Navigation

* [index](http://docs.google.com/genindex.html)
* [next](http://docs.google.com/config.html) |
* [previous](http://docs.google.com/require.html) |
* [phpMyAdmin 4.5.4.1 documentation](http://docs.google.com/index.html) »

Installation[**¶**](#gjdgxs)

phpMyAdmin does not apply any special security methods to the MySQL database server. It is still the system administrator’s job to grant permissions on the MySQL databases properly. phpMyAdmin’s Users page can be used for this.

Warning

[Mac](http://docs.google.com/glossary.html#term-mac) users should note that if you are on a version before [Mac OS X](http://docs.google.com/glossary.html#term-mac-os-x), StuffIt unstuffs with [Mac](http://docs.google.com/glossary.html#term-mac) formats. So you’ll have to resave as in BBEdit to Unix style ALL phpMyAdmin scripts before uploading them to your server, as PHP seems not to like [Mac](http://docs.google.com/glossary.html#term-mac)-style end of lines character (“\r”).

Linux distributions[**¶**](#30j0zll)

phpMyAdmin is included in most Linux distributions. It is recommended to use distribution packages when possible - they usually provide integration to your distribution and you will automatically get security updates from your distribution.

Debian[**¶**](#1fob9te)

Debian’s package repositories include a phpMyAdmin package, but be aware that the configuration file is maintained in /etc/phpmyadmin and may differ in some ways from the official phpMyAdmin documentation.

OpenSUSE[**¶**](#3znysh7)

OpenSUSE already comes with phpMyAdmin package, just install packages from the [openSUSE Build Service](http://software.opensuse.org/package/phpMyAdmin).

Ubuntu[**¶**](#2et92p0)

Ubuntu ships phpMyAdmin package, however if you want to use recent version, you can use packages from [PPA for Michal Čihař](https://launchpad.net/~nijel/+archive/phpmyadmin).

Gentoo[**¶**](#tyjcwt)

Gentoo ships the phpMyAdmin package, both in a near stock configuration as well as in a webapp-config configuration. Use emerge dev-db/phpmyadmin to install.

Mandriva[**¶**](#3dy6vkm)

Mandriva ships the phpMyAdmin package in their contrib branch and can be installed via the usual Control Center.

Fedora[**¶**](#1t3h5sf)

Fedora ships the phpMyAdmin package, but be aware that the configuration file is maintained in /etc/phpMyAdmin/ and may differ in some ways from the official phpMyAdmin documentation.

Red Hat Enterprise Linux[**¶**](#4d34og8)

Red Hat Enterprise Linux itself and thus derivatives like CentOS don’t ship phpMyAdmin, but the Fedora-driven repository [Extra Packages for Enterprise Linux (EPEL)](http://fedoraproject.org/wiki/EPEL) is doing so, if it’s [enabled](http://fedoraproject.org/wiki/EPEL/FAQ#howtouse). But be aware that the configuration file is maintained in /etc/phpMyAdmin/ and may differ in some ways from the official phpMyAdmin documentation.

Installing on Windows[**¶**](#2s8eyo1)

The easiest way to get phpMyAdmin on Windows is using third party products which include phpMyAdmin together with a database and web server such as [XAMPP](http://www.apachefriends.org/en/xampp.html).

You can find more of such options at [Wikipedia](https://en.wikipedia.org/wiki/List_of_AMP_packages).

Quick Install[**¶**](#17dp8vu)

1. Choose an appropriate distribution kit from the phpmyadmin.net Downloads page. Some kits contain only the English messages, others contain all languages. We’ll assume you chose a kit whose name looks like phpMyAdmin-x.x.x -all-languages.tar.gz.
2. Ensure you have downloaded a genuine archive, see [Verifying phpMyAdmin releases](#3rdcrjn).
3. Untar or unzip the distribution (be sure to unzip the subdirectories): tar -xzvf phpMyAdmin\_x.x.x-all-languages.tar.gz in your webserver’s document root. If you don’t have direct access to your document root, put the files in a directory on your local machine, and, after step 4, transfer the directory on your web server using, for example, ftp.
4. Ensure that all the scripts have the appropriate owner (if PHP is running in safe mode, having some scripts with an owner different from the owner of other scripts will be a problem). See [4.2 What’s the preferred way of making phpMyAdmin secure against evil access?](http://docs.google.com/faq.html#faq4-2) and [1.26 I just installed phpMyAdmin in my document root of IIS but I get the error “No input file specified” when trying to run phpMyAdmin.](http://docs.google.com/faq.html#faq1-26) for suggestions.
5. Now you must configure your installation. There are two methods that can be used. Traditionally, users have hand-edited a copy of config.inc.php, but now a wizard-style setup script is provided for those who prefer a graphical installation. Creating a config.inc.php is still a quick way to get started and needed for some advanced features.

