetime(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
The ElevateDB Server date/time (STRING) if successful,
or FALSE if there are any errors
```

Usage

The edb_srvdatetime function returns the ElevateDB Server date and time for a remote connection, and returns an empty string for a local connection. The date and time is returned in ANSI standard date and time format (24-hour), which is:

```
YYYY-MM-DD HH:MM:SS.ZZZ
```

where YYYY is a 4-digit year, MM is a 2-digit month, DD is a 2-digit day, HH is a 2-digit hour, MM is a 2-digit minute, SS is a 2-digit second, and ZZZ is a 3-digit millisecond.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, displays the server date/time
// using the edb_srvdatetime() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_srvdatetime($con);
edb_disconnect($con);
?>
```

2.9 edb_srvutcdatetime

Returns the current ElevateDB Server UTC date and time for a remote connection.

Syntax

```
edb_srvutcdatetime(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
The ElevateDB Server UTC date/time (STRING) if
successful, or FALSE if there are any errors
```

Usage

The edb_srvutcdatetime function returns the ElevateDB Server UTC date and time for a remote connection, and returns an empty string for a local connection. UTC is the symbol for Coordinated Universal Time, which is similar to Greenwich Mean Time, but more precise down to fractions of seconds, and is useful for situations where one needs to compare dates and times, irrespective of the client machine's time zone. The date and time is returned in ANSI standard date and time format (24-hour), which is:

```
YYYY-MM-DD HH:MM:SS.ZZZ
```

where YYYY is a 4-digit year, MM is a 2-digit month, DD is a 2-digit day, HH is a 2-digit hour, MM is a 2-digit minute, SS is a 2-digit second, and ZZZ is a 3-digit millisecond.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, displays the server date/time
// using the edb_srvutcdatetime() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
```

```
echo edb_srvutcdatetime($con);
edb_disconnect($con);
?>
```

2.10 edb_srvname

Returns the current ElevateDB Server name for a remote connection.

Syntax

```
edb_srvname(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
The ElevateDB Server name (STRING) if successful, or
FALSE if there are any errors
```

Usage

The edb_srvname function returns the ElevateDB Server name for a remote connection, and returns an empty string for a local connection.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, displays the server name
// using the edb_srvname() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_srvname($con);
edb_disconnect($con);
?>
```

2.11 edb_srvdesc

Returns the current ElevateDB Server description for a remote connection.

Syntax

```
edb_srvname(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
The ElevateDB Server description (STRING) if successful,
or FALSE if there are any errors
```

Usage

The edb_srvdesc function returns the ElevateDB Server description for a remote connection, and returns an empty string for a local connection.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, displays the server description
// using the edb_srvdesc() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_srvdesc($con);
edb_disconnect($con);
?>
```

2.12 edb_db

Returns the current database name for a connection.

Syntax

```
edb_db(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
The current database name (STRING) if successful, or
FALSE if there are any errors
```

Usage

The edb_db function returns the current database name for a connection. There is always a current database for any given connection, so this function will always return a value unless there is an error.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, displays the database
// name using the edb_db() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_db($con);
edb_disconnect($con);
?>
```

2.13 edb_changedb

Changes the current database for a connection.

Syntax

```
edb_changedb(<ConnectionHandle>, <DatabaseName>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function

<DatabaseName> = Valid database name
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_changedb function changes the current database for a connection. This is especially useful when you want to change to the system Configuration database for the purposes of querying system information tables, and then change back to a user-defined database for normal operations.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and the system Configuration database,
// creates a new database if it doesn't already exist,
// changes to the new database using the edb_changedb()
// function, and then disconnects using the
// edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Configuration");

if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_db($con);
if (edb_execsql($con,"SELECT * FROM Databases WHERE Name='Tutorial'") == 0)
  {
  edb_execsql($con,"CREATE DATABASE Tutorial PATH 'C:\Tutorial'");
  }
edb_changedb($con,"Tutorial");
echo edb_db($con);
```

```
edb_disconnect($con);
?>
```

2.14 edb_execsql

Executes an SQL statement directly against the current database.

Syntax

```
edb_execsql(<ConnectionHandle>, <SQL>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function

<SQL> = Valid SQL statement
```

Returns

```
Affected row count (where applicable) if successful,
or FALSE if there are any errors
```

Usage

The edb_execsql function executes an SQL statement directly. This is useful for DDL statements and other statements where you do not need to bind parameters or use the resultant cursor (SELECT statements).

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and the system Configuration database,
// creates a new database if it doesn't already exist,
// changes to the new database using the edb_changedb()
// function, and then disconnects using the
// edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Configuration");

if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
echo edb_db($con);
if (edb_execsql($con,"SELECT * FROM Databases WHERE Name='Tutorial'") == 0)
  {
  edb_execsql($con,"CREATE DATABASE Tutorial PATH 'C:\Tutorial'");
  }
edb_changedb($con,"Tutorial");
echo edb_db($con);
edb_disconnect($con);
```

```
?>
```

2.15 edb_starttrans

Starts a transaction on one or more tables in the current database.

Syntax

```
edb_starttrans(<ConnectionHandle> [, <TableNames>])

<ConnectionHandle> =

Handle of connection returned by edb_connect function

<TableNames> = A single table name (STRING), or an
array of table names for multiple tables
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_starttrans function starts a transaction on one or more tables in the current database. You can specify the tables that you want to be involved in the transaction as a single string for one table or an array of strings for multiple tables. If you do not specify the table names, then the transaction will cover all tables in the current database.

Please see the Transactions topic in the SQL manual for more information on how transactions work in ElevateDB.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, starts a transaction on
// the Customer and Orders tables using the
// edb_starttrans() function, inserts a row
// into each table, commits the transaction
// using the edb_commit() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

edb_starttrans($con,array("Customer","Orders"));
```

```
echo edb_intrans($con);

$custcmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$custcursor = edb_execute($custcmd);

$ordcmd = edb_prepare($con,"orders",EDB_COMMAND_TABLE);
$ordcursor = edb_execute($ordcmd);

edb_insert($custcursor);
edb_setcol($custcursor,"CustNo",1000);
edb_setcol($custcursor,"Company","Test Company");
if (!edb_post($custcursor))
  {
  $msg = edb_errmsg();
  edb_rollback($con);
  die("Error adding customer: " . $msg);
  }

edb_insert($ordcursor);
edb_setcol($ordcursor,"CustNo",1000);
edb_setcol($ordcursor,"OrderNo",100);
edb_setcol($ordcursor,"SaleDate","2010-02-24");
edb_setcol($ordcursor,"EmpNo",10);
if (!edb_post($ordcursor))
  {
  $msg = edb_errmsg();
  edb_rollback($con);
  die("Error adding order: " . $msg);
  }

edb_commit($con);

edb_disconnect($con);
?>
```

2.16 edb_commit

Commits the current transaction in the current database.

Syntax

```
edb_commit(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_commit function commits the current transaction in the current database.

Please see the Transactions topic in the SQL manual for more information on how transactions work in ElevateDB.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, starts a transaction on
// the Customer and Orders tables using the
// edb_starttrans() function, inserts a row
// into each table, commits the transaction
// using the edb_commit() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

edb_starttrans($con,array("Customer","Orders"));

echo edb_intrans($con);

$custcmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$custcursor = edb_execute($custcmd);
```

```
$ordcmd = edb_prepare($con,"orders",EDB_COMMAND_TABLE);
$ordcursor = edb_execute($ordcmd);

edb_insert($custcursor);
edb_setcol($custcursor,"CustNo",1000);
edb_setcol($custcursor,"Company","Test Company");
if (!edb_post($custcursor))
  {
  $msg = edb_errmsg();
  edb_rollback($con);
  die("Error adding customer: " . $msg);
  }

edb_insert($ordcursor);
edb_setcol($ordcursor,"CustNo",1000);
edb_setcol($ordcursor,"OrderNo",100);
edb_setcol($ordcursor,"SaleDate","2010-02-24");
edb_setcol($ordcursor,"EmpNo",10);
if (!edb_post($ordcursor))
  {
  $msg = edb_errmsg();
  edb_rollback($con);
  die("Error adding order: " . $msg);
  }

edb_commit($con);

edb_disconnect($con);
?>
```

2.17 edb_rollback

Rolls back the current transaction in the current database.

Syntax

```
edb_rollback(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_rollback function rolls back the current transaction in the current database.

Please see the Transactions topic in the SQL manual for more information on how transactions work in ElevateDB.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, starts a transaction on
// the Customer and Orders tables using the
// edb_starttrans() function, inserts a row
// into each table, commits the transaction
// using the edb_commit() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

edb_starttrans($con,array("Customer","Orders"));

echo edb_intrans($con);

$custcmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$custcursor = edb_execute($custcmd);
```

```
$ordcmd = edb_prepare($con,"orders",EDB_COMMAND_TABLE);
$ordcursor = edb_execute($ordcmd);

edb_insert($custcursor);
edb_setcol($custcursor,"CustNo",1000);
edb_setcol($custcursor,"Company","Test Company");
if (!edb_post($custcursor))
  {
  $msg = edb_errmsg();
  edb_rollback($con);
  die("Error adding customer: " . $msg);
  }

edb_insert($ordcursor);
edb_setcol($ordcursor,"CustNo",1000);
edb_setcol($ordcursor,"OrderNo",100);
edb_setcol($ordcursor,"SaleDate","2010-02-24");
edb_setcol($ordcursor,"EmpNo",10);
if (!edb_post($ordcursor))
  {
  $msg = edb_errmsg();
  edb_rollback($con);
  die("Error adding order: " . $msg);
  }

edb_commit($con);

edb_disconnect($con);
?>
```

2.18 edb_intrans

Indicates whether a transaction is active for the current database.

Syntax

```
edb_intrans(<ConnectionHandle>)

<ConnectionHandle> =

Handle of connection returned by edb_connect function
```

Returns

```
TRUE if the current database is in a transaction,
or FALSE if there are any errors
```

Usage

The edb_intrans function indicates whether there is an active transaction for the current database.

Please see the Transactions topic in the SQL manual for more information on how transactions work in ElevateDB.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, starts a transaction on
// the Customer and Orders tables using the
// edb_starttrans() function, inserts a row
// into each table, commits the transaction
// using the edb_commit() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

edb_starttrans($con,array("Customer","Orders"));

echo edb_intrans($con);

$custcmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$custcursor = edb_execute($custcmd);
```

```
$ordcmd = edb_prepare($con,"orders",EDB_COMMAND_TABLE);
$ordcursor = edb_execute($ordcmd);

edb_insert($custcursor);
edb_setcol($custcursor,"CustNo",1000);
edb_setcol($custcursor,"Company","Test Company");
if (!edb_post($custcursor))
  {
  $msg = edb_errmsg();
  edb_rollback($con);
  die("Error adding customer: " . $msg);
  }

edb_insert($ordcursor);
edb_setcol($ordcursor,"CustNo",1000);
edb_setcol($ordcursor,"OrderNo",100);
edb_setcol($ordcursor,"SaleDate","2010-02-24");
edb_setcol($ordcursor,"EmpNo",10);
if (!edb_post($ordcursor))
  {
  $msg = edb_errmsg();
  edb_rollback($con);
  die("Error adding order: " . $msg);
  }

edb_commit($con);

edb_disconnect($con);
?>
```

2.19 edb_jsonccommit

Commits an Elevate Web Builder transaction using JSON transaction data.

Syntax

```
edb_jsoncommit(<ConnectionHandle>, <DataSetTables>,
                       <TransactionJSON>, [<LocalizeDateTimeColumns>])

<ConnectionHandle> =

Handle of connection returned by edb_connect function

<DataSetTables> =

Associative array of dataset names and base table names

<TransactionJSON> =

String containing JSON transaction data

<LocalizeDateTimeColumns> =

Boolean indicating whether to convert date/time columns to local time
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_jsoncommit function is used to consume JSON transactions from an Elevate Web Builder application. The DataSetTables parameter is an associative array of dataset names and their associated base table names. This information is used to determine which base tables should be updated during the transaction commit. If any datasets in the incoming JSON are not specified in the DataSetTables associative array, then an error will result.

The LocalizeDateTimeColumns parameter indicates whether date/time columns should be converted from UTC time to local time before being stored in the database. The default value for this parameter is False.

Please see the Elevate Web Builder manual for more information on datasets and the JSON data formats used with them.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, and commits an Elevate
```

```
// Web Builder JSON transaction on
// the Customer and Orders tables using the
// edb_jsoncommit() function, and then
// disconnects using the edb_disconnect() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

if ($_GET["method"])
   {
   switch ($_GET["method"])
      {
      case "commit":
         if (!edb_jsoncommit($con, array("Customer" => "customer",
                                         "Orders" => "orders"),
                                         $HTTP_RAW_POST_DATA))
            {
            die("Error committing database transaction (" . edb_errmsg() .
      ")";
            }
         break;
      }
  }
else
  {
  die("Database method not specified";
  }

edb_disconnect($con);
?>
```

2.20 edb_freecachedstmts

Frees any cached SQL statements for the specified open database.

