PHP Login Suite V2

User Manual
About the PHP Login Suite

I’d like to start by saying thank you for purchasing the PHP Login Suite. The suite has been a lot more successful than I imagined, and now has a large number of users, and a nice community growing in the PHP Login Suite support newsgroup.

What is the PHP Login Suite?

The suite is a set of Dreamweaver MX Objects and Server Behaviours, which allow you to easily create a secure login system using PHP. There are three sections in the login suite, two of which contain Server Behaviors for working with Sessions and Cookies, and the third is a section containing Server Behaviors which perform general tasks.

What’s new in the PHP Login Suite Version 2?

Version 2 of the PHP Login Suite contains a number of new extensions, doubling the size of the existing suite. Many of the new Server Behaviors were based on feedback that I’ve received since launching the original suite.

The main enhancement is that all of the Server Behaviors that work with passwords now have a duplicate Server Behavior to enable you to use encrypted passwords. There is also a Server Behavior that allows you to convert a table containing user’s details with a plaintext password, into an encrypted password, to work with the new extensions.

Another new Server Behavior allows you to check whether a user’s username is unique, and to insert it in the database if it is. Plus there are a number of Server Behaviors that work with email, and allow you to create a page that sends the user their password by email, if they forget it.

IMPORTANT NOTE - PHP Global Variables

Since the PHP Login Suite was written, there has been a major change in the PHP installation concerning the Global Variables options, which is now set to OFF by default.

When Global Variables where on, you could refer to a session for example, just by using its name for example $user Session. Now that Global Variables are off, you can no longer do this, instead you have to specify the PHP array in which the data is held.

The array names are shown below:

<table>
<thead>
<tr>
<th>To get the value of a form field using POST</th>
<th>$HTTP_POST_VARS['fieldname']</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the value from a URL QueryString</td>
<td>$HTTP_GET_VARS['variablename']</td>
</tr>
<tr>
<td>To get the value from a Cookie</td>
<td>$HTTP_COOKIE_VARS['cookiename']</td>
</tr>
<tr>
<td>To get the value from a Session Variable</td>
<td>$HTTP_SESSION_VARS['sessionname']</td>
</tr>
</tbody>
</table>

The relevant behaviours in the login suite have now been changed to refer to the proper array that the variables are stored in. This means however, that existing extensions may not be recognised correctly. It’s recommended that you remove the existing extensions, and reapply the new extension, if the old extension is no longer recognized correctly.
PHP Login Suite Index

Installing the PHP Login Suite 4

PHP Login Suite - Objects 5

Server Behaviors - Cookies 7
PHPLSCO1 – Login & Set Cookie 8
PHPLSCO2 – Login & Set Level 11
PHPLSCO3 – Access By Cookie 13
PHPLSCO4 – Access By Level 14
PHPLSCO5 – User Details 15
PHPLSCO6 – Kill a Cookie 17
PHPLSCO7 – Insert & Login 18
PHPLSCO8 – Check Username is Unique & Insert 20
PHPLSCO9 – User Change Password 23

Cookies & Encrypted Passwords 25
PHPLSCE01 – Login & Set Cookie 26
PHPLSCE02 – Login & Set Level 29
PHPLSCE03 – Check Username is Unique & Insert 31
PHPLSCE04 – User Change Password 34

Server Behaviors – Session Variables 36
PHPLS01 – Login & Set Session 37
PHPLS02 – Login & Set Level 39
PHPLS03 – Access By Session 41
PHPLS04 – Access By Level 42
PHPLS05 – Enable Sessions 43
PHPLS06 – User Details 44
PHPLS07 – Kill a Session 46
PHPLS08 – Kill all Sessions 47
PHPLS09 – Insert & Login 48
PHPLS10 – Check Username is Unique and Insert 49
PHPLS11 – User Change Password 51