Manually creating the file[**¶**](#26in1rg)

To manually create the file, simply use your text editor to create the file config.inc.php (you can copy config.sample.inc.php to get a minimal configuration file) in the main (top-level) phpMyAdmin directory (the one that contains index.php). phpMyAdmin first loads libraries/config.default.php and then overrides those values with anything found in config.inc.php. If the default value is okay for a particular setting, there is no need to include it in config.inc.php. You’ll probably need only a few directives to get going; a simple configuration may look like this:

<?php  
$cfg['blowfish\_secret'] = 'ba17c1ec07d65003'; // use here a value of your choice  
  
$i=0;  
$i++;  
$cfg['Servers'][$i]['auth\_type'] = 'cookie';  
?>

Or, if you prefer to not be prompted every time you log in:

<?php  
  
$i=0;  
$i++;  
$cfg['Servers'][$i]['user'] = 'root';  
$cfg['Servers'][$i]['password'] = 'cbb74bc'; // use here your password  
$cfg['Servers'][$i]['auth\_type'] = 'config';  
?>

For a full explanation of possible configuration values, see the [Configuration](http://docs.google.com/config.html#config) of this document.

Using Setup script[**¶**](#lnxbz9)

Instead of manually editing config.inc.php, you can use phpMyAdmin’s setup feature. First you must manually create a folder config in the phpMyAdmin directory. This is a security measure. On a Debian system you can use the following commands:

/usr/sbin/pma-configure

Note

Debian and Ubuntu have simplified this setup and all you need to do is to execute **/usr/sbin/pma-configure**.

On other platforms, simply create the folder and ensure that your web server has read and write access to it. [1.26 I just installed phpMyAdmin in my document root of IIS but I get the error “No input file specified” when trying to run phpMyAdmin.](http://docs.google.com/faq.html#faq1-26) can help with this.

Next, open your browser and visit the location where you installed phpMyAdmin, with the /setup suffix. If you have an existing configuration, use the Load button to bring its content inside the setup panel. Note that **changes are not saved to disk until you explicitly choose ``Save``** from the *Configuration* area of the screen. Normally the script saves the new config.inc.php to the config/ directory, but if the webserver does not have the proper permissions you may see the error “Cannot load or save configuration.” Ensure that the config/ directory exists and has the proper permissions - or use the Download link to save the config file locally and upload it (via FTP or some similar means) to the proper location.

Once the file has been saved, the permissions must be reset, again as a security measure:

/usr/sbin/pma-secure

Note

Debian and Ubuntu have simplified this setup and all you need to do is to execute **/usr/sbin/pma-secure**.

Now the file is ready to be used. You can choose to review or edit the file with your favorite editor, if you prefer to set some advanced options which the setup script does not provide.

1. If you are using the auth\_type “config”, it is suggested that you protect the phpMyAdmin installation directory because using config does not require a user to enter a password to access the phpMyAdmin installation. Use of an alternate authentication method is recommended, for example with HTTP–AUTH in a [.htaccess](http://docs.google.com/glossary.html#term-htaccess) file or switch to using auth\_type cookie or http. See the [ISPs, multi-user installations](http://docs.google.com/faq.html#faqmultiuser) for additional information, especially [4.4 phpMyAdmin always gives “Access denied” when using HTTP authentication.](http://docs.google.com/faq.html#faq4-4).
2. Open the [main phpMyAdmin directory](http://docs.google.com/index.php) in your browser. phpMyAdmin should now display a welcome screen and your databases, or a login dialog if using [HTTP](http://docs.google.com/glossary.html#term-http) or cookie authentication mode.
3. You should deny access to the ./libraries and ./setup/lib subfolders in your webserver configuration. Such configuration prevents from possible path exposure and cross side scripting vulnerabilities that might happen to be found in that code. For the Apache webserver, this is often accomplished with a [.htaccess](http://docs.google.com/glossary.html#term-htaccess) file in those directories.
4. It is generally a good idea to protect a public phpMyAdmin installation against access by robots as they usually can not do anything good there. You can do this using robots.txt file in root of your webserver or limit access by web server configuration, see [1.42 How can I prevent robots from accessing phpMyAdmin?](http://docs.google.com/faq.html#faq1-42).

Verifying phpMyAdmin releases[**¶**](#35nkun2)

Since July 2015 all phpMyAdmin releases are cryptographically signed by the releasing developer, who is currently Marc Delisle. His key id is 0x81AF644A, his PGP fingerprint is:

436F F188 4B1A 0C3F DCBF 0D79 FEFC 65D1 81AF 644A

and you can get more identification information from <https://keybase.io/lem9>. You should verify that the signature matches the archive you have downloaded. This way you can be sure that you are using the same code that was released.