Syntax

```
edb_freecachedstmts(<ConnectionHandle>, [<DatabaseName>])

<ConnectionHandle> =

Handle of connection returned by edb_connect function

<DatabaseName> =

Name of open database that contains cached SQL statements to free
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_freecachedstmts function is used to free any cached SQL statements for the specified open database. If a database is not specified, then any cached SQL statements in the open databases for the connection will be freed. Please see the Buffering and Caching topic for more information on caching SQL statements.

```
<?php

// The following script connects to an ElevateDB
// Server and database, executes a series of queries,
// and then calls the edb_freecachedstmts() and
// edb_freecachedprocs() functions to free any cached
// SQL statements and functions/procedures.  This is for
// example purposes only - you would not normally do
// this because it defeats the the purpose of the
// SQL statement and function/procedure caching.

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test;"+
                   "stmtcachesize=32;proccachesize=32");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

edb_execsql($con,"SELECT * FROM customer WHERE State='FL' AND "+
                 "TotalOrders(CustNo) > 10");
edb_execsql($con,"SELECT * FROM customer WHERE State='NY' AND "+
                 "TotalOrders(CustNo) > 10");
```

```
edb_execsql($con,"SELECT * FROM customer WHERE State='CA' AND "+
                 "TotalOrders(CustNo) > 10");

edb_freecachedstmts($con);
edb_freecachedprocs($con);

edb_disconnect($con);
?>
```

2.21 edb_freecachedprocs

Frees any cached functions/procedures for the specified open database.

Syntax

```
edb_freecachedprocs(<ConnectionHandle>, [<DatabaseName>])

<ConnectionHandle> =

Handle of connection returned by edb_connect function

<DatabaseName> =

Name of open database that contains cached functions/procedures to free
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_freecachedprocss function is used to free any cached functions/procedures for the specified open database. If a database is not specified, then any cached functions/procedures in the open databases for the connection will be freed. Please see the Buffering and Caching topic for more information on caching functions/procedures.

```
<?php

// The following script connects to an ElevateDB
// Server and database, executes a series of queries,
// and then calls the edb_freecachedstmts() and
// edb_freecachedprocs() functions to free any cached
// SQL statements and functions/procedures.  This is for
// example purposes only - you would not normally do
// this because it defeats the the purpose of the
// SQL statement and function/procedure caching.

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test;"+
                   "stmtcachesize=32;proccachesize=32");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

edb_execsql($con,"SELECT * FROM customer WHERE State='FL' AND "+
                 "TotalOrders(CustNo) > 10");
edb_execsql($con,"SELECT * FROM customer WHERE State='NY' AND "+
                 "TotalOrders(CustNo) > 10");
```

```
edb_execsql($con,"SELECT * FROM customer WHERE State='CA' AND "+
                 "TotalOrders(CustNo) > 10");

edb_freecachedstmts($con);
edb_freecachedprocs($con);

edb_disconnect($con);
?>
```

This page intentionally left blank

Chapter 3
Command Functions

3.1 Introduction

The ElevateDB PHP Extension includes several command-related functions that provide functionality for executing SQL statements, scripts, directly-opening tables, and executing procedures/functions, getting/setting parameters, and retrieving information about a command execution.

Notation

The notation used in the syntax section for each function is as follows:

Notation	Description
<Element>	Specifies an element of the statement that may be expanded upon further on in the syntax section
<Element> =	Describes an element specified earlier in the syntax section
[Optional Element]	Describes an optional element by enclosing it in square brackets []
Element|Element	Describes multiple elements, of which one and only one may be used in the syntax

3.2 edb_prepare

Prepares a command using an SQL statement, direct table access, or a stored procedure/function.

Syntax

```
edb_prepare(<ConnectionHandle>, <Command> [, <CommandType>])

<ConnectionHandle> =

Handle of connection returned by edb_connect function

<Command> =

SQL Statement |
Table Name |
Procedure/Function Name

<CommandType> =

EDB_COMMAND_TEXT (0) |
EDB_COMMAND_TABLE (1) |
EDB_COMMAND_PROCFUNC (2)
```

Returns

```
Command handle (INTEGER) if successful, or
FALSE if there are any errors
```

Usage

The edb_prepare function prepares a command, returning a newly-allocated command handle as a result. A command can be one of 3 types:

Command Type	Description

EDB_COMMAND_TEXT	This command type (the default) allows you to specify any valid SQL statement or script as the command. If the command is an SQL statement and is a DML statement, it may also contained named parameters in the form of :<ParameterName>, where <ParameterName> is the name of the parameter to use. If the command is a script, then after executing this function, the named parameters for such a script will automatically be available.
EDB_COMMAND_TABLE	This command type allows you to specify a table name that you wish to open directly. Opening a table directly will allow you to perform index-based operations that you cannot necessarily perform with an SQL result set, such as changing the active index, finding a specific key, and setting a range on a set of starting and ending keys.
EDB_COMMAND_PROCFUNC	This command type allows you to specify a stored procedure/function name that you wish to execute. After executing this function for a stored procedure/function, the named parameters for such a procedure/function will automatically be available. Functions also have one extra system-generated parameter named "Result" that will contain the result of a function after it is executed.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter values,
// executes the statement, and then displays the
// result set in an HTML table

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");
$cursor = edb_execute($cmd);

echo "<table border=\"1\" cellpadding=\"3\">";

echo "<tr>";

// Dump out the column headers

for ($i = 0; $i < edb_colcount($cursor); $i++)
   {
   $colinfo = edb_colinfo($cursor,$i);
   echo "<td>" . $colinfo["Name"] . "</td>";
   }
```

```
echo "</tr>";

// Now dump out the rows

while (!edb_eof($cursor)):
   echo "<tr>";
   for ($i = 0; $i < edb_colcount($cursor); $i++)
      {
      $col = edb_getcol($cursor,$i);
      if (is_null($col))
         echo "<td>NULL</td>";
      else
         echo "<td>" . $col . "</td>";
      }
   echo "</tr>";
   edb_next($cursor);
endwhile;

echo "</table>";

edb_disconnect($con);
?>
```

3.3 edb_paramcount

Returns the number of named parameters for a command.

Syntax

```
edb_paramcount(<CommandHandle>)

<CommandHandle> =

Handle of command returned by edb_prepare function
```

Returns

```
The number of parameters (INTEGER) if successful, or
FALSE if there are any errors
```

Usage

The edb_paramcount function returns the number of named parameters in a prepared command.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, and then displays the parameter
// count using the edb_paramcount() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");

echo edb_paramcount($cmd);

edb_disconnect($con);
?>
```

3.4 edb_paraminfo

Returns information about a given parameter as an associative array.

Syntax

```
edb_paraminfo(<CommandHandle>, <ParamNumber> | <ParamName>)

<CommandHandle> =

Handle of command returned by edb_prepare function

<ParamNumber> = Ordinal position of parameter

<ParamName> = Name of parameter
```

Returns

```
Associative array of parameter information
if successful, or FALSE if there are any errors
```

Usage

The edb_paraminfo function returns the following information about the specified parameter:

Key	Description
Name	The name of the parameter
Type	The type of the parameter, which can be one of the following: EDB_PARAM_INPUT_TYPE (1) EDB_PARAM_OUTPUT_TYPE (2) EDB_PARAM_INPUTOUTPUT_TYPE (3) EDB_PARAM_RESULT_TYPE (4)
DataType	The data type of the parameter, which can be any of the valid ElevateDB data types. For a list of valid data types in ElevateDB, please see the DataTypes system information table.
Collation	The collation for any CHAR/VARCHAR/CLOB parameter. For a list of valid collations in ElevateDB, please see the Collations system information table.
Length	The length of a CHAR/VARCHAR/BYTE/VARBYTE parameter.
Scale	The scale of a DECIMAL/NUMERIC parameter.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, and then displays the parameter
// information using the var_dump() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");

var_dump(edb_paraminfo($cmd,"State"));

edb_disconnect($con);
?>
```

3.5 edb_getparamtype

Returns the parameter type for a given parameter.

Syntax

```
edb_getparamtype(<CommandHandle>, <ParamNumber> | <ParamName>)

<CommandHandle> =

Handle of command returned by edb_prepare function

<ParamNumber> = Ordinal position of parameter

<ParamName> = Name of parameter
```

Returns

```
Parameter type (INTEGER) if successful, or
FALSE if there are any errors
```

Usage

The edb_getparamtype function returns the type of the parameter as one of the following values:

Type	Description
EDB_PARAM_INPUT_TYPE (1)	The parameter is an input parameter
EDB_PARAM_OUTPUT_TYPE (2)	The parameter is an output parameter
EDB_PARAM_INPUTOUTPUT_TYPE (3)	The parameter is both and input and output parameter
EDB_PARAM_RESULT_TYPE (4)	The parameter is a result parameter (functions)

Note
Only INSERT SQL statements, scripts, or procedures/functions can have output or result parameters.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, and then displays the parameter
// type using the edb_getparamtype() function
```

```
$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");

edb_getparamtype($cmd,"State");

edb_disconnect($con);
?>
```

3.6 edb_setparamtype

Sets the parameter type for a given parameter.

Syntax

```
edb_setparamtype(<CommandHandle>,
                 <ParamNumber> | <ParamName>,
                 <ParamType>)

<CommandHandle> =

Handle of command returned by edb_prepare function

<ParamNumber> = Ordinal position of parameter

<ParamName> = Name of parameter

<ParamType> = EDB_PARAM_INPUT_TYPE (1) |
EDB_PARAM_OUTPUT_TYPE (2) |
EDB_PARAM_INPUTOUTPUT_TYPE (3) |
EDB_PARAM_RESULT_TYPE (4)
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_setparamtype function sets the type of the parameter as one of the following values:

Type	Description
EDB_PARAM_INPUT_TYPE (1)	The parameter is an input parameter
EDB_PARAM_OUTPUT_TYPE (2)	The parameter is an output parameter
EDB_PARAM_INPUTOUTPUT_TYPE (3)	The parameter is both and input and output parameter
EDB_PARAM_RESULT_TYPE (4)	The parameter is a result parameter (functions)

Note
Only INSERT SQL statements, scripts, or procedures/functions can have output or result parameters. In all but a few rare cases, you will never need to use this function.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, and then sets the parameter
// type using the edb_setparamtype() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");

edb_setparamtype($cmd,"State",EDB_PARAM_INPUT_TYPE);

edb_disconnect($con);
?>
```

3.7 edb_getparam

Gets the parameter value for a given parameter.

Syntax

```
edb_getparam(<CommandHandle>, <ParamNumber> | <ParamName>)

<CommandHandle> =

Handle of command returned by edb_prepare function

<ParamNumber> = Ordinal position of parameter

<ParamName> = Name of parameter
```

Returns

```
Parameter value if successful, or FALSE if there are any errors
```

Usage

The edb_getparam function retrieves the value of the specified parameter.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter value
// using the edb_setparam() function, and then
// displays the parameter value using the
// edb_getparam() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");

edb_setparam($cmd,"State","FL");

echo edb_getparam($cmd,"State");

edb_disconnect($con);
?>
```

3.8 edb_setparam

Sets the parameter value for a given parameter.

Syntax

```
edb_setparam(<CommandHandle>,
             <ParamNumber> | <ParamName>,
             <Value>)

<CommandHandle> =

Handle of command returned by edb_prepare function

<ParamNumber> = Ordinal position of parameter

<ParamName> = Name of parameter

<Value> = Any valid value
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_setparam function sets the value of the specified parameter.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter value
// using the edb_setparam() function, and then
// displays the parameter value using the
// edb_getparam() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");

edb_setparam($cmd,"State","FL");
```

```
echo edb_getparam($cmd,"State");

edb_disconnect($con);
?>
```

3.9 edb_getparams

Gets all parameter values for a command as an associative array.

Syntax

```
edb_getparams(<CommandHandle>)

<CommandHandle> =

Handle of command returned by edb_prepare function
```

Returns

```
Associative array of parameter values if
successful, or FALSE if there are any errors
```

Usage

The edb_getparams function retrieves all of the parameter values for a command.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter value
// using the edb_setparam() function, and then
// displays the parameter values using the
// edb_getparams() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");

edb_setparam($cmd,"State","FL");

var_dump(edb_getparams($cmd));

edb_disconnect($con);
?>
```

3.10 edb_setparams

Sets the parameter values for a command using an associative array.