Session Variables and Encrypted Passwords 53
PHPLSSE01 – Login & Set Session 54
PHPLSSE02 – Login & Set Level 56
PHPLSSE03 – Check Username is Unique & Insert 58
PHPLSSE04 – User Change Password 61
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PHP Login Suite – General Behaviors</strong></td>
<td>63</td>
</tr>
<tr>
<td>PHPLSG01 – Turn Off Notices</td>
<td>64</td>
</tr>
<tr>
<td>PHPLSG02 – PHP Info</td>
<td>65</td>
</tr>
<tr>
<td>PHPLSG03 – Email User their Forgotten Password</td>
<td>66</td>
</tr>
<tr>
<td><strong>PHP Login Suite – General Behaviors - Encrypted</strong></td>
<td>68</td>
</tr>
<tr>
<td>PHPLSGE01 – Convert Plain text password to Encrypted</td>
<td>68</td>
</tr>
<tr>
<td>PHPLSGE02 – Reset Users Forgotten Password</td>
<td>70</td>
</tr>
<tr>
<td><strong>Special Techniques</strong></td>
<td>72</td>
</tr>
<tr>
<td>Redirect User to a Page depending on their Access Level</td>
<td>72</td>
</tr>
<tr>
<td><strong>Troubleshooting</strong></td>
<td>75</td>
</tr>
<tr>
<td>Unwanted Notices</td>
<td>75</td>
</tr>
<tr>
<td>Error: Cannot send header information, header sent</td>
<td>76</td>
</tr>
<tr>
<td><strong>Session Variable Problems</strong></td>
<td>78</td>
</tr>
<tr>
<td>Testing Session Variables</td>
<td>78</td>
</tr>
<tr>
<td>Enabling Session Variables</td>
<td>79</td>
</tr>
<tr>
<td><strong>PHP Resources</strong></td>
<td>80</td>
</tr>
<tr>
<td>Dreamweaver MX Support Website</td>
<td>80</td>
</tr>
<tr>
<td>PHP Login Suite Newsgroup</td>
<td>80</td>
</tr>
<tr>
<td>PHP Online Manual</td>
<td>81</td>
</tr>
</tbody>
</table>
Installing the PHP Login Suite

To install the PHP Login Suite, you need to open the Macromedia Extension Manager, which is shown below.

If you already have the PHP Login Suite, and you are upgrading to a later version, you must first remove the previous version of the extension.

If Dreamweaver MX is running, then close it while the extension is removed and installed. Next, select the PHP Login Suite entry in the Extension Manager, and remove it by clearing its check box. Next, select “Remove” from the “File” menu, or press Ctrl + R, and then select “Yes” to totally remove the extension.

To Install the PHP Login Suite

From the “File” menu, select “Install Extension”, or press Ctrl + I. You can then select the PHP Login Suite extension, click “OK”, and the login suite will install.

Note that Dreamweaver MX should not be running while you install the extension.

Once you have installed the suite, it will appear in the Server Behaviors menu on the Dreamweaver MX “Application Panel”, for any sites using the Dreamweaver MX PHP/MySQL server model.

There are also some objects in the Dreamweaver MX “Insert Bar”, on the “Buzz Inet” tab.
PHP Login Suite – Objects

The first section of the Login Suite that we’re going to look at is the **Objects** Section. All of these objects are located in the Dreamweaver MX “Insert Panel”, which is usually at the top of your working page in Dreamweaver MX. Click on the “**Buzz Inet**” tab, and you’ll see the following objects on the Insert Bar.

![Insert Bar](image)

*Figure 1 – The Insert Bar*

**Form Objects**

The first two objects, containing a Padlock Icon, are used to easily create a form on your page, with a username and password field. The first icon, with the yellow border, creates a form, a table with a border, and the relevant fields, named **username** and **password**.

![Form with border](image)

*Figure 2 – Table with Border*

The second icon creates a similar form, but this time with a borderless table as shown below.

![Borderless Form](image)

*Figure 3 – Borderless Table*

These two forms can be easily and quickly inserted, simply by placing the cursor on the page at the desired insertion point, and clicking the icon. These forms can then be used with the relevant server behaviours.
Link Objects

The next set of icons when clicked, open a new Browser window, and opens the website that corresponds to the icon.

This icon opens a new browser window, showing the Buzz inet site at:

http://www.buzzinet.co.uk/

This icon opens a new browser window, showing the PHP Login Suite page at the Dreamweaver MX Support website at:


This page will always contain the latest information about the PHP Login Suite.

This icon when clicked opens a new browser window at the index page of the Online PHP manual, at:


This is really useful when you’re developing with PHP, as it allows you to quickly and easily get to the PHP online manual to lookup a PHP function for example.

This completes the Objects section, and in the next section we start to look at the Login Suites Server Behaviors, starting with the Cookie Behaviors.
Server Behaviors – Cookies

The Server Behaviors are divided into two types, allowing you to work with either Cookies or Session Variables to store the logged in user. In this section we look at the Cookie Server Behaviors that are available.