Each archive is accompanied with .asc files which contains the PGP signature for it. Once you have both of them in the same folder, you can verify the signature:

$ gpg --verify phpMyAdmin-4.4.9-all-languages.zip.asc  
gpg: Signature made Fri Jun 12 13:09:58 2015 CEST using RSA key ID 81AF644A  
gpg: Can't check signature: No public key

As you can see gpg complains that it does not know the public key. At this point you should do one of the following steps:

* Download the keyring from [our download server](https://files.phpmyadmin.net/phpmyadmin.keyring), then import it with:

$ gpg --import phpmyadmin.keyring

* Download and import the key from one of the key servers:

$ gpg --keyserver hkp://pgp.mit.edu --recv-keys 81AF644A  
gpg: requesting key 81AF644A from hkp server pgp.mit.edu  
gpg: key 81AF644A: public key "Marc Delisle <marc@infomarc.info>" imported  
gpg: no ultimately trusted keys found  
gpg: Total number processed: 1  
gpg: imported: 1 (RSA: 1)

This will improve the situation a bit - at this point you can verify that the signature from the given key is correct but you still can not trust the name used in the key:

$ gpg --verify phpMyAdmin-4.4.9-all-languages.zip.asc  
gpg: Signature made Fri Jun 12 13:09:58 2015 CEST using RSA key ID 81AF644A  
gpg: Good signature from "Marc Delisle <marc@infomarc.info>" [unknown]  
gpg: WARNING: This key is not certified with a trusted signature!  
gpg: There is no indication that the signature belongs to the owner.  
Primary key fingerprint: 436F F188 4B1A 0C3F DCBF 0D79 FEFC 65D1 81AF 644A

The problem here is that anybody could issue the key with this name. You need to ensure that the key is actually owned by the mentioned person. The GNU Privacy Handbook covers this topic in the chapter [Validating other keys on your public keyring](https://www.gnupg.org/gph/en/manual.html#AEN335). The most reliable method is to meet the developer in person and exchange key fingerprints, however you can also rely on the web of trust. This way you can trust the key transitively though signatures of others, who have met the developer in person. For example you can see how [Marc’s key links to Linus’s key](http://pgp.cs.uu.nl/mk_path.cgi?FROM=00411886&TO=81AF644A).

Once the key is trusted, the warning will not occur:

$ gpg --verify phpMyAdmin-4.4.9-all-languages.zip.asc  
gpg: Signature made Fri Jun 12 13:09:58 2015 CEST using RSA key ID 81AF644A  
gpg: Good signature from "Marc Delisle <marc@infomarc.info>" [full]

Should the signature be invalid (the archive has been changed), you would get a clear error regardless of the fact that the key is trusted or not:

$ gpg --verify phpMyAdmin-4.4.9-all-languages.zip.asc  
gpg: Signature made Fri Jun 12 13:09:58 2015 CEST using RSA key ID 81AF644A  
gpg: BAD signature from "Marc Delisle <marc@infomarc.info>" [unknown]

phpMyAdmin configuration storage[**¶**](#1ksv4uv)

For a whole set of additional features (bookmarks, comments, [SQL](http://docs.google.com/glossary.html#term-sql)-history, tracking mechanism, [PDF](http://docs.google.com/glossary.html#term-pdf)-generation, column contents transformation, etc.) you need to create a set of special tables. Those tables can be located in your own database, or in a central database for a multi-user installation (this database would then be accessed by the controluser, so no other user should have rights to it).

Zero configuration[**¶**](#44sinio)

In many cases, this database structure can be automatically created and configured. This is called “Zero Configuration” mode and can be particularly useful in shared hosting situations. “Zeroconf” mode is on by default, to disable set [$cfg['ZeroConf']](http://docs.google.com/config.html#cfg_ZeroConf) to false.

The following three scenarios are covered by the Zero Configuration mode:

* When entering a database where the configuration storage tables are not present, phpMyAdmin offers to create them from the Operations tab.
* When entering a database where the tables do already exist, the software automatically detects this and begins using them. This is the most common situation; after the tables are initially created automatically they are continually used without disturbing the user; this is also most useful on shared hosting where the user is not able to edit config.inc.php and usually the user only has access to one database.
* When having access to multiple databases, if the user first enters the database containing the configuration storage tables then switches to another database, phpMyAdmin continues to use the tables from the first database; the user is not prompted to create more tables in the new database.

Manual configuration[**¶**](#2jxsxqh)

Please look at your ./sql/ directory, where you should find a file called *create\_tables.sql*. (If you are using a Windows server, pay special attention to [1.23 I’m running MySQL on a Win32 machine. Each time I create a new table the table and column names are changed to lowercase!](http://docs.google.com/faq.html#faq1-23)).

If you already had this infrastructure and:

* upgraded to MySQL 4.1.2 or newer, please use sql/upgrade\_tables\_mysql\_4\_1\_2+.sql.
* upgraded to phpMyAdmin 4.3.0 or newer from 2.5.0 or newer (<= 4.2.x), please use sql/upgrade\_column\_info\_4\_3\_0+.sql.

and then create new tables by importing sql/create\_tables.sql.