Syntax

```
edb_setparams(<CommandHandle>, <Values>)

<CommandHandle> =

Handle of command returned by edb_prepare function

<Values> = Associative array of parameter names and values
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_setparams function sets the parameter values for a command using an associative array of parameter names (as the keys) along with the parameter values.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter value
// using the edb_setparams() function, and then
// displays the parameter values using the
// edb_getparams() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");

edb_setparams($cmd,array("State"=>"FL"));

var_dump(edb_getparams($cmd));

edb_disconnect($con);
?>
```

3.11 edb_unprepare

Unprepares a command, freeing all associated resources.

Syntax

```
edb_unprepare(<CommandHandle>)

<CommandHandle> =

Handle of command returned by edb_prepare function
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_unprepare function unprepares a command, freeing the command handle and releasing all resources associated with the command. Any attempts to use the command handle after calling edb_unprepare will result in an error.

Warning
If you have executed a command using the edb_execute() function to return a cursor, the cursor will also become invalid after calling edb_unprepare.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter values,
// executes the statement, displays the result set
// in an HTML table, and then unprepares the command
// using the edb_unprepare() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");
$cursor = edb_execute($cmd);
```

```
echo "<table border=\"1\" cellpadding=\"3\">";

echo "<tr>";

// Dump out the column headers

for ($i = 0; $i < edb_colcount($cursor); $i++)
   {
   $colinfo = edb_colinfo($cursor,$i);
   echo "<td>" . $colinfo["Name"] . "</td>";
   }

echo "</tr>";

// Now dump out the rows

while (!edb_eof($cursor)):
   echo "<tr>";
   for ($i = 0; $i < edb_colcount($cursor); $i++)
      {
      $col = edb_getcol($cursor,$i);
      if (is_null($col))
         echo "<td>NULL</td>";
      else
         echo "<td>" . $col . "</td>";
      }
   echo "</tr>";
   edb_next($cursor);
endwhile;

echo "</table>";

edb_unprepare($cmd);

edb_disconnect($con);
?>
```

3.12 edb_execute

Executes a command, returning a cursor if the command is a SELECT statement, a script that returns a cursor, a direct table command, or a procedure that returns a cursor.

Syntax

```
edb_execute(<CommandHandle>
            [[, <RequestSensitive>]
             [,<RequestExecutionPlan>]])

<CommandHandle> =

Handle of command returned by edb_prepare function

<RequestSensitive> =

If TRUE, requests a sensitive result set for SELECT statements

<RequestExecutionPlan> =

If TRUE, requests an execution plan for a SELECT, INSERT,
UPDATE, or DELETE statement
```

Returns

```
Cursor handle if successful, or FALSE if there are any errors
```

Usage

The edb_execute function executes a command, returning a cursor if the command is a SELECT statement, a script that returns a cursor, a direct table command, or a procedure that returns a cursor. If the command does not return a cursor, then the cursor handle returned will be 0.

You can request that a SELECT statement return a sensitive result set by setting the RequestSensitive parameter to TRUE. For more information on sensitive and insensitive result sets, please see the Result Set Cursor Sensitivity topic in the SQL manual.

You can request an execution plan for any SELECT, INSERT, UPDATE, or DELETE statement by setting the RequestExecutionPlan parameter to TRUE. You can then retrieve the execution plan using the edb_execplan function.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter values,
```

```
// executes the statement, and then displays the
// result set in an HTML table

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");
$cursor = edb_execute($cmd);

echo "<table border=\"1\" cellpadding=\"3\">";

echo "<tr>";

// Dump out the column headers

for ($i = 0; $i < edb_colcount($cursor); $i++)
   {
   $colinfo = edb_colinfo($cursor,$i);
   echo "<td>" . $colinfo["Name"] . "</td>";
   }

echo "</tr>";

// Now dump out the rows

while (!edb_eof($cursor)):
   echo "<tr>";
   for ($i = 0; $i < edb_colcount($cursor); $i++)
      {
      $col = edb_getcol($cursor,$i);
      if (is_null($col))
         echo "<td>NULL</td>";
      else
         echo "<td>" . $col . "</td>";
      }
   echo "</tr>";
   edb_next($cursor);
endwhile;

echo "</table>";

edb_disconnect($con);
?>
```

3.13 edb_exectime

Returns the execution time for a command.

Syntax

```
edb_exectime(<CommandHandle>)

<CommandHandle> =

Handle of command returned by edb_prepare function
```

Returns

```
Execution time (DOUBLE) if successful, or FALSE
if there are any errors
```

Usage

The edb_exectime function returns the execution time for any text or procedure/function command. Direct table commands do not return execution times, since they are only opening a table directly.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter values,
// executes the statement, and then displays the
// execution time

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");
$cursor = edb_execute($cmd);

echo edb_exectime($cmd);

edb_disconnect($con);
?>
```

3.14 edb_execresult

Returns the execution result for a command.

Syntax

```
edb_execresult(<CommandHandle>)

<CommandHandle> =

Handle of command returned by edb_prepare function
```

Returns

```
Execution result (BOOLEAN) if successful, or FALSE
if there are any errors
```

Usage

The edb_execresult function returns the execution result for any text command that can return a result. At this time, only the VERIFY TABLE and REPAIR TABLE statements set an execution result.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a VERIFY TABLE
// statement, executes the statement, and then
// displays execution result

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"VERIFY TABLE customer");
$cursor = edb_execute($cmd);

echo edb_execresult($cmd);

edb_disconnect($con);
?>
```

3.15 edb_execplan

Returns the execution plan for a command.

Syntax

```
edb_execplan(<CommandHandle>)

<CommandHandle> =

Handle of command returned by edb_prepare function
```

Returns

```
Execution plan (STRING) if successful, or FALSE
if there are any errors
```

Usage

The edb_execplan function returns the execution plan for any SELECT, INSERT, UPDATE, or DELETE statement.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter values,
// executes the statement, and then displays the
// execution plan

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");
$cursor = edb_execute($cmd,TRUE,TRUE);

echo "<pre>" . edb_execplan($cmd) . "</pre>";

edb_disconnect($con);
?>
```

3.16 edb_rowsaffected

Returns the number of rows affected by, or returned by, a command.

Syntax

```
edb_rowsaffected(<CommandHandle>)

<CommandHandle> =

Handle of command returned by edb_prepare function
```

Returns

```
Rows affected (INTEGER) if successful, or FALSE
if there are any errors
```

Usage

The edb_rowsaffected function returns the number of rows affected for any SELECT, INSERT, UPDATE, or DELETE statement.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter values,
// executes the statement, and then displays the
// number of affected rows

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");
$cursor = edb_execute($cmd,TRUE);

echo edb_rowsaffected($cmd);

edb_disconnect($con);
?>
```

3.17 edb_sensitive

Returns the whether a command returned a sensitive result set cursor.

Syntax

```
edb_sensitive(<CommandHandle>)

<CommandHandle> =

Handle of command returned by edb_prepare function
```

Returns

```
Result set cursor sensitivity (BOOLEAN) if successful,
or FALSE if there are any errors
```

Usage

The edb_sensitive function returns whether any SELECT statement returned a sensitive result set cursor.

> **Note**
> Scripts and procedures can also return sensitive result set cursors, but the ElevateDB PHP extension cannot detect their status, so this function will always return FALSE. Also, direct table commands always return FALSE for the edb_sensitive function, but are always sensitive due to the fact that they are direct table opens.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter values,
// executes the statement, and then displays the
// whether the result set cursor is sensitive or not

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");
$cursor = edb_execute($cmd,TRUE);
```

```
if (edb_sensitive($cmd))
  {
  echo "Command returned a sensitive result set cursor";
  }
else
  {
  echo "Command did not return a sensitive result set cursor";
  }

edb_disconnect($con);
?>
```

Chapter 4
Cursor Functions

4.1 Introduction

The ElevateDB PHP Extension includes several cursor-related functions that provide functionality for navigating, searching, and updating cursors.

Notation

The notation used in the syntax section for each function is as follows:

Notation	Description
<Element>	Specifies an element of the statement that may be expanded upon further on in the syntax section
<Element> =	Describes an element specified earlier in the syntax section
[Optional Element]	Describes an optional element by enclosing it in square brackets []
Element|Element	Describes multiple elements, of which one and only one may be used in the syntax

4.2 edb_rowcount

Returns the row count of a cursor.

Syntax

```
edb_rowcount(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
Cursor row count (INTEGER) if successful, or FALSE if there are any errors
```

Usage

The edb_rowcount function returns the number of rows present in a cursor. The row count can change depending upon whether there is an active filter and/or range present on the cursor.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter values,
// executes the statement, and then displays the
// the result set cursor row count

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");
$cursor = edb_execute($cmd,TRUE);

echo edb_rowcount($cursor);

edb_disconnect($con);
?>
```

4.3 edb_colcount

Returns the column count of a cursor.

Syntax

```
edb_colcount(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
Cursor column count (INTEGER) if successful, or FALSE if there are any errors
```

Usage

The edb_colcount function returns the number of columns present in a cursor.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
// SELECT statement, sets the parameter values,
// executes the statement, and then displays the
// result set in an HTML table by looping through
// the columns using the edb_colcount() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");
edb_setparam($cmd,"State","FL");
$cursor = edb_execute($cmd);

echo "<table border=\"1\" cellpadding=\"3\">";

echo "<tr>";

// Dump out the column headers

for ($i = 0; $i < edb_colcount($cursor); $i++)
   {
```

```
   $colinfo = edb_colinfo($cursor,$i);
   echo "<td>" . $colinfo["Name"] . "</td>";
   }

echo "</tr>";

// Now dump out the rows

while (!edb_eof($cursor)):
   echo "<tr>";
   for ($i = 0; $i < edb_colcount($cursor); $i++)
      {
      $col = edb_getcol($cursor,$i);
      if (is_null($col))
         echo "<td>NULL</td>";
      else
         echo "<td>" . $col . "</td>";
      }
   echo "</tr>";
   edb_next($cursor);
endwhile;

echo "</table>";

edb_disconnect($con);
?>
```

4.4 edb_colinfo

Returns information about a given cursor column as an associative array.

Syntax

```
edb_colinfo(<CursorHandle>, <ColumnNumber> | <ColumnName>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<ColumnNumber> = Ordinal position of column

<ColumnName> = Name of column
```

Returns

```
Associative array of column information
if successful, or FALSE if there are any errors
```

Usage

The edb_colinfo function returns the following information about the specified column:

Key	Description
Name	The name of the column
DataType	The data type of the column, which can be any of the valid ElevateDB data types. For a list of valid data types in ElevateDB, please see the DataTypes system information table.
Collation	The collation for any CHAR/VARCHAR/CLOB column. For a list of valid collations in ElevateDB, please see the Collations system information table.
Length	The length of a CHAR/VARCHAR/BYTE/VARBYTE column.
Scale	The scale of a DECIMAL/NUMERIC column.
Nullable	Whether or not the column is nullable.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a parameterized
```

```
// SELECT statement, executes the statement, and
// then displays the cursor column information
// using the var_dump() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT State, COUNT(State) AS Total FROM customer
      GROUP BY State");

$cursor = edb_execute($cmd);

var_dump(edb_colinfo($cursor,"Total"));

edb_disconnect($con);
?>
```

4.5 edb_getcol

Gets the column value for a given column.

Syntax

```
edb_getcol(<CursorHandle>, <ColumnNumber> | <ColumnName>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<ColumnNumber> = Ordinal position of column

<ColumnName> = Name of column
```

Returns

```
Column value if successful, or FALSE if there are any errors
```

Usage

The edb_getcolumn function retrieves the value of the specified column.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a columnized
// SELECT statement, executs the statement, and
// displays the column value for the first row
// using the edb_getcol() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer WHERE State=:State");

edb_setparam($cmd,"State","FL");

$cursor = edb_execute($cmd);

echo edb_getcol($cursor,"State");

edb_disconnect($con);
```

```
?>
```

4.6 edb_setcol

Sets the column value for a given column.