When to use Cookies

Cookies are normally used for long term storage. For example, once a user has logged in, each time they go back to the site they will be automatically logged back in, using their details stored in a cookie. You can set a timeout period for the cookie, so that it expires after a certain amount of time, such as 7 days or a month.

One limitation of Cookies is that a maximum of 4K of data can be stored in a cookie. In practice though, this rarely presents a problem, as it's usually only a username for example that is stored in a cookie, which only requires a few bytes.

Plain Text Password or Encrypted Passwords

In the new login suite, there are now two sections containing two types of Server Behaviors for working with Cookies, shown in the screenshot below.

![Figure 7 – Cookie Server Behaviors](image)

The Server Behaviors, PHPLSC01 – PHPLSC04 are new, and allow you to work with encrypted passwords in your database. These use the SQL PASSWORD() function to encrypt the users password entered in a form, and compare it against an encrypted password stored in a database.

The Server Behaviors, PHPLSC01 – PHPLSC04 allow you to work with passwords that have been stored as plain text in your database, i.e. they’re not encrypted.

The other Cookie Behaviors are not password specific. You should only work with the encrypted password behaviors if you store encrypted passwords in your database. There is a new behavior in the General folder, which we look at later, which allows you to convert plain text passwords in your database into an encrypted form. We are now going to look at each of the Server Behaviors individually.
PHPLSC01 – Login and Set Cookie

PHPLSC01 is used to log a user in, checking their username and password against details stored in a database table, and if the details match the user is logged in, and a field such as their username can be stored in a cookie. You can then allow only users who have cookies set to view certain pages.

Before you apply the behavior, you need to add a form and two fields, one for the user’s username, and one for their password, and you should then save the page.

Figure 8 – PHPLSC01 Dialog Box
The first set of options allows you to select the form that you want to work with. The *Forms* dropdown contains a list of all the forms on your page. Once you have selected a form to use, the *Username* and *Password* menus will be populated with a list of fields in the selected form. You can then set the username to the field that the user will use to enter their username and the password to the field that the user uses to enter their password.

The next section refers to the database table that you want to use to check the users details against. The *DB Connection* menu contains a list of all the database connections that you have set up for the current site in Dreamweaver MX, select the database connection that you wish to use.

Once you have selected a database connection to use, the *DB Table* menu will contain a list of all the tables in the current database. Select the table that holds your user details. Once you have selected the database table to use, you can select the field in the database table that holds *Usernames* and the field that stores *Passwords*.

The third section allows you to setup details for the Cookie that’s set when the user logs in, and is used to store the user’s username. The first option allows you to set the *Cookie Name*, which can be anything you wish.

The next two options, *Cookie Expire Time*, and *Cookie Expire Date*, allow you to set how long the cookie will last for, and you can use the following combinations.

| Cookie Expire Time: | 0 |
| Cookie Expire Date: | 0 |

This means the Cookie will expire as soon as the user has closed their browser.

| Cookie Expire Time: | 0 |
| Cookie Expire Date: | Number of days e.g. 30 |

This means the Cookie will expire in 30 days.

| Cookie Expire Time: | hh:mm:ss e.g. 12:00:00 |
| Cookie Expire Date: | dd/mm/yy e.g. 15/03/03 |

This means the cookie will expire at 12:00:00 on the 15th March 2003.

The last section allows you to set an *Error Message* that will be displayed to the user if their username or password is not found in the database.

You also need to enter a page that the user is *Redirected To* if their login attempt is successful. You can enter an address by hand, or click “*Browse*” to select another page on your site, and the link will be entered automatically.

Once you have filled in all the options, you can click “OK” and the required code will be added to your page.
You can go back at any time by double clicking on the Server Behavior name in the Dreamweaver MX Application Panel, shown in the screenshot below:

![Dreamweaver MX Server Behaviors Panel](image)

*Figure 9 – Dreamweaver MX Server Behaviors Panel*

Also note that when you apply the Server Behavior, a Session Variable icon is created in the Dreamweaver MX Bindings tab, which can be used to access the value stored in the Cookie on your page, as shown below:

![Dreamweaver MX Bindings Tab](image)

*Figure 10 – Dreamweaver MX Bindings Tab*

Note that you can click the “Help” button at anytime, to see the Server Behaviors Help file. When the user has logged in, their username will be stored in the Cookie.
PHPLSC02 – Login and Set Level

PHPLSC02 is used to log a user in, checking their username and password against details stored in a database table, and if the details match the user is logged in, and their username is stored in one cookie, and an Access Level stored in another cookie. The Access Level needs to be numerical e.g. 1, 2 or 3 etc. The Dialog Box for this extension is shown below:

Figure 11 – PHPLSC02 Dialog Box
First select the form on your page that you wish to work with, and select the form field that will be used for the user’s username, and the form field that will hold the user’s password.