You can use your phpMyAdmin to create the tables for you. Please be aware that you may need special (administrator) privileges to create the database and tables, and that the script may need some tuning, depending on the database name.

After having imported the sql/create\_tables.sql file, you should specify the table names in your config.inc.php file. The directives used for that can be found in the [Configuration](http://docs.google.com/config.html#config).

You will also need to have a controluser ([$cfg['Servers'][$i]['controluser']](http://docs.google.com/config.html#cfg_Servers_controluser) and [$cfg['Servers'][$i]['controlpass']](http://docs.google.com/config.html#cfg_Servers_controlpass) settings) with the proper rights to those tables. For example you can create it using following statement:

GRANT SELECT, INSERT, UPDATE, DELETE ON <pma\_db>.\* TO 'pma'@'localhost' IDENTIFIED BY 'pmapass';

Upgrading from an older version[**¶**](#z337ya)

**Never** extract the new version over an existing installation of phpMyAdmin; we had evidence of problems caused by this.

Simply copy config.inc.php from your previous installation into the newly unpacked one. Configuration files from old versions may require some tweaking as some options have been changed or removed. For compatibility with PHP 5.3 and later, remove a set\_magic\_quotes\_runtime(0); statement that you might find near the end of your configuration file.

You should **not** copy libraries/config.default.php over config.inc.php because the default configuration file is version- specific.

If you have upgraded your MySQL server from a version previous to 4.1.2 to version 5.x or newer and if you use the phpMyAdmin configuration storage, you should run the [SQL](http://docs.google.com/glossary.html#term-sql) script found in sql/upgrade\_tables\_mysql\_4\_1\_2+.sql.

If you have upgraded your phpMyAdmin to 4.3.0 or newer from 2.5.0 or newer (<= 4.2.x) and if you use the phpMyAdmin configuration storage, you should run the [SQL](http://docs.google.com/glossary.html#term-sql) script found in sql/upgrade\_column\_info\_4\_3\_0+.sql.

Do not forget to clear the browser cache and to empty the old session by logging out and logging in again.

Using authentication modes[**¶**](#3j2qqm3)

[HTTP](http://docs.google.com/glossary.html#term-http) and cookie authentication modes are recommended in a **multi-user environment** where you want to give users access to their own database and don’t want them to play around with others. Nevertheless be aware that MS Internet Explorer seems to be really buggy about cookies, at least till version 6. Even in a **single-user environment**, you might prefer to use [HTTP](http://docs.google.com/glossary.html#term-http) or cookie mode so that your user/password pair are not in clear in the configuration file.

[HTTP](http://docs.google.com/glossary.html#term-http) and cookie authentication modes are more secure: the MySQL login information does not need to be set in the phpMyAdmin configuration file (except possibly for the [$cfg['Servers'][$i]['controluser']](http://docs.google.com/config.html#cfg_Servers_controluser)). However, keep in mind that the password travels in plain text, unless you are using the HTTPS protocol. In cookie mode, the password is stored, encrypted with the AES algorithm, in a temporary cookie.

Then each of the *true* users should be granted a set of privileges on a set of particular databases. Normally you shouldn’t give global privileges to an ordinary user, unless you understand the impact of those privileges (for example, you are creating a superuser). For example, to grant the user *real\_user* with all privileges on the database *user\_base*:

GRANT ALL PRIVILEGES ON user\_base.\* TO 'real\_user'@localhost IDENTIFIED BY 'real\_password';

What the user may now do is controlled entirely by the MySQL user management system. With HTTP or cookie authentication mode, you don’t need to fill the user/password fields inside the [$cfg['Servers']](http://docs.google.com/config.html#cfg_Servers).

HTTP authentication mode[**¶**](#1y810tw)

* Uses [HTTP](http://docs.google.com/glossary.html#term-http) Basic authentication method and allows you to log in as any valid MySQL user.
* Is supported with most PHP configurations. For [IIS](http://docs.google.com/glossary.html#term-iis) ([ISAPI](http://docs.google.com/glossary.html#term-isapi)) support using [CGI](http://docs.google.com/glossary.html#term-cgi) PHP see [1.32 Can I use HTTP authentication with IIS?](http://docs.google.com/faq.html#faq1-32), for using with Apache [CGI](http://docs.google.com/glossary.html#term-cgi) see [1.35 Can I use HTTP authentication with Apache CGI?](http://docs.google.com/faq.html#faq1-35).
* See also [4.4 phpMyAdmin always gives “Access denied” when using HTTP authentication.](http://docs.google.com/faq.html#faq4-4) about not using the [.htaccess](http://docs.google.com/glossary.html#term-htaccess) mechanism along with ‘[HTTP](http://docs.google.com/glossary.html#term-http)‘ authentication mode.