Syntax

```
edb_setcol(<CursorHandle>,
             <ColumnNumber> | <ColumnName>,
             <Value>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<ColumnNumber> = Ordinal position of column

<ColumnName> = Name of column

<Value> = Any valid value
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_setcolumn function sets the value of the specified column.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, updates the first row, sets
// the column value using the edb_setcolumn()
// function, and then posts the updates

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_update($cursor);
edb_setcol($cursor,"State","NY");
```

```
edb_post($cursor);

edb_disconnect($con);
?>
```

4.7 edb_getrow

Gets all column values for the current row in a cursor as an associative array.

Syntax

```
edb_getrow(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
Associative array of column values if
successful, or FALSE if there are any errors
```

Usage

The edb_getrow function retrieves all of the column values for the current row in a cursor.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, and then displays the column
// values for the first row in the table

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

var_dump(edb_getrow($cursor));

edb_disconnect($con);
?>
```

4.8 edb_setrow

Sets the column values for the current row in a cursor using an associative array.

Syntax

```
edb_setrow(<CursorHandle>, <Values>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<Values> = Associative array of column names and values
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_setrow function sets the column values for the current row in a cursor using an associative array of column names (as the keys) along with the column values.

Warning
The cursor state, retrievable via the edb_state function, must be in an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE) in order for this function to succeed.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, updates the first row, sets
// the column value using the edb_setrow()
// function, and then posts the updates

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);
```

```
edb_update($cursor);
edb_setrow($cursor,array("State"=>"NY"));
edb_post($cursor);

edb_disconnect($con);
?>
```

4.9 edb_readonly

Indicates whether a cursor is read-only or not.

Syntax

```
edb_readonly(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if the cursor is read-only, or FALSE if
not (or there are any errors)
```

Usage

The edb_readonly function returns whether a cursor is read-only or not. A cursor may be read-only under the following conditions:

- Insensitive result set
- Read-Only Table (via the table definition or the operating system security)

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares an SQL statement
// and executes it, and displays the read-only
// status for the result set cursor using the
// edb_readonly() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"SELECT * FROM customer");
$cursor = edb_execute($cmd);

echo edb_readonly($cursor);

edb_disconnect($con);
?>
```

4.10 edb_getreadsize

Returns the row read size for a remote cursor.

Syntax

```
edb_getreadsize(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
Row read size (INTEGER) for the cursor if successful, or
FALSE if there are any errors
```

Usage

The edb_getreadsize function returns the row read size for a cursor whose parent connection is a remote connection to an ElevateDB Server. The row read size determines how many rows are read in one chunk whenever ElevateDB needs to read more rows from an ElevateDB Server. Using a larger read size can result in better performance when navigating a large number of rows using a cursor. The default row read size for any cursor is 1.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets the row read size using
// the edb_setreadsize() function, goes to the start
// of the cursor, and navigates to the end of the cursor
// by using the edb_eof() function to test when to
// stop navigating

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setreadsize($cursor,50);
```

```
edb_first($cursor);

while (!edb_eof($cursor)):
   edb_next($cursor);
endwhile;

echo edb_state($cursor);

edb_disconnect($con);
?>
```

4.11 edb_setreadsize

Sets the row read size for a remote cursor.

Syntax

```
edb_setreadsize(<CursorHandle>, <RowReadSize>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<RowReadSize> = Number of rows to read
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_setreadsize function sets the row read size for a cursor whose parent connection is a remote connection to an ElevateDB Server. The row read size determines how many rows are read in one chunk whenever ElevateDB needs to read more rows from an ElevateDB Server. Using a larger read size can result in better performance when navigating a large number of rows using a cursor. The default row read size for any cursor is 1.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets the row read size using
// the edb_setreadsize() function, goes to the start
// of the cursor, and navigates to the end of the cursor
// by using the edb_eof() function to test when to
// stop navigating

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setreadsize($cursor,50);
```

```
edb_first($cursor);

while (!edb_eof($cursor)):
   edb_next($cursor);
endwhile;

echo edb_state($cursor);

edb_disconnect($con);
?>
```

4.12 edb_filtered

Indicates whether a cursor is filtered or not.

Syntax

```
edb_filtered(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if the cursor is filtered, or FALSE if
not (or there are any errors)
```

Usage

The edb_filtered function returns whether a cursor is filtered or not. A cursor can be filtered using the edb_setfilter function.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets a filter, and displays
// the filter status of the cursor using the
// edb_filtered() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setfilter($cursor,"State='FL'");

echo edb_filtered($cursor);

edb_disconnect($con);
?>
```

4.13 edb_getfilter

Gets the filter for a filtered cursor.

Syntax

```
edb_getfilter(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
Filter (STRING) if successful, or FALSE if there are any errors
```

Usage

The edb_getfilter function returns the filter for a cursor that has been filtered using the edb_setfilter function.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets a filter, and displays
// the filter of the cursor using the edb_getfilter()
// function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setfilter($cursor,"State='FL'");

echo edb_getfilter($cursor);

edb_disconnect($con);
?>
```

4.14 edb_setfilter

Sets a filter on a cursor.

Syntax

```
edb_setfilter(<CursorHandle>, <Filter>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<Filter> = Any valid Boolean SQL expression
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_setfilter function sets a filter on the specified cursor. A filter may be any valid SQL expression that can normally be used in a SELECT statement's WHERE clause, which is any valid SQL expression that evaluates to a Boolean true or false.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

A filter can be cleared by using the edb_clearfilter function.

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, and sets a filter using the
// edb_setfilter() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);
```

```
edb_setfilter($cursor,"State='FL'");

edb_disconnect($con);
?>
```

4.15 edb_clearfilter

Clears a filter on a cursor.

Syntax

```
edb_clearfilter(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_clearfilter function clears a filter on the specified cursor.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets a filter, and then clears
// the filter using the edb_clearfilter() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setfilter($cursor,"State='FL'");
edb_clearfilter($cursor);

echo edb_filtered($cursor);
```

```
edb_disconnect($con);
?>
```

4.16 edb_locate

Locates a row in a cursor.

Syntax

```
edb_locate(<CursorHandle>, <SearchValues>,
          [<CaseInsSearch> [, <PartialSearch>]])

<CursorHandle> =

Handle of cursor returned by edb_execute function

<SearchValues> = Associative array of column names and values

<CaseInsSearch> = Perform a case-insensitive search
(STRING values only)

<PartialSearch> = Perform a partial search on the last
search value (STRING values only)
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_locate function performs a free-form search of a cursor dataset, using the input associative array to determine which columns (and their values) to search on. The case-insensitive parameter will allow for a case-insensitive search (STRING values only), and the partial search parameter will cause the engine to perform a partial-length search on the last provided search value (STRING values only).

If an index exists that matches the columns being searched on, as well as the case-insensitive flag, matched on case-insensitive (CI) flag of indexed column collation, then the index will be used to optimize the search. If no matching index exists, then a scan of all of the cursor rows will take place instead.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, and then performs a case-insensitive,
// partial-length locate using the edb locate()
```

```
// function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

if (edb_locate($cursor,array("State"=>"FL"),FALSE))
{
  echo "Row found";
}
else
{
  echo "Row not found";
}

edb_disconnect($con);
?>
```

4.17 edb_getkey

Gets the primary or unique key's index name for a cursor.

Syntax

```
edb_getkey(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
The primary key or unique key's index name
if successful, or FALSE if there are any errors or
if the cursor does not contain a primary or unique
key
```

Usage

The edb_getkey function gets the primary or unique key's index name for a cursor. This is useful when you need an index that uniquely identifies each row. This function can only be used with cursors on directly-opened tables. Please see the edb_prepare function for more information on opening tables directly.

You can use the edb_setindex to change the active index order for a table cursor.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a direct table
// open, sets the active index to the primary key's index
// and then displays the active index using the edb_getindex()
// function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setindex($cursor,edb_getkey($cursor));
echo edb_getindex($cursor);
```

```
edb_disconnect($con);
?>
```

4.18 edb_getkeycols

Gets all key column values for the current row in a cursor as an associative array.

Syntax

```
edb_getkeycols(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
Associative array of column values if
successful, or FALSE if there are any errors or
if the cursor does not contain a primary or unique
key
```

Usage

The edb_getkeycols function gets the primary or unique key column values for a cursor. This function can only be used with cursors on directly-opened tables. Please see the edb_prepare function for more information on opening tables directly.

You can use the edb_getkey to retrieve the primary or unique key index name for a table cursor.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a direct table
// open, sets the active index, and then displays
// the active index using the edb_getindex() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

var_dump(edb_getkeycols($cursor));

edb_disconnect($con);
```

```
?>
```

4.19 edb_getindex

Gets the active index for a cursor.

Syntax

```
edb_getindex(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
The active index name if successful, or FALSE
if there are any errors
```

Usage

The edb_getindex function gets the active index for a cursor. This function can only be used with cursors on directly-opened tables. Please see the edb_prepare function for more information on opening tables directly.

You can use the edb_setindex to change the active index order for a table cursor.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a direct table
// open, sets the active index, and then displays
// the active index using the edb_getindex() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setindex($cursor,"ByCompany");
echo edb_getindex($cursor);

edb_disconnect($con);
?>
```

4.20 edb_setindex

Sets the active index for a cursor.

Syntax

```
edb_getindex(<CursorHandle>, <IndexName>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<IndexName> = Name of index to make the active index order
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_setindex function sets the active index for a cursor. This function can only be used with cursors on directly-opened tables. Please see the edb_prepare function for more information on opening tables directly.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares a direct table
// open, sets the active index using the edb_setindex()
// function, and then displays the active index

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);
```

```
edb_setindex($cursor,"ByCompany");
echo edb_getindex($cursor);

edb_disconnect($con);
?>
```

4.21 edb_find

Finds a row in a cursor by key, using the active index.

Syntax

```
edb_find(<CursorHandle>, <KeyValues>,
          [<NearSearch>])

<CursorHandle> =

Handle of cursor returned by edb_execute function

<SearchValues> = Simple array of values, corresponding
to the indexed columns in the active index

<NearSearch> = Perform a near search
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_find function performs a key search on the active index of a cursor. This function can only be used with cursors on directly-opened tables. Please see the edb_prepare function for more information on opening tables directly. The near search parameter will cause the search to search for the specified key, and if it doesn't exist, to position on the next highest key in the active index order.

You can use the edb_setindex to change the active index order for a table cursor.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets the active index order,
// and then performs a near find using the edb_find()
// function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
```

```
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setindex($cursor,"ByCompany");

edb_find($cursor,array("Diver"),TRUE);
var_dump(edb_getrow($cursor));

edb_disconnect($con);
?>
```

4.22 edb_ranged

Indicates whether a cursor has an active range or not.

Syntax

```
edb_ranged(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if the cursor has an active range, or
FALSE if not (or there are any errors)
```

Usage

The edb_ranged function returns whether a cursor has an active range or not. You can set a range on a cursor by using the edb_setrange function.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets a range, and displays
// the ranged status of the cursor using the
// edb_ranged() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setrange($cursor,array(1000,3000));

echo edb_ranged($cursor);

edb_disconnect($con);
?>
```

4.23 edb_getrangestart

Gets the starting key values for the active range in a cursor as a simple array.

Syntax

```
edb_getrangestart(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
Simple array of key values if successful, or FALSE
if there are any errors
```

Usage

The edb_getrangestart function retrieves the starting key values for the active range on a cursor.

You can set a range on a cursor by using the edb_setrange function.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets a range, and then
// displays the starting/ending key values for the range
// using the edb_getrangestart() and getrangeend() functions

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setrange($cursor,array(1000,3000));

var_dump(edb_getrangestart($cursor));
var_dump(edb_getrangeend($cursor));

edb_disconnect($con);
```

```
?>
```

4.24 edb_getrangeend

Gets the ending key values for the active range in a cursor as a simple array.

Syntax

```
edb_getrangeend(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
Simple array of key values if successful, or FALSE
if there are any errors
```

Usage

The edb_getrangeend function retrieves the ending key values for the active range on a cursor.