Next, select the Database Connection that you wish to work with, and then select a table from the database that holds the user information. You then need to select the database field that contains the username, the field that contains the user’s password, and finally the field that holds the users access level.

The next section allows you to create the Cookies you wish to use. This time there are two cookies that are set, one to hold the users username, and the other cookie to hold the users access level.

First, enter the names for the two Cookies that will store the username and access level. The next set of options for Cookie Expire Time, and Cookie Expire Date, allow you to set how long the cookies last before they expire. You can use the following combinations.

| Cookie Expire Time: | 0          |
| Cookie Expire Date: | 0          |

This means the Cookie will expire as soon as the user has closed their browser.

| Cookie Expire Time: | 0          |
| Cookie Expire Date: | Number of days e.g. 30 |

This means the Cookie will expire in 30 days.

| Cookie Expire Time: | hh:mm:ss e.g. 12:00:00 |
| Cookie Expire Date: | dd/mm/yy e.g. 15/03/03 |

This means the cookie will expire at 12:00:00 on the 15th March 2003.

Finally, you can set a Message that is displayed if the user could not be logged in, for example telling them their username or password were incorrect, and to try again.

You can then click “Browse”, or manually type in the page that the user will be redirected to once a successful login is made.

Click OK to apply the Server Behavior, and the relevant code will be generated and added to your page.

Once again, the two Cookies that were created are available on the Dreamweaver MX Bindings Tab for use on your page.

After the behavior has been applied, you can double click the server behavior name in the Dreamweaver MX “Server Behaviors” window, and the dialog box will be displayed with your existing parameters. You can then change the parameters and reapply the extension, and it will modify the generated code accordingly.
PHPLSC03 – Access by Cookie

PHPLSC03 is used to secure pages, and works in conjunction with PHPLSC01 – Login and Set Cookie. You use PHPLSC01 to log a user in and to store their username in a Cookie. You then apply this behavior, PHPLSC03, to all pages that need protecting.

Only users who have the relevant cookie set will be allowed to view the page, and all other users are redirected to a page of your choice, for example a page to allow them to log in.

To apply the Server Behavior, simple select the Cookie that you wish to check from the menu, and enter or Browse to a page that the user is directed to if they are not allowed to see the page, such as a login page.

You can change the parameters used at any time, by double clicking the behavior name in the Dreamweaver MX Server Behaviors tab, and then changing either the cookie or the page the user is redirected to if they are not allowed to see the page.
PHPLSC04 – Access by Level

PHPLSC04 is used in conjunction with PHPLSC02 – Log in and set Level. You apply it to every page of your site that needs securing. When a user tries to view a secured page, the code first checks that the user has a cookie set for both their username and access level, and that the users access level is sufficient to view the page.

First, select the Cookie that was used to store the users Access Level.

Next, you use a combination of Operator and Access Level, to set which users CAN'T see the page. So for example, with the settings above, any one with a User Level of 1 will be refused access to the page and redirected, and everyone else who has an Access Level of 2 or more are allowed to view the page.

Finally, you can specify a page to redirect the user to if they are NOT allowed to see the page that's being protected.

You can change the parameters at any time, by double clicking the Server Behavior in the Dreamweaver MX “Server Behaviors” tab, and then changing the desired parameter.
PHPLSC05 – User Details

PHPLSC05 is used to select a user's record in a database by their username which is stored in a cookie.

For example, imagine that a user has logged in, and their username which is “fred” is stored in a cookie called userCookie.

With the settings in the Dialog Box shown above, the username field in the pusers table will be queried with the username held in cookie UserCookie, i.e. fred. So the recordset created by this Server Behavior will shows all records from the pusers table, where the username field is equal to “fred”.

As an example, you could use this behavior to allow a logged in user to change the details stored in their user record, and the users record is selected from their username held in a cookie and then displayed on screen so they can change it if they wish.

As well as inserting its own code, this Server Behavior creates a Dreamweaver MX recordset called rsUserDetails with the SQL needed to find the correct record. This Recordset can be accessed separately to the extension, and as Dreamweaver MX recognises it as one of its own recordsets, it is displayed in the Dreamweaver MX “Bindings” tab, as with any other recordset, and shown below.
This means that you can work with the created recordset easily, and drag its fields onto your page to display the field's contents, or use the fields to dynamically populate a form.