Cookie authentication mode[**¶**](#4i7ojhp)

* Username and password are stored in cookies during the session and password is deleted when it ends.
* With this mode, the user can truly log out of phpMyAdmin and log back in with the same username.
* If you want to allow users to enter any hostname to connect (rather than only servers that are configured in config.inc.php), see the [$cfg['AllowArbitraryServer']](http://docs.google.com/config.html#cfg_AllowArbitraryServer) directive.
* As mentioned in the [Requirements](http://docs.google.com/require.html#require) section, having the mcrypt extension will speed up access considerably, but is not required.

Signon authentication mode[**¶**](#2xcytpi)

* This mode is a convenient way of using credentials from another application to authenticate to phpMyAdmin to implement single signon solution.
* The other application has to store login information into session data (see [$cfg['Servers'][$i]['SignonSession']](http://docs.google.com/config.html#cfg_Servers_SignonSession)) or you need to implement script to return the credentials (see [$cfg['Servers'][$i]['SignonScript']](http://docs.google.com/config.html#cfg_Servers_SignonScript)).
* When no credentials are available, the user is being redirected to [$cfg['Servers'][$i]['SignonURL']](http://docs.google.com/config.html#cfg_Servers_SignonURL), where you should handle the login process.

The very basic example of saving credentials in a session is available as examples/signon.php:

<?php  
/\* vim: set expandtab sw=4 ts=4 sts=4: \*/  
/\*\*  
 \* Single signon for phpMyAdmin  
 \*  
 \* This is just example how to use session based single signon with  
 \* phpMyAdmin, it is not intended to be perfect code and look, only  
 \* shows how you can integrate this functionality in your application.  
 \*  
 \* @package PhpMyAdmin  
 \* @subpackage Example  
 \*/  
  
/\* Need to have cookie visible from parent directory \*/  
session\_set\_cookie\_params(0, '/', '', false);  
/\* Create signon session \*/  
$session\_name = 'SignonSession';  
session\_name($session\_name);  
// Uncomment and change the following line to match your $cfg['SessionSavePath']  
//session\_save\_path('/foobar');  
session\_start();  
  
/\* Was data posted? \*/  
if (isset($\_POST['user'])) {  
 /\* Store there credentials \*/  
 $\_SESSION['PMA\_single\_signon\_user'] = $\_POST['user'];  
 $\_SESSION['PMA\_single\_signon\_password'] = $\_POST['password'];  
 $\_SESSION['PMA\_single\_signon\_host'] = $\_POST['host'];  
 $\_SESSION['PMA\_single\_signon\_port'] = $\_POST['port'];  
 /\* Update another field of server configuration \*/  
 $\_SESSION['PMA\_single\_signon\_cfgupdate'] = array('verbose' => 'Signon test');  
 $id = session\_id();  
 /\* Close that session \*/  
 session\_write\_close();  
 /\* Redirect to phpMyAdmin (should use absolute URL here!) \*/  
 header('Location: ../index.php');  
} else {  
 /\* Show simple form \*/  
 header('Content-Type: text/html; charset=utf-8');  
 echo '<?xml version="1.0" encoding="utf-8"?>' . "\n";  
 ?>  
 <!DOCTYPE HTML>  
 <html lang="en" dir="ltr">  
 <head>  
 <link rel="icon" href="../favicon.ico" type="image/x-icon" />  
 <link rel="shortcut icon" href="../favicon.ico" type="image/x-icon" />  
 <meta charset="utf-8" />  
 <title>phpMyAdmin single signon example</title>  
 </head>  
 <body>  
 <?php  
 if (isset($\_SESSION['PMA\_single\_signon\_error\_message'])) {  
 echo '<p class="error">';  
 echo $\_SESSION['PMA\_single\_signon\_error\_message'];  
 echo '</p>';  
 }  
 ?>  
 <form action="signon.php" method="post">  
 Username: <input type="text" name="user" /><br />  
 Password: <input type="password" name="password" /><br />  
 Host: (will use the one from config.inc.php by default)  
 <input type="text" name="host" /><br />  
 Port: (will use the one from config.inc.php by default)  
 <input type="text" name="port" /><br />  
 <input type="submit" />  
 </form>  
 </body>  
 </html>  
 <?php  
}  
?>

Alternatively you can also use this way to integrate with OpenID as shown in examples/openid.php:

<?php  
/\* vim: set expandtab sw=4 ts=4 sts=4: \*/  
/\*\*  
 \* Single signon for phpMyAdmin using OpenID  
 \*  
 \* This is just example how to use single signon with phpMyAdmin, it is  
 \* not intended to be perfect code and look, only shows how you can  
 \* integrate this functionality in your application.  
 \*  
 \* It uses OpenID pear package, see http://pear.php.net/package/OpenID  
 \*  
 \* User first authenticates using OpenID and based on content of $AUTH\_MAP  
 \* the login information is passed to phpMyAdmin in session data.  
 \*  
 \* @package PhpMyAdmin  
 \* @subpackage Example  
 \*/  
  
if (false === @include\_once 'OpenID/RelyingParty.php') {  
 exit;  
}  
  
/\*\*  
 \* Map of authenticated users to MySQL user/password pairs.  
 \*/  
$AUTH\_MAP = array(  
 'http://launchpad.net/~username' => array(  
 'user' => 'root',  
 'password' => '',  
 ),  
 );  
  