You can set a range on a cursor by using the edb_setrange function.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets a range, and then
// displays the starting/ending key values for the range
// using the edb_getrangestart() and getrangeend() functions

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setrange($cursor,array(1000,3000));

var_dump(edb_getrangestart($cursor));
var_dump(edb_getrangeend($cursor));

edb_disconnect($con);
```

```
?>
```

4.25 edb_setrange

Sets a range on a cursor, using the active index.

Syntax

```
edb_setrange(<CursorHandle>,
             <BeginRangeValues>,
             <EndRangeValues>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<BeginRangeValues> = Simple array of values, corresponding
to the indexed columns in the active index

<EndRangeValues> = Simple array of values, corresponding
to the indexed columns in the active index
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_setrange function sets a range on the specified cursor, using the active index. A range is simply a pair of beginning and ending key values that limit the rows in a cursor to those whose index key values match fall within them (inclusive).

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

A range can be cleared by using the edb_clearrange function.

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets a range using the
// edb_setrange() function, and then displays
// the row count

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
```

```
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setrange($cursor,array(1000,3000));

echo edb_rowcount($cursor);

edb_disconnect($con);
?>
```

4.26 edb_clearrange

Clears a range on a cursor.

Syntax

```
edb_clearrange(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, or FALSE if there are any errors
```

Usage

The edb_clearrange function clears a range on the specified cursor.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, sets a range, and then clears
// the range using the edb_clearrange() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_setrange($cursor,array(1000,3000));
edb_clearrange($cursor);

echo edb_ranged($cursor);
```

```
edb_disconnect($con);
?>
```

4.27 edb_state

Indicates whether the state of a cursor.

Syntax

```
edb_state(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
Cursor state (INTEGER) if successful, or
FALSE there are any errors
```

Usage

The edb_state function returns the state of a cursor. The state will be one of the following values (INTEGER):

State	Description
EDB_BROWSE_STATE (0)	The cursor is in a browse state
EDB_INSERT_STATE (1)	The cursor is in the insert state. Calling the edb_post function will insert the current row into the underlying table, while the edb_cancel function will cancel the insert and return the cursor to the EDB_BROWSE_STATE.
EDB_UPDATE_STATE (2)	The cursor is in the update state. Calling the edb_post function will update the current row in the underlying table, while the edb_cancel function will cancel the update and return the cursor to the EDB_BROWSE_STATE.

The following functions can change the state of a cursor:

edb_setfilter
edb_clearfilter
edb_locate
edb_setindex
edb_find
edb_setrange
edb_clearrange
edb_refresh
edb_insert
edb_update
edb_post
edb_cancel

edb_lock
edb_unlock
edb_unlockall

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, begins an insert, and displays
// the state of the cursor using the edb_state() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_insert($cursor);

echo edb_state($cursor);

edb_disconnect($con);
?>
```

4.28 edb_bof

Indicates whether the current row is positioned at the beginning of the cursor.

Syntax

```
edb_bof(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if the current row is at BOF, or
FALSE if not (or if there are any errors)
```

Usage

The edb_bof function returns the whether the current row is positioned at the beginning of the cursor. The BOF (Beginning Of File) flag indicates that the current row has been positioned at the beginning of the cursor using the edb_first function, or an attempt has been made to navigate past the beginning of the cursor using the edb_prior function.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, goes to the end of the cursor,
// and navigates to the beginning of the cursor
// by using the edb_bof() function to test when to
// stop navigating

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_last($cursor);

while (!edb_bof($cursor)):
   edb_prior($cursor);
```

```
endwhile;

echo edb_state($cursor);

edb_disconnect($con);
?>
```

4.29 edb_eof

Indicates whether the current row is positioned at the end of the cursor.

Syntax

```
edb_eof(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if the current row is at EOF, or
FALSE if not (or if there are any errors)
```

Usage

The edb_eof function returns the whether the current row is positioned at the end of the cursor. The EOF (End Of File) flag indicates that the current row has been positioned at the end of the cursor using the edb_last function, or an attempt has been made to navigate past the end of the cursor using the edb_next function.

Examples

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, goes to the start of the cursor,
// and navigates to the end of the cursor
// by using the edb_eof() function to test when to
// stop navigating

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_first($cursor);

while (!edb_eof($cursor)):
   edb_next($cursor);
```

```
endwhile;

echo edb_state($cursor);

edb_disconnect($con);
?>
```

4.30 edb_refresh

Refreshes the cursor and, if changes are found, the current row.

Syntax

```
edb_refresh(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if any changes are found to the underlying
dataset, or FALSE if there are no changes (or if there are
any errors
```

Usage

The edb_refresh function performs a refresh of the cursor and, if changes have been made to the underlying dataset since the cursor last accessed the underlying dataset, the current row is also refreshed and the function returns TRUE.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, and then performs a refresh
// using the edb_refresh() function, displaying
// the result

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);
```

```
if (edb_refresh($cursor))
{
  echo "Changes were found";
}
else
{
  echo "No changes were found";
}

edb_disconnect($con);
?>
```

4.31 edb_first

Navigates to the first row in the cursor.

Syntax

```
edb_first(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_first function navigates to the first row in a cursor, setting the BOF flag in the process.

> **Warning**
> If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, goes to the start of the cursor,
// and navigates to the end of the cursor
// by using the edb_eof() function to test when to
// stop navigating

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_first($cursor);

while (!edb_eof($cursor)):
```

```
   edb_next($cursor);
endwhile;

echo edb_state($cursor);

edb_disconnect($con);
?>
```

4.32 edb_next

Navigates to the next row in the cursor.

Syntax

```
edb_next(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_next function navigates to the next row in a cursor. If there are no more rows in the cursor, then the EOF flag will be set.

> **Warning**
> If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, goes to the start of the cursor,
// and navigates to the end of the cursor
// by using the edb_eof() function to test when to
// stop navigating

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_first($cursor);
```

```
while (!edb_eof($cursor)):
   edb_next($cursor);
endwhile;

echo edb_state($cursor);

edb_disconnect($con);
?>
```

4.33 edb_prior

Navigates to the prior row in the cursor.

Syntax

```
edb_prior(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_prior function navigates to the prior row in a cursor. If there are no more prior rows in the cursor, then the BOF flag will be set.

> **Warning**
> If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, goes to the end of the cursor,
// and navigates to the beginning of the cursor
// by using the edb_bof() function to test when to
// stop navigating

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_last($cursor);
```

```
while (!edb_bof($cursor)):
   edb_prior($cursor);
endwhile;

echo edb_state($cursor);

edb_disconnect($con);
?>
```

4.34 edb_last

Navigates to the last row in the cursor.

Syntax

```
edb_last(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_last function navigates to the last row in a cursor, setting the EOF flag in the process.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, goes to the end of the cursor,
// and navigates to the beginning of the cursor
// by using the edb_bof() function to test when to
// stop navigating

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_last($cursor);

while (!edb_bof($cursor)):
```

```
   edb_prior($cursor);
endwhile;

echo edb_state($cursor);

edb_disconnect($con);
?>
```

4.35 edb_insert

Puts a cursor into the insert state and initializes the current row with any default values.

Syntax

```
edb_insert(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_insert function puts a cursor into an insert state (EDB_INSERT_STATE), and initializes the current row with any default values defined for the underlying dataset.

> **Warning**
> If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, puts the cursor into an
// insert state using the edb_insert() function,
// sets the row values, and then completes the
// insert

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_insert($cursor);
edb_setcol($cursor,"CustNo",1000);
```

```
edb_setcol($cursor,"Company","My Company");
if (!edb_post($cursor))
  {
  $msg = edb_errmsg();
  die("Error adding customer: " . $msg);
  }

edb_disconnect($con);
?>
```

4.36 edb_update

Puts a cursor into the update state and retrieves the most current version of the current row.

Syntax

```
edb_update(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_update function puts a cursor into an update state (EDB_UPDATE_STATE), and retrieves the most current version of the current row. If the cursor's parent connection is using pessmistic row locking, then the current row is automatically locked at this time. If the cursor's parent connection is using optimistic row locking, then the current row will not be locked until the edb_post function is called. For more information on setting the row lock protocol, please see the Connection Strings topic.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, finds a row, puts the cursor
// into an update state using the edb_update() function,
// sets the row values, and then completes the update

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);
```

```
if (edb_find($cursor,array(1000)))
{
  edb_update($cursor);
  edb_setcol($cursor,"Company","My Company 2 - Electric Boogaloo");
  if (!edb_post($cursor))
    {
    $msg = edb_errmsg();
    die("Error updating customer: " . $msg);
    }
}

edb_disconnect($con);
?>
```

4.37 edb_post

Posts an active insert or update for a cursor, completing the insert or update.

Syntax

```
edb_post(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_post function posts an active insert or update for a cursor. If there are no errors, such as constraint errors, then the insert or update is considered complete, any pessimistic row locks are released, and the cursor is returned to the browse state (EDB_BROWSE_STATE). If the cursor's parent connection is using optimistic row locking, then the current row will not be locked until this function executes.

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, puts the cursor into an
// insert state, sets the row values, and then
// completes the insert with the edb_post() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_insert($cursor);
edb_setcol($cursor,"CustNo",1000);
edb_setcol($cursor,"Company","My Company");
if (!edb_post($cursor))
  {
  $msg = edb_errmsg();
  die("Error adding customer: " . $msg);
  }
```

```
edb_disconnect($con);
?>
```

4.38 edb_cancel

Cancels an active insert or update for a cursor, abandoning the insert or update.

Syntax

```
edb_cancel(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_cancel function cancels an active insert or update for a cursor. Any pessimistic row locks are released, and the cursor is returned to the browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, puts the cursor into an
// insert state, sets the row values, and then
// cancels the insert using the edb_cancel() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_insert($cursor);
edb_setcol($cursor,"CustNo",1000);
edb_setcol($cursor,"Company","My Company");
edb_cancel($cursor);

edb_disconnect($con);
?>
```

4.39 edb_delete

Deletes the current row in a cursor.

Syntax

```
edb_delete(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_delete function deletes the current row in a cursor.

> **Warning**
> If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, finds a row, and then deletes
// the row using the edb_delete() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

if (edb_find($cursor,array(1000)))
{
  if (!edb_delete($cursor))
    {
    $msg = edb errmsg();
```

```
    die("Error deleting customer: " . $msg);
    }
}

edb_disconnect($con);
?>
```

4.40 edb_lock

Manually locks the current row in a cursor.

Syntax

```
edb_lock(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_lock function manually locks the current row in a cursor. Manual locks are different from automatic locks in the following ways:

- Automatic locks are acquired (automatically) by the engine when calling the edb_insert, edb_update, and edb_delete functions, while manual locks must be handled by the developer via the edb_lock, edb_unlock, and edb_unlockall functions.
- Manual locks supercede automatic locks, and have a longer lifespan than automatic locks. For example, if you use the edb_lock function to acquire a manual row lock on the current row, and then proceed to update the current row. The manual row lock will still exist after the update is complete. Manual row locks exist until they are manually unlocked with the edb_unlock and edb_unlockall functions, the locked rows are deleted with the edb_delete function, or the cursor's parent command is released with the edb_unprepare function.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, locks the current row in the
// cursor using the edb_lock() function, updates
// the current row, and then unlocks the current row.

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
```

```
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

if (edb_find($cursor,array(1000)))
{
  if (edb_lock($cursor))
    {
    edb_update($cursor);
    edb_setcol($cursor,"Company","My Company 2 - Electric Boogaloo");
    if (!edb_post($cursor))
      {
      $msg = edb_errmsg();
      die("Error updating customer: " . $msg);
      }
    edb_unlock($cursor);
    }
}

edb_disconnect($con);
?>
```

4.41 edb_unlock

Unlocks the current manually-locked row in a cursor.

Syntax

```
edb_unlock(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_unlock function unlocks the current manually-locked row in a cursor. Please see the edb_lock function for more information on manual locks.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, locks the current row in the
// cursor, updates the current row, and then unlocks
// the current row in the cursor using the edb_unlock()
// function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

if (edb_find($cursor,array(1000)))
{
```

```
  if (edb_lock($cursor))
    {
    edb_update($cursor);
    edb_setcol($cursor,"Company","My Company 2 - Electric Boogaloo");
    if (!edb_post($cursor))
      {
      $msg = edb_errmsg();
      die("Error updating customer: " . $msg);
      }
    edb_unlock($cursor);
    }
}

edb_disconnect($con);
?>
```

4.42 edb_unlockall

Unlocks all manually-locked rows in a cursor.