Again, at any time you can double click the Server Behavior name in the Dreamweaver MX “Server Behaviors” Panel, and you can change any of the existing parameters.

Note, that if you delete the Server Behavior, using the – button, you'll get the following message:

The reason this message appears, is because the Server Behavior controls the Dreamweaver MX Recordset as well, and so removing the behavior will remove the recordset. So you can safely ignore this message and click the “OK” button, and both the server behavior and recordset will be removed from your page.
PHPLSC06 – Kill a Cookie

PHPLSC06 is used to kill a cookie, so that a user's cookie is no longer set, and the user will be logged out.

Figure 17 – PHPLSC06 Dialog Box

This Server Behavior is very simple to use, you just select the cookie you want to kill, and then click “OK”.

Like all the Server Behaviors you can double click its name in the “Dreamweaver MX” Server Behaviors panel if you wish to change the cookie parameter.

With this extension, the cookie that is selected is shown in the Server Behavior name, as shown in the screenshot below.

Figure 18 – Dreamweaver MX Server Behaviors Tab
PHPLSC07 – Insert and Login

PHPLSC07 is used on new user forms, to add new users to a database. First you need to apply the Dreamweaver MX “Insert Record” Server Behavior to the page, which will insert the user's details into the database. This Server Behavior is then applied, which logs the user in, by storing their username in a cookie.

![PHPLSC07 Dialog Box](image)

*Figure 19 – PHPLSC07 Dialog Box*

The first section allows you to select the field on the form whose value will be placed in the cookie.

The second section allows you to set the Cookie Name, and how long the cookie takes to expire, using a set of parameters shown below.

- **Cookie Expire Time**: 0
- **Cookie Expire Date**: 0

This means the Cookie will expire as soon as the user has closed their browser.
Cookie Expire Time: 0
Cookie Expire Date: Number of days e.g. 30

This means the Cookie will expire in 30 days.

Cookie Expire Time: hh:mm:ss e.g. 12:00:00
Cookie Expire Date: dd/mm/yy e.g. 15/03/03

This means the cookie will expire at 12:00:00 on the 15th March 2003.

To change any of the parameters once the behavior has been applied, double click on the server behavior name in the Dreamweaver MX “Server Behaviors” tab. You can then change the parameters and reapply the server behavior.

Note that you must have saved your page, and applied the Dreamweaver MX Insert Server Behavior before you apply this extension.
PHPLSC08 – Check Username & Insert

PHPLSC08 is used to check that a new users Username is unique before inserting it into the database. It also checks that the password is greater than a minimum length that you can specify, and that the password field and confirm password fields match.

You can use this behavior to check that a user’s username is unique, if it is unique, the user’s username and password are inserted into the database, and their username is stored in a cookie. You can then use another page to fill in the rest of the user’s details, and update the user’s record using their username stored in a cookie.

Figure 20 – PHPLSC08 Dialog Box
First, create a form with a username field and two password fields. In this example the username field is named *username* and the two password fields are named *password1* and *password2*. You create two password fields, so that the user has type the password twice, to make sure they have typed the correct password, as they can't see what they have typed because data entered into a password field is hidden.

![Figure 21 – Example Form](image)

Once you have the form created as above, you can then save the page, and then apply the extension.

The first section of the dialog box allows you to select the form that you are using to gather the users username and password. Once you have selected a form, you can select the form field that holds the users username. Next, you can select the form field that contains the user’s password, and a second password field to hold the password again as confirmation. If you don't want to use have an extra password field to confirm the password, you can set both options to the same password field.

The next section concerns the database used to store the user’s information. First select a database connection to use to connect to your database. Next, you can select the table that will contain user’s information. You can then select the database field that will contain the user’s username, and the database field that will hold the users password.

The cookie options are held in the next section. First, enter a name for the Cookie that will be used to store the user's username, once their details have been inserted into the database. Next, you need to select the options for the expire time of the Cookie, which controls how long the cookie will last before it expires. These parameters can be one of the options shown below.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>0</td>
</tr>
</tbody>
</table>

This means the Cookie will expire as soon as the user has closed their browser.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>Number of days e.g. 30</td>
</tr>
</tbody>
</table>

This means the Cookie will expire in 30 days.
This means the cookie will expire at 12:00:00 on the 15th March 2003.