/\*\*  
 \* Simple function to show HTML page with given content.  
 \*  
 \* @param string $contents Content to include in page  
 \*  
 \* @return void  
 \*/  
function Show\_page($contents)  
{  
 header('Content-Type: text/html; charset=utf-8');  
 echo '<?xml version="1.0" encoding="utf-8"?>' . "\n";  
 ?>  
 <!DOCTYPE HTML>  
 <html lang="en" dir="ltr">  
 <head>  
 <link rel="icon" href="../favicon.ico" type="image/x-icon" />  
 <link rel="shortcut icon" href="../favicon.ico" type="image/x-icon" />  
 <meta charset="utf-8" />  
 <title>phpMyAdmin OpenID signon example</title>  
 </head>  
 <body>  
 <?php  
 if (isset($\_SESSION) && isset($\_SESSION['PMA\_single\_signon\_error\_message'])) {  
 echo '<p class="error">' . $\_SESSION['PMA\_single\_signon\_message'] . '</p>';  
 unset($\_SESSION['PMA\_single\_signon\_message']);  
 }  
 echo $contents;  
 ?>  
 </body>  
 </html>  
 <?php  
}  
  
/\* Need to have cookie visible from parent directory \*/  
session\_set\_cookie\_params(0, '/', '', false);  
/\* Create signon session \*/  
$session\_name = 'SignonSession';  
session\_name($session\_name);  
session\_start();  
  
// Determine realm and return\_to  
$base = 'http';  
if (isset($\_SERVER['HTTPS']) && $\_SERVER['HTTPS'] == 'on') {  
 $base .= 's';  
}  
$base .= '://' . $\_SERVER['SERVER\_NAME'] . ':' . $\_SERVER['SERVER\_PORT'];  
  
$realm = $base . '/';  
$returnTo = $base . dirname($\_SERVER['PHP\_SELF']);  
if ($returnTo[/\*overload\*/mb\_strlen($returnTo) - 1] != '/') {  
 $returnTo .= '/';  
}  
$returnTo .= 'openid.php';  
  
/\* Display form \*/  
if (!count($\_GET) && !count($\_POST) || isset($\_GET['phpMyAdmin'])) {  
 /\* Show simple form \*/  
 $content = '<form action="openid.php" method="post">  
OpenID: <input type="text" name="identifier" /><br />  
<input type="submit" name="start" />  
</form>  
</body>  
</html>';  
 Show\_page($content);  
 exit;  
}  
  
/\* Grab identifier \*/  
if (isset($\_POST['identifier'])) {  
 $identifier = $\_POST['identifier'];  
} else if (isset($\_SESSION['identifier'])) {  
 $identifier = $\_SESSION['identifier'];  
} else {  
 $identifier = null;  
}  
  
/\* Create OpenID object \*/  
try {  
 $o = new OpenID\_RelyingParty($returnTo, $realm, $identifier);  
} catch (OpenID\_Exception $e) {  
 $contents = "<div class='relyingparty\_results'>\n";  
 $contents .= "<pre>" . $e->getMessage() . "</pre>\n";  
 $contents .= "</div class='relyingparty\_results'>";  
 Show\_page($contents);  
 exit;  
}  
  
/\* Redirect to OpenID provider \*/  
if (isset($\_POST['start'])) {  
 try {  
 $authRequest = $o->prepare();  
 } catch (OpenID\_Exception $e) {  
 $contents = "<div class='relyingparty\_results'>\n";  
 $contents .= "<pre>" . $e->getMessage() . "</pre>\n";  
 $contents .= "</div class='relyingparty\_results'>";  
 Show\_page($contents);  
 exit;  
 }  
  
 $url = $authRequest->getAuthorizeURL();  
  
 header("Location: $url");  
 exit;  
} else {  
 /\* Grab query string \*/  
 if (!count($\_POST)) {  
 list(, $queryString) = explode('?', $\_SERVER['REQUEST\_URI']);  
 } else {  
 // I hate php sometimes  
 $queryString = file\_get\_contents('php://input');  
 }  
  
 /\* Check reply \*/  
 $message = new OpenID\_Message($queryString, OpenID\_Message::FORMAT\_HTTP);  
  