Syntax

```
edb_unlockall(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_unlockall function unlocks all manually-locked rows in a cursor. Please see the edb_lock function for more information on manual locks.

Warning
If the cursor state, retrievable via the edb_state function, is an insert state (EDB_INSERT_STATE) or update state (EDB_UPDATE_STATE), then this function will cause an automatic call to the edb_post function in order to force the cursor into a browse state (EDB_BROWSE_STATE).

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, locks the first two rows
// in the cursor, and then unlocks them all using
// the edb_unlockall() function

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_lock($cursor);
edb_next($cursor);
edb_lock($cursor);
```

```
edb_unlockall($cursor);

edb_disconnect($con);
?>
```

4.43 edb_jsoncols

Generates the JSON for the column definitions of an Elevate Web Builder dataset from a cursor.

Syntax

```
edb_jsoncols(<CursorHandle>)

<CursorHandle> =

Handle of cursor returned by edb_execute function
```

Returns

```
The column definitions as a JSON string
if successful, and FALSE if there are any errors
```

Usage

The edb_jsoncols function generates the column definitions for a cursor in Elevate Web Builder JSON format.

Please see the Elevate Web Builder manual for more information on datasets and the JSON data formats used with them.

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, and then handles the incoming
// Elevate Web Builder dataset requests for the
// column definitions, rows, or BLOB data
//
// THIS IS JUST FOR ILLUSTRATIVE PURPOSES
//
// If you really need to use the ElevateDB PHP
// Extension with Elevate Web Builder, please see
// the Elevate Web Builder PHP DataSet Manager
// included with Elevate Web Builder.  It requires
// much less coding, and handles the dataset JSON
// transparently

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
```

```
$cursor = edb_execute($cmd);

if (isset($_SERVER["HTTP_X_EWBUSER"]) and
      isset($_SERVER["HTTP_X_EWBPASSWORD"]))
   {
   $user = $_SERVER["HTTP_X_EWBUSER"];
   $pwd = $_SERVER["HTTP_X_EWBPASSWORD"];
   }
else if (isset($_GET["user"]) and isset($_GET["password"]))
   {
   $user = $_GET["user"];
   $pwd = $_GET["password"];
   }
else
   {
   $user = NULL;
   $pwd = NULL;
   }

if ($_GET["method"])
   {
   switch ($_GET["method"])
      {
      case "columns":
         {
         $cols = edb_jsoncols($cursor);
         if ($cols)
            {
            header("Content-Type: application/json; charset=utf-8");
            echo $cols;
            }
         else
            {
            die "Error retrieving dataset columns (" . edb_errmsg() . ")";
            }
         break;
         }
      case "rows":
         {
         $rows = edb_jsonrows($cursor,$_SERVER["SCRIPT_NAME"],"customer",
      FALSE,$_GET,$user,$pwd);
         if ($rows)
            {
            header("Content-Type: application/json; charset=utf-8");
            echo $rows;
            }
         else
            {
            die "Error retrieving dataset rows (" . edb_errmsg() . ")";
            }
         break;
         }
      case "load":
         {
         $data = edb_jsonload($cursor,$_GET["column"],$_GET["row"]);
         $content_type = edb_jsonloadtype($cursor,$_GET["column"],
      $_GET["row"]);
         if ($content_type <> "")
            {
```

```
            header("Content-Type: " . $content_type);
            }
          echo $data;
          break;
          }
       }
  }
else
  {
  die "Dataset method not specified";
  }

edb_disconnect($con);
?>
```

4.44 edb_jsonrows

Generates the JSON for the rows of an Elevate Web Builder dataset from a cursor.

Syntax

```
edb_jsonrows(<CursorHandle>, <ResourceName>,
                     <DataSetName>, [<LocalizedDateTimeColumns>],
                     [<DataSetParams>], [<UserName>, <Password>])

<CursorHandle> =

Handle of cursor returned by edb_execute function

<ResourceName> =

String containing resource name to use for embedded BLOB URLs

<DataSetName> =

String containing dataset name to use for embedded BLOB URLs

<LocalizedDateTimeColumns> =

Boolean indicating whether to convert date/time columns to UTC time

<DataSetParams> =

String containing ampersand-delimited (&) list of dataset parameters to use
      for embedded BLOB URLs

<UserName> =

String containing user name to use for embedded BLOB URLs

<Password> =

String containing password to use for embedded BLOB URLs
```

Returns

```
The cursor rows as a JSON string
if successful, and FALSE if there are any errors
```

Usage

The edb_jsonrows function generates the rows for a cursor in Elevate Web Builder JSON format. The ResourceName parameter is used to indicate the base URL to use for any BLOB URLs in the JSON rows. Embedded URLs are used in Elevate Web Builder to load BLOB data on demand. The DataSetName parameter is used to indicate the dataset name to use for any BLOB URLs in the JSON rows. The dataset name is passed as a key-value parameter:

```
dataset=<DataSetName>
```

The complete URL used for embedded BLOB URLs would then be:

```
<ResourceName>load?dataset=<DataSetName>&col=<ColumnName>&row=<RowKey>
```

The LocalizeDateTimeColumns parameter indicates whether date/time columns should be converted from local time to UTC time after being retrieved from the cursor. The default value for this parameter is False.

The DataSetParams parameter is an ampersand-delimited (&) of URL parameters to pass through with the embedded BLOB URLs. This parameter is important if the dataset is a query that requires parameters for filtering.

The UserName and Password parameters are used for authentication with any embedded BLOB URLs. They are passed back to the web server as authentication parameters when the BLOB URLs are used by the Elevate Web Builder application to load the BLOB data. You **must** specify both parameters, otherwise the UserName parameter will be ignored.

Warning
This function moves the current row pointer in the cursor.

Please see the Elevate Web Builder manual for more information on datasets and the JSON data formats used with them.

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, and then handles the incoming
// Elevate Web Builder dataset requests for the
// column definitions, rows, or BLOB data
//
// THIS IS JUST FOR ILLUSTRATIVE PURPOSES
//
// If you really need to use the ElevateDB PHP
// Extension with Elevate Web Builder, please see
// the Elevate Web Builder PHP DataSet Manager
// included with Elevate Web Builder.  It requires
// much less coding, and handles the dataset JSON
// transparently

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }
```

```
$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);

$cursor = edb_execute($cmd);

if (isset($_SERVER["HTTP_X_EWBUSER"]) and
      isset($_SERVER["HTTP_X_EWBPASSWORD"]))
   {
   $user = $_SERVER["HTTP_X_EWBUSER"];
   $pwd = $_SERVER["HTTP_X_EWBPASSWORD"];
   }
else if (isset($_GET["user"]) and isset($_GET["password"]))
   {
   $user = $_GET["user"];
   $pwd = $_GET["password"];
   }
else
   {
   $user = NULL;
   $pwd = NULL;
   }

if ($_GET["method"])
   {
   switch ($_GET["method"])
      {
      case "columns":
         {
         $cols = edb_jsoncols($cursor);
         if ($cols)
            {
            header("Content-Type: application/json; charset=utf-8");
            echo $cols;
            }
         else
            {
            die "Error retrieving dataset columns (" . edb_errmsg() . ")";
            }
         break;
         }
      case "rows":
         {
         $rows = edb_jsonrows($cursor,$_SERVER["SCRIPT_NAME"],"customer",
      FALSE,$_GET,$user,$pwd);
         if ($rows)
            {
            header("Content-Type: application/json; charset=utf-8");
            echo $rows;
            }
         else
            {
            die "Error retrieving dataset rows (" . edb_errmsg() . ")";
            }
         break;
         }
      case "load":
         {
         $data = edb_jsonload($cursor,$_GET["column"],$_GET["row"]);
         $content_type = edb_jsonloadtype($cursor,$_GET["column"],
      $_GET["row"]);
         if ($content type <> "")
```

```
            {
            header("Content-Type: " . $content_type);
            }
         echo $data;
         break;
         }
      }
  }
else
  {
  die "Dataset method not specified";
  }

edb_disconnect($con);
?>
```

4.45 edb_jsonload

Returns the data for a BLOB column in an Elevate Web Builder dataset from a cursor.

Syntax

```
edb_jsonload(<CursorHandle>, <ColumnName>,
                     <RowKeyValues>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<ColumnName> =

Name of BLOB column from which to retrieve data

<RowKeyValues> =

String containing a semicolon-delimited list of key values
```

Returns

```
The BLOB column data as a string
if successful, and FALSE if there are any errors
```

Usage

The edb_jsonload function generates the BLOB column data for a given row in a cursor.

Warning
This function moves the current row pointer in the cursor.

Please see the Elevate Web Builder manual for more information on datasets and the JSON data formats used with them.

```
<?php

// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, and then handles the incoming
// Elevate Web Builder dataset requests for the
// column definitions, rows, or BLOB data
//
// THIS IS JUST FOR ILLUSTRATIVE PURPOSES
//
// If you really need to use the ElevateDB PHP
```

```
// Extension with Elevate Web Builder, please see
// the Elevate Web Builder PHP DataSet Manager
// included with Elevate Web Builder.  It requires
// much less coding, and handles the dataset JSON
// transparently

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);

$cursor = edb_execute($cmd);

if (isset($_SERVER["HTTP_X_EWBUSER"]) and
      isset($_SERVER["HTTP_X_EWBPASSWORD"]))
   {
   $user = $_SERVER["HTTP_X_EWBUSER"];
   $pwd = $_SERVER["HTTP_X_EWBPASSWORD"];
   }
else if (isset($_GET["user"]) and isset($_GET["password"]))
   {
   $user = $_GET["user"];
   $pwd = $_GET["password"];
   }
else
   {
   $user = NULL;
   $pwd = NULL;
   }

if ($_GET["method"])
   {
   switch ($_GET["method"])
      {
      case "columns":
         {
         $cols = edb_jsoncols($cursor);
         if ($cols)
            {
            header("Content-Type: application/json; charset=utf-8");
            echo $cols;
            }
         else
            {
            die "Error retrieving dataset columns (" . edb_errmsg() . ")";
            }
         break;
         }
      case "rows":
         {
         $rows = edb_jsonrows($cursor,$_SERVER["SCRIPT_NAME"],"customer",
      FALSE,$_GET,$user,$pwd);
         if ($rows)
            {
            header("Content-Type: application/json; charset=utf-8");
            echo $rows;
```

```
            }
         else
            {
            die "Error retrieving dataset rows (" . edb_errmsg() . ")";
            }
         break;
         }
      case "load":
         {
         $data = edb_jsonload($cursor,$_GET["column"],$_GET["row"]);
         $content_type = edb_jsonloadtype($cursor,$_GET["column"],
      $_GET["row"]);
         if ($content_type <> "")
            {
            header("Content-Type: " . $content_type);
            }
         echo $data;
         break;
         }
      }
   }
else
   {
   die "Dataset method not specified";
   }

edb_disconnect($con);
?>
```

4.46 edb_jsonloadtype

Returns the MIME type for a BLOB column in an Elevate Web Builder dataset from a cursor.

Syntax

```
edb_jsonloadtype(<CursorHandle>, <ColumnName>,
                         <RowKeyValues>)

<CursorHandle> =

Handle of cursor returned by edb_execute function

<ColumnName> =

Name of BLOB column to retrieve content type for

<RowKeyValues> =

String containing a semicolon-delimited list of key values
```

Returns

```
The BLOB column MIME type as a string
if successful, and FALSE if there are any errors
```

Usage

The edb_jsonloadtype function returns the MIME type BLOB column data for a given row in a cursor. The MIME type for a BLOB column is determined by the ElevateDB PHP Extension by looking for a column with a name of:

```
<ColumnName>_ContentType
```

If a column exists with the given name, then the value contained within the column for the specified row is returned as the MIME type for the BLOB column.

Warning
This function moves the current row pointer in the cursor.