In the final section, the first option to set is the minimum length of the user’s password. If you don’t want to impose a minimum password length, you can set this option to 0. Otherwise, set this to 6 for example, which means the users password must be 6 characters or more, otherwise they will be shown the general error message.

Next, you can set the “General Error Message”, this is a message which is shown to the user if there is an error with their details, such as the passwords not matching, or being less than the minimum password length.

You can then set a message which is shown to the user, if their desired username is already in use, so you can ask them to try a different username.

Lastly, you can set the page that the user is redirected to, once everything is correct and their details are inserted into the database.

It’s suggested that you redirect to another page, where the user can enter their name and address details etc if needed. You can then insert these details into the correct database record by using an SQL Update operation, using the users username stored in a Cookie to update the correct record.

Finally, click “OK” to apply the Server Behavior.

To change any of the parameters at any time, double click on the Server Behavior name in the Dreamweaver MX “Server Behaviors” tab, and you can then change a parameters and reapply the extension.
PHPLSC09 – User Change Password

PHPLSC09 is used after a user has logged in, and allows you to create a page so that the user can change their password.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form</td>
<td>form1</td>
</tr>
<tr>
<td>Original Password Field</td>
<td>originalPassword</td>
</tr>
<tr>
<td>New Password Field 1</td>
<td>newPassword1</td>
</tr>
<tr>
<td>New Password Field 2</td>
<td>newPassword2</td>
</tr>
<tr>
<td>DB Connection</td>
<td>phpls</td>
</tr>
<tr>
<td>DB Table</td>
<td>pusers</td>
</tr>
<tr>
<td>DB UserName Field</td>
<td>username</td>
</tr>
<tr>
<td>DB Password Field</td>
<td>password</td>
</tr>
<tr>
<td>Minimum Password Length</td>
<td>6</td>
</tr>
<tr>
<td>User Name Cookie</td>
<td>userCookie</td>
</tr>
<tr>
<td>General Error Message</td>
<td>Password Error, Try Again</td>
</tr>
<tr>
<td>Redirect To</td>
<td>password_changed.php</td>
</tr>
</tbody>
</table>

Figure 22 – PHPLSC09 Dialog Box
Before you apply the extension, you need to add a form to your page similar to the one below:

![Figure 23 – Example Change Password Form](image)

This form has 3 fields, one called `originalPassword` that is used for the user to enter their original password. Next, there are two fields called `newPassword1` and `newPassword2`. These allow the user to enter their new password, and then to enter it again to confirm it to make sure it has been entered correctly.

Once you have created the form, you need to save the page before you apply the Server Behavior.

The first section of the Server Behavior allows you to tell the Server Behavior which form you want to work with. You can then select the form field in which the username is entered, and the two password fields for the new and confirmation password. If you don't want to use a confirmation password, set the password and confirmation passwords to the same password form field.

Next, you need to select the Database connection to use, and the database table that you use to store users details. Next select the field in your database table that will store the user’s username, and the field that stores the user’s password.

In the last section, you can set the minimum length that the user’s password is required to be. If you don't want to impose a minimum password length, then you can set this to 0.

Next, select the Cookie that contains the user’s username. This will have been set with another behavior when the user logged in.

You can then set a “General Error Message”, which the user will see if they type their original password incorrectly, the new password is less than the required length, or the new password and confirmation passwords do not match.

Finally, you can set the page that the user is redirected to once their password has been changed.

Again, you can reopen the Server Behavior at any time, and change any of the parameters and then reapply the behavior.

This completes the normal Cookie Server Behaviors, in the next section we look at the Server Behaviors that work with Cookies and Encrypted Passwords.
Cookies and Encrypted Passwords

In this section, we look at a number of Server Behaviors that are new to version 2 of the PHP Login Suite, and allow you to work with Encrypted Passwords.

![Figure 24 – Cookie Server Behaviors for Encrypted Passwords](image)

Note that you can only use the Encrypted Passwords Server Behaviors if you have encrypted passwords stored in your database. The Passwords should be encrypted using the SQL Password() function.

If you currently have plaintext passwords stored in your database, i.e. passwords that aren't encrypted, you can use the Server Behavior, PHPLSGE01, which can be found in the “Encrypted Passwords” section of the “General” menu. This Server Behavior can be used to convert plain text passwords into encrypted passwords. You will then need to make sure that you only use the Encrypted Passwords Server Behaviors.
PHPLSCE01 – Login and Set Cookie

PHPLSCE01 is used to log a user in, checking their username and password against details stored in a database table, and if the details match the user is logged in, and a field such as their username can be stored in a cookie. You can then allow only users who have the cookie set to view certain pages. This behavior works with encrypted passwords.