 $id = $message->get('openid.claimed\_id');  
  
 if (!empty($id) && isset($AUTH\_MAP[$id])) {  
 $\_SESSION['PMA\_single\_signon\_user'] = $AUTH\_MAP[$id]['user'];  
 $\_SESSION['PMA\_single\_signon\_password'] = $AUTH\_MAP[$id]['password'];  
 session\_write\_close();  
 /\* Redirect to phpMyAdmin (should use absolute URL here!) \*/  
 header('Location: ../index.php');  
 } else {  
 Show\_page('<p>User not allowed!</p>');  
 exit;  
 }  
}

If you intend to pass the credentials using some other means than, you have to implement wrapper in PHP to get that data and set it to [$cfg['Servers'][$i]['SignonScript']](http://docs.google.com/config.html#cfg_Servers_SignonScript). There is very minimal example in examples/signon-script.php:

<?php  
/\* vim: set expandtab sw=4 ts=4 sts=4: \*/  
/\*\*  
 \* Single signon for phpMyAdmin  
 \*  
 \* This is just example how to use script based single signon with  
 \* phpMyAdmin, it is not intended to be perfect code and look, only  
 \* shows how you can integrate this functionality in your application.  
 \*  
 \* @package PhpMyAdmin  
 \* @subpackage Example  
 \*/  
  
  
/\*\*  
 \* This function returns username and password.  
 \*  
 \* It can optionally use configured username as parameter.  
 \*  
 \* @param string $user User name  
 \*  
 \* @return array  
 \*/  
function get\_login\_credentials($user)  
{  
 /\* Optionally we can use passed username \*/  
 if (!empty($user)) {  
 return array($user, 'password');  
 }  
  
 /\* Here we would retrieve the credentials \*/  
 $credentials = array('root', '');  
  
 return $credentials;  
}

See also

[$cfg['Servers'][$i]['auth\_type']](http://docs.google.com/config.html#cfg_Servers_auth_type), [$cfg['Servers'][$i]['SignonSession']](http://docs.google.com/config.html#cfg_Servers_SignonSession), [$cfg['Servers'][$i]['SignonScript']](http://docs.google.com/config.html#cfg_Servers_SignonScript), [$cfg['Servers'][$i]['SignonURL']](http://docs.google.com/config.html#cfg_Servers_SignonURL)

Config authentication mode[**¶**](#1ci93xb)

* This mode is sometimes the less secure one because it requires you to fill the [$cfg['Servers'][$i]['user']](http://docs.google.com/config.html#cfg_Servers_user) and [$cfg['Servers'][$i]['password']](http://docs.google.com/config.html#cfg_Servers_password) fields (and as a result, anyone who can read your config.inc.php can discover your username and password).
* In the [ISPs, multi-user installations](http://docs.google.com/faq.html#faqmultiuser) section, there is an entry explaining how to protect your configuration file.
* For additional security in this mode, you may wish to consider the Host authentication [$cfg['Servers'][$i]['AllowDeny']['order']](http://docs.google.com/config.html#cfg_Servers_AllowDeny_order) and [$cfg['Servers'][$i]['AllowDeny']['rules']](http://docs.google.com/config.html#cfg_Servers_AllowDeny_rules) configuration directives.
* Unlike cookie and http, does not require a user to log in when first loading the phpMyAdmin site. This is by design but could allow any user to access your installation. Use of some restriction method is suggested, perhaps a [.htaccess](http://docs.google.com/glossary.html#term-htaccess) file with the HTTP-AUTH directive or disallowing incoming HTTP requests at one’s router or firewall will suffice (both of which are beyond the scope of this manual but easily searchable with Google).

Swekey authentication mode[**¶**](#3whwml4)

The Swekey is a low cost authentication USB key that can be used in web applications. When Swekey authentication is activated, phpMyAdmin requires the users’s Swekey to be plugged before entering the login page (currently supported for cookie authentication mode only). Swekey Authentication is disabled by default. To enable it, add the following line to config.inc.php:

$cfg['Servers'][$i]['auth\_swekey\_config'] = '/etc/swekey.conf';

You then have to create the swekey.conf file that will associate each user with their Swekey Id. It is important to place this file outside of your web server’s document root (in the example, it is located in /etc). Feel free to use it with your own users’ information. If you want to purchase a Swekey please visit <https://www.phpmyadmin.net/auth_key/> since this link provides funding for phpMyAdmin.