Please see the Elevate Web Builder manual for more information on datasets and the JSON data formats used with them.

```
<?php
```

```
// The following script connects to an ElevateDB
// Server and database, prepares and executes a
// direct table open, and then handles the incoming
// Elevate Web Builder dataset requests for the
// column definitions, rows, or BLOB data
//
// THIS IS JUST FOR ILLUSTRATIVE PURPOSES
//
// If you really need to use the ElevateDB PHP
// Extension with Elevate Web Builder, please see
// the Elevate Web Builder PHP DataSet Manager
// included with Elevate Web Builder.  It requires
// much less coding, and handles the dataset JSON
// transparently

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);

$cursor = edb_execute($cmd);

if (isset($_SERVER["HTTP_X_EWBUSER"]) and
      isset($_SERVER["HTTP_X_EWBPASSWORD"]))
   {
   $user = $_SERVER["HTTP_X_EWBUSER"];
   $pwd = $_SERVER["HTTP_X_EWBPASSWORD"];
   }
else if (isset($_GET["user"]) and isset($_GET["password"]))
   {
   $user = $_GET["user"];
   $pwd = $_GET["password"];
   }
else
   {
   $user = NULL;
   $pwd = NULL;
   }

if ($_GET["method"])
   {
   switch ($_GET["method"])
      {
      case "columns":
         {
         $cols = edb_jsoncols($cursor);
         if ($cols)
            {
            header("Content-Type: application/json; charset=utf-8");
            echo $cols;
            }
         else
            {
            die "Error retrieving dataset columns (" . edb_errmsg() . ")";
            }
```

```
         break;
         }
      case "rows":
         {
         $rows = edb_jsonrows($cursor,$_SERVER["SCRIPT_NAME"],"customer",
      FALSE,$_GET,$user,$pwd);
         if ($rows)
            {
            header("Content-Type: application/json; charset=utf-8");
            echo $rows;
            }
         else
            {
            die "Error retrieving dataset rows (" . edb_errmsg() . ")";
            }
         break;
         }
      case "load":
         {
         $data = edb_jsonload($cursor,$_GET["column"],$_GET["row"]);
         $content_type = edb_jsonloadtype($cursor,$_GET["column"],
      $_GET["row"]);
         if ($content_type <> "")
            {
            header("Content-Type: " . $content_type);
            }
         echo $data;
         break;
         }
      }
  }
else
  {
  die "Dataset method not specified";
  }

edb_disconnect($con);
?>
```

4.47 edb_jsoninsert

Inserts a row into a cursor using Elevate Web Builder dataset JSON row values.

Syntax

```
edb_jsoninsert(<CursorHandle>, <RowValues>,
                       [<LocalizedDateTimeColumns>])

<CursorHandle> =

Handle of cursor returned by edb_execute function

<RowValues> =

Associative array of column names and values

<LocalizedDateTimeColumns> =

Boolean indicating whether to convert date/time columns to local time
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_jsoninsert function inserts a row into the specified cursor using the provided associative array of column names and values. The column values are expected to be in Elevate Web Builder format, meaning that data types like dates, times, or date/times (timestamps) will be expressed as integer Javascript timestamp values.

Warning
This function moves the current row pointer in the cursor.

Please see the Elevate Web Builder manual for more information on datasets and the JSON data formats used with them.

Note
This function is used exclusively by the Elevate Web Builder PHP DataSet Manager included with Elevate Web Builder for inserting rows using incoming JSON transaction data. Although it can be used outside of this context, it is made available primarily for use in this manner. The benefit of using this function is that it transparently handles date/time conversions (and all other types of data type conversions) properly between JSON and ElevateDB columns.

4.48 edb_jsonupdate

Updates a row in a cursor using Elevate Web Builder dataset JSON before and after row values.

Syntax

```
edb_jsonupdate(<CursorHandle>, <BeforeRowValues>, <AfterRowValues>,
                       [<LocalizedDateTimeColumns>])

<CursorHandle> =

Handle of cursor returned by edb_execute function

<BeforeRowValues>
<AfterRowValues> =

Associative array of column names and values

<LocalizedDateTimeColumns> =

Boolean indicating whether to convert date/time columns to local time
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_jsonupdate function updates an existing row in the specified cursor using the provided associative arrays of column names and values. The column values are expected to be in Elevate Web Builder format, meaning that data types like dates, times, or date/times (timestamps) will be expressed as integer Javascript timestamp values.

Warning
This function moves the current row pointer in the cursor.

Please see the Elevate Web Builder manual for more information on datasets and the JSON data formats used with them.

Note
This function is used exclusively by the Elevate Web Builder PHP DataSet Manager included with Elevate Web Builder for updating rows using incoming JSON transaction data. Although it can be used outside of this context, it is made available primarily for use in this manner. The benefit of using this function is that it transparently handles date/time conversions (and all other types of data type conversions) properly between JSON and ElevateDB columns.

4.49 edb_jsondelete

Deletes a row from a cursor using Elevate Web Builder dataset JSON row values.

Syntax

```
edb_jsondelete(<CursorHandle>, <RowValues>,
                       [<LocalizedDateTimeColumns>])

<CursorHandle> =

Handle of cursor returned by edb_execute function

<RowValues> =

Associative array of column names and values

<LocalizedDateTimeColumns> =

Boolean indicating whether to convert date/time columns to local time
```

Returns

```
TRUE if successful, and FALSE if there are any errors
```

Usage

The edb_jsondelete function deletes a row from the specified cursor using the provided associative array of column names and values. The column values are expected to be in Elevate Web Builder format, meaning that data types like dates, times, or date/times (timestamps) will be expressed as integer Javascript timestamp values.

Warning
This function moves the current row pointer in the cursor.

Please see the Elevate Web Builder manual for more information on datasets and the JSON data formats used with them.

Note
This function is used exclusively by the Elevate Web Builder PHP DataSet Manager included with Elevate Web Builder for deleting rows using incoming JSON transaction data. Although it can be used outside of this context, it is made available primarily for use in this manner. The benefit of using this function is that it transparently handles date/time conversions (and all other types of data type conversions) properly between JSON and ElevateDB columns.

Chapter 5
Error Handling Functions

5.1 Introduction

The ElevateDB PHP Extension includes two error handling functions that provide functionality for retrieving error codes and messages after failures during API calls.

Notation

The notation used in the syntax section for each function is as follows:

Notation	Description
<Element>	Specifies an element of the statement that may be expanded upon further on in the syntax section
<Element> =	Describes an element specified earlier in the syntax section
[Optional Element]	Describes an optional element by enclosing it in square brackets []
Element|Element	Describes multiple elements, of which one and only one may be used in the syntax

5.2 edb_errcode

Returns the error code for the last API call.

Syntax

```
edb_errcode()
```

Returns

```
Last error code (INTEGER) if successful, and FALSE if
there are any errors
```

Usage

The edb_errcode function returns the error code for the last call to any of the edb_* functions present in the ElevateDB PHP Extension API. For a list of all ElevateDB error codes and messages, please see Appendix A - Error Codes and Messages.

Note
This function only returns the error code for the last API call. Any subsequent calls to edb_* functions will clear any error codes/messages.

```
<?php

// The following script connects to an ElevateDB
// Server and database, returning any error code/message
// if there was an error connecting

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: (" . edb_errcode() . ") " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_disconnect($con);
?>
```

5.3 edb_errmsg

Returns the error message for the last API call.

Syntax

```
edb_errmsg()
```

Returns

```
Last error message (STRING) if successful, and FALSE if
there are any errors
```

Usage

The edb_errmsg function returns the error message for the last call to any of the edb_* functions present in the ElevateDB PHP Extension API. For a list of all ElevateDB error codes and messages, please see Appendix A - Error Codes and Messages.

Note
This function only returns the error message for the last API call. Any subsequent calls to edb_* functions will clear any error codes/messages.

```
<?php

// The following script connects to an ElevateDB
// Server and database, returning any error code/message
// if there was an error connecting

$con = edb_connect("type=remote;charset=Ansi;address=127.0.0.1;"+
                   "uid=Administrator;pwd=EDBDefault;database=Test");
if (!$con)
  {
  die("Could not connect: (" . edb_errcode() . ") " . edb_errmsg());
  }

$cmd = edb_prepare($con,"customer",EDB_COMMAND_TABLE);
$cursor = edb_execute($cmd);

edb_disconnect($con);
?>
```

This page intentionally left blank

Appendix A - Error Codes and Messages

The following is a table of the error codes and messages for ElevateDB.

Error Code	Message and Further Details
EDB_ERROR_VALIDATE (100)	There is an error in the metadata for the <ObjectType> <ObjectName> (<ErrorMessage>)This error is raised whenever an attempt is made to create a new catalog or configuration object, and there is an error in the specification of the object. The specific error message is indicated within the parentheses.
EDB_ERROR_UPDATE (101)	There was an error updating the <ObjectType> <ObjectName> (<ErrorMessage>)This error is raised whenever ElevateDB encounters an issue while trying to update the disk file used to store a catalog or configuration. The specific error message is indicated within the parentheses.
EDB_ERROR_SYSTEM (200)	This operation cannot be performed on the system <ObjectType> <ObjectName> or any privileges granted to itThis error is raised whenever an attempt is made to alter or drop any system-defined catalog or configuration objects. Please see the System Information topic for more information on the system-defined objects in ElevateDB.
EDB_ERROR_DEPENDENCY (201)	The <ObjectType> <ObjectName> cannot be dropped or moved because it is still referenced by the <ObjectType> <ObjectName>This error is raised whenever an attempt is made to drop any catalog or configuration object, and that catalog or configuration object is still being referenced by another catalog or configuration object. You must first remove the reference to the object that you wish to drop before you can drop the referenced object.
EDB_ERROR_MODULE (202)	An error occurred with the module <ModuleName> (<ErrorMessage>)This error is raised whenever ElevateDB encounters an issue with loading an external module. Please see the External Modules topic for more information.
EDB_ERROR_LOCK (300)	Cannot lock <ObjectType> <ObjectName> for <AccessType> accessThis error is raised whenever ElevateDB cannot obtain the desired lock access to a given catalog or configuration object. This is usually due to another session already having an incompatible lock on the object already. Please see the Locking and Concurrency topic for more information.
EDB_ERROR_UNLOCK (301)	Cannot unlock <ObjectType> <ObjectName> for <AccessType> accessThis error is raised whenever ElevateDB cannot unlock a given catalog or configuration object. If this error occurs during normal operation of ElevateDB, please contact Elevate Software for further

	instructions on how to correct the issue
EDB_ERROR_EXISTS (400)	The <ObjectType> <ObjectName> already existsThis error is raised whenever an attempt is made to create a new catalog or configuration object, and a catalog or configuration object already exists with that name.
EDB_ERROR_NOTFOUND (401)	The <ObjectType> <ObjectName> does not existThis error is raised when an attempt is made to open/execute, alter, or drop a catalog or configuration object that does not exist.
EDB_ERROR_NOTOPEN (402)	The database <DatabaseName> must be open in order to perform this operation (<OperationName>)This error is raised when an attempt is made to perform an operation on a given database before it has been opened.
EDB_ERROR_READONLY (403)	The <ObjectType> <ObjectName> is read-only and this operation cannot be performed (<OperationName>)This error is raised whenever a create, alter, or drop operation is attempted on an object that is read-only.
EDB_ERROR_TRANS (404)	This operation cannot be performed while the database <DatabaseName> has an active transaction (<OperationName>)This error is raised whenever ElevateDB encounters an invalid transaction operation. Some SQL statements cannot be executed within a transaction. For a list of transaction-compatible statements, please see the Transactions topic.
EDB_ERROR_MAXIMUM (405)	The maximum number of <ObjectType>s has been reached (<MaximumObjectsAllowed>)This error is raised when an attempt is made to create a new catalog or configuration object and doing so would exceed the maximum allowable number of objects. Please see the Appendix B - System Capacities topic for more information.
EDB_ERROR_IDENTIFIER (406)	Invalid <ObjectType> identifier '<ObjectName>'This error is raised when an attempt is made to create a new catalog or configuration object with an invalid name. Please see the Identifiers topic for more information on what constitutes a valid identifier.
EDB_ERROR_FULL (407)	The table <TableName> is full (<FileName>)This error occurs when a given table contains the maximum number of rows or the maximum file size is reached for one of the files that make up the table. The file name is indicated within the parentheses.
EDB_ERROR_CONFIG (409)	There is an error in the configuration (<ErrorMessage>)This error is raised whenever there is an error in the configuration. The specific error message is indicated within the parentheses.