Before you apply the behavior, you need to add a form and two fields, one for the user’s username, and one for their password, and you should then save the page.

Figure 25 – PHPLSCE01 Dialog Box
The first set of options allows you to select the form that you want to work with. The Forms dropdown contains a list of all the forms on your page. Once you have selected a form to use, the Username and Password menus will be populated with a list of fields in the selected form. You can then set username to the field that the user will use to enter their username, and password to the field that the user uses to enter their password.

The next section refers to the database table that you want to use to check the users details against. The DB Connection menu contains a list of all the database connections that you have set up for the current site in Dreamweaver MX, select the database connection that you wish to use.

Once you have selected a database connection to use, the DB Table menu will contain a list of all the tables in the current database. Select the table that holds your user details. Once you have selected the database table to use, you can select the field in the database table that holds Usernames and the field that stores Passwords.

The third section allows you to setup details for the Cookie that’s set when the user logs in, and is used to store the user’s username. The first option allows you to set the Cookie Name, which can be anything you wish.

The next two options, Cookie Expire Time, and Cookie Expire Date, allow you to set how long the cookie will last for, and you can use the following combinations.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>0</td>
</tr>
</tbody>
</table>

This means the Cookie will expire as soon as the user has closed their browser.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>Number of days e.g. 30</td>
</tr>
</tbody>
</table>

This means the Cookie will expire in 30 days.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>hh:mm:ss e.g. 12:00:00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>dd/mm/yy e.g. 15/03/03</td>
</tr>
</tbody>
</table>

This means the cookie will expire at 12:00:00 on the 15th March 2003.

The last section allows you to set an Error Message that will be displayed to the user if their username or password is not found in the database.

You also need to enter a page that the user is Redirected To if their login attempt is successful. You can enter an address by hand, or click “Browse” to select another page on your site, and the link will be entered automatically.

Once you have filled in all the options, you can click “OK” and the required code will be added to your page.
You can go back at any time by double clicking on the Server Behavior name in the Dreamweaver MX Application Panel.

Also note that when you apply the Server Behavior, a Session Variable icon is created in the Dreamweaver MX Bindings tab, which can be used to access the value stored in the Cookie on your page, as shown below:

Note that you can click the “Help” button at anytime, to see the Server Behaviors Help file. When the user has logged in, their username will be stored in the Cookie.
PHPLSCE02 – Login and Set Level

PHPLSCE02 is used to log a user in, checking their username and password against details stored in a database table, and if the details match the user is logged in, and their username is stored in one cookie, and an Access Level stored in another cookie. This behavior works with encrypted passwords. The Access Level needs to be numerical e.g. 1, 2 or 3 etc. The Dialog Box for this extension is shown below:

![Figure 27 – PHPLSCE02 Dialog Box](image.png)
First select the form on your page that you wish to work with, and select the form field that will be used for the user’s username, and the form field that will hold the user’s password.

Next, select the Database Connection that you wish to work with, and then select a table from the database that holds user information. You then need to select the database field that contains the username, the field that contains the user’s password, and finally the field that holds the users access level.

The next section allows you to create the cookies you wish to use. This time there are two cookies that are set, one to hold the users username, and the other cookie to hold the users access level.

First, enter the names for the two Cookies that will store the username and access level. The next set of options for Cookie Expire Time, and Cookie Expire Date, allow you to set how long the cookies last before they expire. You can use the following combinations.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>0</td>
</tr>
</tbody>
</table>

This means the Cookie will expire as soon as the user has closed their browser.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>Number of days e.g. 30</td>
</tr>
</tbody>
</table>

This means the Cookie will expire in 30 days.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>hh:mm:ss e.g. 12:00:00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>dd/mm/yy e.g. 15/03/03</td>
</tr>
</tbody>
</table>

This means the cookie will expire at 12:00:00 on the 15th March 2003.

Finally, you can set a Message that is displayed if the user could not be logged in, for example telling them their username or password were incorrect, and to try again.

You can then click “Browse”, or manually type in the page that the user will be redirected to once a successful login is made.

Click OK to apply the Server Behavior, and the relevant code will be generated and added to your page.

Once again, the two Cookies that were created are available on the Dreamweaver MX Bindings Tab for use on your page.