A self documented sample file is provided in the file examples/swekey.sample.conf:

# This is a typical file used to enable Swekey hardware authentication.  
#  
# To activate the Swekey authentication add the following line to your config.inc.php file.  
# $cfg['Servers'][$i]['auth\_swekey\_config'] = '/etc/swekey-pma.conf';  
# Then rename this file "swekey-pma.conf" and copy it to the /etc directory.  
# Add all the Swekey ids you want to grant access to in the file.  
# After each Swekey id put the corresponding user name.  
#  
# If you don't know the id of a Swekey just visit http://www.swekey.com?sel=support  
# while your Swekey is connected.  
#  
# If you need to purchase a Swekey please visit https://www.phpmyadmin.net/auth\_key/  
# since this link provides funding to PhpMyAdmin.  
#  
  
0000000000000000000000000000763A:root  
000000000000000000000000000089E4:steve  
0000000000000000000000000000231E:scott  
  
#  
# It is recommended to include the following lines to contact the  
# authentication servers in SSL mode.  
#  
  
SERVER\_CHECK=https://auth-check-ssl.musbe.net  
SERVER\_RNDTOKEN=https://auth-rnd-gen-ssl.musbe.net  
SERVER\_STATUS=https://auth-status-ssl.musbe.net  
  
#  
# The path of the root certificate file used to ensure a secure  
# communication with the authentication servers in SSL mode.  
# If not specified, will use musbe-ca.crt found in your  
# phpMyAdmin/libraries/auth/swekey.  
#  
  
#CA\_FILE=/var/http-root/phpmyadmin/libraries/auth/swekey/musbe-ca.crt  
  
#  
# If your server receives many login requests, you can enable the random  
# token caching to accelerate the authentication process.  
# Token caching is enabled by default.  
#  
  
#ENABLE\_TOKEN\_CACHE=0

See also

[$cfg['Servers'][$i]['auth\_swekey\_config']](http://docs.google.com/config.html#cfg_Servers_auth_swekey_config)

Securing your phpMyAdmin installation[**¶**](#2bn6wsx)

The phpMyAdmin team tries hard to make the application secure, however there are always ways to make your installation more secure:

* Remove the setup directory from phpMyAdmin, you will probably not use it after the initial setup.
* Properly choose an authentication method - [Cookie authentication mode](#qsh70q) is probably the best choice for shared hosting.
* In case you don’t want all MySQL users to be able to access phpMyAdmin, you can use [$cfg['Servers'][$i]['AllowDeny']['rules']](http://docs.google.com/config.html#cfg_Servers_AllowDeny_rules) to limit them.
* Consider hiding phpMyAdmin behind an authentication proxy, so that users need to authenticate prior to providing MySQL credentials to phpMyAdmin. You can achieve this by configuring your web server to request HTTP authentication. For example in Apache this can be done with:  
  AuthType Basic  
  AuthName "Restricted Access"  
  AuthUserFile /usr/share/phpmyadmin/passwd  
  Require valid-user  
    
  Once you have changed the configuration, you need to create a list of users which can authenticate. This can be done using the **htpasswd** utility:  
  htpasswd -c /usr/share/phpmyadmin/passwd username
* If you are afraid of automated attacks, enabling Captcha by [$cfg['CaptchaLoginPublicKey']](http://docs.google.com/config.html#cfg_CaptchaLoginPublicKey) and [$cfg['CaptchaLoginPrivateKey']](http://docs.google.com/config.html#cfg_CaptchaLoginPrivateKey) might be an option.

[**Table Of Contents**](http://docs.google.com/index.html)

* [Installation](#3as4poj)
  + [Linux distributions](#30j0zll)
    - [Debian](#1fob9te)
    - [OpenSUSE](#3znysh7)
    - [Ubuntu](#2et92p0)
    - [Gentoo](#tyjcwt)
    - [Mandriva](#3dy6vkm)
    - [Fedora](#1t3h5sf)
    - [Red Hat Enterprise Linux](#4d34og8)
  + [Installing on Windows](#2s8eyo1)
  + [Quick Install](#17dp8vu)
    - [Manually creating the file](#26in1rg)
    - [Using Setup script](#lnxbz9)
  + [Verifying phpMyAdmin releases](#35nkun2)
  + [phpMyAdmin configuration storage](#1ksv4uv)
    - [Zero configuration](#44sinio)
    - [Manual configuration](#2jxsxqh)
  + [Upgrading from an older version](#z337ya)
  + [Using authentication modes](#3j2qqm3)
    - [HTTP authentication mode](#1y810tw)
    - [Cookie authentication mode](#4i7ojhp)
    - [Signon authentication mode](#2xcytpi)
    - [Config authentication mode](#1ci93xb)
    - [Swekey authentication mode](#3whwml4)
  + [Securing your phpMyAdmin installation](#2bn6wsx)

#### Previous topic

[Requirements](http://docs.google.com/require.html)

#### Next topic

[Configuration](http://docs.google.com/config.html)

### This Page

* [Show Source](http://docs.google.com/_sources/setup.txt)

Quick search

Enter search terms or a module, class or function name.

Navigation

* [index](http://docs.google.com/genindex.html)
* [next](http://docs.google.com/config.html) |
* [previous](http://docs.google.com/require.html) |
* [phpMyAdmin 4.5.4.1 documentation](http://docs.google.com/index.html) »

© [Copyright](http://docs.google.com/copyright.html) 2012 - 2014, The phpMyAdmin devel team. Created using [Sphinx](http://sphinx-doc.org/) 1.3.6.