EDB_ERROR_NOLOGIN (500)	A user must be logged in in order to perform this operation (<OperationName>)This error is raised whenever an attempt is made to perform an operation for a session that has not been logged in yet with a valid user name and password.
EDB_ERROR_LOGIN (501)	Login failed (<ErrorMessage>)This error is raised whenever a user login fails. ElevateDB allows for a maximum of 3 login attempts before raising a login exception.
EDB_ERROR_ADMIN (502)	Administrator privileges are required to perform this operation (<Operation>)This error is raised when an attempt is made to perform an operation that requires administrator privileges. Administrator privileges are granted to a given user by granting the system-defined "Administrators" role to that user. Please see the User Security topic for more information.
EDB_ERROR_PRIVILEGE (503)	The current user does not have the proper privileges to perform this operation (<OperationName>)This error is raised when a user attempts an operation when he/she does not have the proper privileges required to execute the operation. Please see the User Security topic for more information.
EDB_ERROR_MAXSESSIONS (504)	Maximum number of concurrent sessions reached for the configuration <ConfigurationName>This error is raised when the maximum number of licensed sessions for a given configuration is exceeded. The number of licensed sessions for a given configuration depends upon the ElevateDB product purchased along with the particular compilation of the application made by the developer using the ElevateDB product.
EDB_ERROR_SERVER (505)	The ElevateDB Server cannot be started (<ErrorMessage>) The ElevateDB Server cannot be stopped (<ErrorMessage>)This error is raised when the ElevateDB Server cannot be started or stopped for any reason. Normally, the error message will contain a native operating system error message that will reveal the reason for the issue.
EDB_ERROR_FILEMANAGER (600)	File manager error (<ErrorMessage>)This error is raised whenever ElevateDB encounters a file manager error while trying to create, open, close, delete, or rename a file. The specific error message, including operating system error code (if available), is indicated within the parentheses.
EDB_ERROR_CORRUPT (601)	The table <TableName> is corrupt (<ErrorMessage>)This error is raised when ElevateDB encounters an issue while reading, writing, or validating a table. If this error occurs during normal operation of ElevateDB, please contact Elevate Software for further instructions on how to correct the issue. The specific error message is indicated within the parentheses.

EDB_ERROR_COMPILE (700)	An error was found in the <ObjectType> at line <Line> and column <Column> (<ErrorMessage>)This error is raised whenever an error is encountered while compiling an SQL expression, statement, or routine. The specific error message is indicated within the parentheses.
EDB_ERROR_BINDING (800)	A row binding error occurredThis error is raised when ElevateDB encounters an issue while trying to bind the cursor row values in a cursor row. It is an internal error and will not occur unless there is a bug in ElevateDB.
EDB_ERROR_STATEMENT (900)	An error occurred with the statement <StatementName> (<ErrorMessage>)This error is raised whenever an issue is encountered while executing a statement. The specific error message is indicated within the parentheses.
EDB_ERROR_PROCEDURE (901)	An error occurred with the procedure <ProcedureName> (<ErrorMessage>)This error is raised whenever an issue is encountered while executing a procedure. The specific error message is indicated within the parentheses.
EDB_ERROR_VIEW (902)	An error occurred with the view <ViewName> (<ErrorMessage>)This error is raised whenever an issue is encountered while opening a view. The specific error message is indicated within the parentheses.
EDB_ERROR_JOB (903)	An error occurred with the job <JobName> (<ErrorMessage>)This error is raised whenever an issue is encountered while running a job. The specific error message is indicated within the parentheses.
EDB_ERROR_IMPORT (904)	Error importing the file <FileName> into the table <TableName> (<ErrorMessage>)This error is raised when an error occurs during the import process for a given table. The specific error message is indicated within the parentheses.
EDB_ERROR_EXPORT (905)	Error exporting the table <TableName> to the file <FileName> (<ErrorMessage>)This error is raised when an error occurs during the export process for a given table. The specific error message is indicated within the parentheses.
EDB_ERROR_CURSOR (1000)	An error occurred with the cursor <CursorName> (<ErrorMessage>)This error is raised whenever an issue is encountered while operating on a cursor. The specific error message is indicated within the parentheses.
EDB_ERROR_FILTER (1001)	A filter error occurred (<ErrorMessage>)This error is raised whenever ElevateDB encounters an issue while trying to set or clear a filter on a given cursor. The specific error message is indicated within the parentheses.
EDB_ERROR_LOCATE (1002)	A locate error occurred (<ErrorMessage>)This error is raised whenever ElevateDB encounters an issue while trying to locate a row in a given cursor. The specific error message is indicated within the parentheses.

EDB_ERROR_STREAM (1003)	An error occurred in the cursor stream (<ErrorMessage>)This error is raised whenever an issue is encountered while loading or saving a cursor to or from a stream. The specific error message is indicated within the parentheses.
EDB_ERROR_CONSTRAINT (1004)	The constraint <ConstrainName> has been violated (<ErrorMessage>)This error is raised when a constraint that has been defined for a table is violated. This includes primary key, unique key, foreign key, and check constraints. The specific error message is indicated within the parentheses.
EDB_ERROR_LOCKROW (1005)	Cannot lock the row in the table <TableName>This error is raised when a request is made to lock a given row and the request fails because another session has the row already locked. Please see the Locking and Concurrency topic for more information.
EDB_ERROR_UNLOCKROW (1006)	Cannot unlock the row in the table <TableName>This error is raised whenever ElevateDB cannot unlock a specific row because the row had not been previously locked, or had been locked and the lock has since been cleared. Please see the Locking and Concurrency topic for more information.
EDB_ERROR_ROWDELETED (1007)	The row has been deleted since last cached for the table <TableName>This error is raised whenever an attempt is made to update or delete a row, and the row no longer exists because it has been deleted by another session. Please see the Updating Rows topic for more information.
EDB_ERROR_ROWMODIFIED (1008)	The row has been modified since last cached for the table <TableName>This error is raised whenever an attempt is made to update or delete a row, and the row has been updated by another session since the last time it was cached by the current session. Please see the Updating Rows topic for more information.
EDB_ERROR_CONSTRAINED (1009)	The cursor is constrained and this row violates the current cursor constraint condition(s)This error is raised when an attempt is made to insert a new row into a constrained cursor that violates the filter constraints defined for the cursor. Both views defined in database catalogs and the result sets of dynamic queries can be defined as constrained, and the filter constraints in both cases are the WHERE conditions defined for the underlying SELECT query that the view or dynamic query is based upon.
EDB_ERROR_ROWVISIBILITY (1010)	The row is no longer visible in the table <TableName>This error is raised whenever an attempt is made to update or delete a row within the context of a cursor with an active filter or range condition, and the row has been updated by another session since the last time it was cached by the current session, thus causing it to fall out of the scope of the cursor's active filter or range condition. Please see the Updating Rows topic for

	more information.
EDB_ERROR_VALUE (1011)	An error occurred with the <ObjectType> <ObjectName> (<ErrorMessage>)This error is raised whenever an attempt is made to store a value in a column, parameter, or variable and the value is invalid because it is out of range or would be truncated. The specific error message is indicated within the parentheses.
EDB_ERROR_CLIENTCONN (1100)	A connection to the server at <ServerAddress> cannot be established (<ErrorMessage>)This error is raised when ElevateDB encounters an issue while trying to connect to a remote ElevateDB Server. The error message will indicate the reason why the connection cannot be completed.
EDB_ERROR_CLIENTLOST (1101)	A connection to the server at <ServerAddress> has been lost <ErrorMessage>)This error is raised when ElevateDB encounters an issue while connected to a remote ElevateDB Server. The error message will indicate the reason why the connection was lost.
EDB_ERROR_INVREQUEST (1103)	An invalid or unknown request was sent to the serverThis error is raised when an ElevateDB Server encounters an unknown request from a client session.
EDB_ERROR_ADDRBLOCK (1104)	The IP address <IPAddress> is blockedThis error is raised when a session tries to connect to an ElevateDB Server, and the originating IP address for the session matches one of the configured blocked IP addresses in the ElevateDB Server, or does not match one of the configured authorized IP addresses in the ElevateDB Server.
EDB_ERROR_ENCRYPTREQ (1105)	An encrypted connection is requiredThis error is raised when a non-encrypted session tries to connect to an ElevateDB Server that has been configured to only accept encrypted session connections.
EDB_ERROR_SESSIONNOTFOUND (1107)	The session ID <SessionID> is no longer present on the serverThis error is raised whenever a remote session attempts to reconnect to a session that has already been designated as a dead session and removed by the ElevateDB Server. This can occur when a session is inactive for a long period of time, or when the ElevateDB Server has been stopped and then restarted.
EDB_ERROR_SESSIONCURRENT (1108)	The current session ID <SessionID> cannot be disconnected or removedThis error is raised whenever a remote session attempts to disconnect or remove itself.
EDB_ERROR_COMPRESS (1200)	An error occurred while compressing data (<ErrorMessage>)This error is raised when ElevateDB encounters an issue while attempting to compress data. It is an internal error and will not occur unless there is a bug in ElevateDB. The specific error message is indicated within the parentheses.
EDB_ERROR_DECOMPRESS (1201)	An error occurred while uncompressing data

	(<ErrorMessage>)This error is raised when ElevateDB encounters an issue while attempting to decompress data. It is an internal error and will not occur unless there is a bug in ElevateDB. The specific error message is indicated within the parentheses.
EDB_ERROR_BACKUP (1300)	Error backing up the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the backing up of a database. The specific error message is indicated within the parentheses.
EDB_ERROR_RESTORE (1301)	Error restoring the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the restore of a database. The specific error message is indicated within the parentheses.
EDB_ERROR_PUBLISH (1302)	Error publishing the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the publishing of a database. The specific error message is indicated within the parentheses.
EDB_ERROR_UNPUBLISH (1303)	Error unpublishing the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the unpublishing of a database. The specific error message is indicated within the parentheses.
EDB_ERROR_SAVEUPDATES (1304)	Error saving updates for the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the saving of the updates for a database. The specific error message is indicated within the parentheses.
EDB_ERROR_LOADUPDATES (1305)	Error loading updates for the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the loading of the updates for a database. The specific error message is indicated within the parentheses.
EDB_ERROR_STORE (1306)	Error with the store <StoreName> (<ErrorMessage>)This error is raised when any error occurs while trying to access a store, such as a read or write error while working with files in the store. The specific error message is indicated within the parentheses.
EDB_ERROR_CACHEUPDATES (1307)	Error caching updates for the cursor <CursorName> (<ErrorMessage>)This error is raised when any error occurs during the caching of updates for a specific table, view, or query cursor. The specific error message is indicated within the parentheses.
EDB_ERROR_FORMAT (1400)	Error in the format string <FormatString> (<ErrorMessage>)This error is raised when ElevateDB encounters an issue with a format string used in a date, time, or timestamp format used in a table import or export. The specific error message is indicated within the parentheses.

This page intentionally left blank

Appendix B - System Capacities

The following is a list of the capacities for the different objects in ElevateDB. Any object that is not specifically mentioned here has an implicit capacity of 2147483647, or High(Integer). For example, there is no stated capacity for the maximum number of roles allowed in a configuration. Therefore, the implicit capacity is 2147483647 roles.

Capacity	Details
Max BLOB Column Size	The maximum size of a BLOB column is 2GB.
Max CHAR/VARCHAR Column Length	The maximum length of a VARCHAR/CHAR columns is 1024 characters.
Max Identifier Length	The maximum length of an identifier is 80 characters.
Max Number of Columns in a Table	The maximum number of columns in a table is 2048.
Max Number of Columns in an Index	The maximum number of columns in an index is limited by the table's defined index page size.
Max Number of Concurrent Sessions	The maximum number of concurrent sessions for an application or ElevateDB server is 4096.
Max Number of Indexes in a Table	The maximum number of indexes in a table is 512.
Max Number of Jobs in a Configuration	The maximum number of jobs in a configuration is 4096.
Max Number of Routines in a Database	The maximum number of routines (procedures and functions combined) in a database is 4096.
Max Number of Rows in a Table	The maximum number of rows in a table is determined by whether global file I/O buffering is enabled in ElevateDB. If global file I/O buffering is enabled, then the maximum number of rows is determined by the maximum file size permitted in the operating system. If global file I/O buffering is not enabled, then the approximate maximum number of rows can be determined by dividing 128GB by the row size.
Max Number of Rows in a Transaction	The maximum number of rows in a single transaction is only limited by the available memory constraints of the operating system and/or hardware.
Max Number of Tables in a Database	The maximum number of tables in a database is 4096.
Max Number of Users in a Configuration	The maximum number of users in a configuration is 4096.
Max Row Size for a Table	The maximum row size for a table is 2GB.
Max Scale for DECIMAL/NUMERIC Columns	The maximum scale for DECIMAL or NUMERIC columns is 4.
Max Size of an In-Memory Table	The maximum size of an in-memory table is only limited by the available memory constraints of the operating system or hardware.
Min/Max BLOB Block Size for a Table	The minimum BLOB block size is 64 bytes for ANSI databases and 128 bytes for Unicode databases. The maximum BLOB block size is 2GB.

Min/Max Index Page Size for a Table	The minimum index page size is 1 kilobyte for ANSI databases and 2 kilobytes for Unicode databases. The maximum index page size is 2GB.