After the behavior has been applied, you can double click the server behavior name in the Dreamweaver MX “Server Behaviors” window, and the dialog box will be displayed with your existing parameters. You can then change the parameters and reapply the extension, and it will modify the generated code accordingly.
PHPLSCE03 – Check Username & Insert

PHPLSCE03 is used to check that a new user's username is unique before inserting it into the database. It also checks that the password is greater than a minimum length that you can specify, and that the password field and confirm password fields match. This Server Behavior works with encrypted passwords.

You can use this behavior to check that a user's username is unique, if it is unique, the user's username and password are inserted into the database, and their username is stored in a cookie. You can then use another page to fill in the rest of the user's details, and update the user's record using their username stored in a cookie.
Figure 28 – PHPLSCE03 Dialog Box

First, create a form with a username field and two password fields. In this example the username field is named *username* and the two password fields are named *password1* and *password2*. You create two password fields, so that the user has to type the password twice, to make sure they have typed the correct password, as they can't see what they have typed because data entered into a password field is hidden.

![Figure 29 – Example Form](image)

Figure 29 – Example Form

Once you have the form created as above, you can then save the page, and then apply the extension.

The first section of the dialog box allows you to select the form that you are using to gather the users username and password. Once you have selected a form, you can select the form field that holds the users username. Next, you can select the form field that contains the user’s password, and a second password field to hold the password again as confirmation. If you don't want to use have an extra password field to confirm the password, you can set both options to the same password field.

The next section concerns the database used to store the user’s information. First select a database connection to use to connect to your database. Next, you can select the table that will contain user's information. You can then select the database field that will contain the user’s username, and the database field that will hold the users password.

The cookie options are held in the next section. First, enter a name for the Cookie that will be used to store the user’s username, once their details have been inserted into the database. Next, you need to select the options for the expire time of the Cookie, which controls how long the cookie will last before it expires. These parameters can be one of the options shown below.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>0</td>
</tr>
</tbody>
</table>

This means the Cookie will expire as soon as the user has closed their browser.

<table>
<thead>
<tr>
<th>Cookie Expire Time:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Expire Date:</td>
<td>Number of days e.g. 30</td>
</tr>
</tbody>
</table>

This means the Cookie will expire in 30 days.
Cookie Expire Time: hh:mm:ss e.g. **12:00:00**  
Cookie Expire Date: dd/mm/yy e.g. **15/03/03**

This means the cookie will expire at 12:00:00 on the 15th March 2003.

In the final section, the first option to set is the minimum length of the user's password. If you don't want to impose a minimum password length, you can set this option to 0. Otherwise, set this to 6 for example, which means the users password must be 6 characters or more, otherwise they will be shown the general error message.

Next, you can set the “General Error Message”, this is a message which is shown to the user if there is an error with their details, such as the passwords not matching, or being less than the minimum password length.

You can then set a message which is shown to the user, if their desired username is already in use, so you can ask them to try a different username.

Lastly, you can set the page that the user is redirected to, once everything is correct and their details are inserted into the database.

It’s suggested that you redirect to another page, where the user can enter their name and address details etc if needed. You can then insert these details into the correct database record by using an SQL Update operation, using the users username stored in a Cookie to update the correct record.

Finally, click “OK” to apply the Server Behavior.

To change any of the parameters at any time, double click on the Server Behavior name in the Dreamweaver MX “Server Behaviors” tab, and you can then change a parameters and reapply the extension.
PHPLSCE04 – User Change Password

PHPLSCE04 is used after a user has logged in, and allows you to create a page so that the user can change their password. This behavior works with encrypted passwords.

Figure 30 – PHPLSCE04 Dialog Box
Before you apply the extension, you need to add a form to your page similar to the one below:

![Figure 31 – Example Change Password Form](image)

This form has 3 fields, one called `originalPassword` that is used for the user to enter their original password. Next, there are two fields called `newPassword1` and `newPassword2`. These allow the user to enter their new password, and then to enter it again to confirm it to make sure it has been entered correctly.

Once you have created the form, you need to save the page before you apply the Server Behavior.

The first section of the Server Behavior allows you to tell the Server Behavior which form you want to work with. You can then select the form field in which the username is entered, and the two password fields for the new and confirmation password. If you don't want to use a confirmation password, set the password and confirmation passwords to the same password form field.

Next, you need to select the Database connection to use, and the database table that you use to store users details. Next select the field in your database table that will store the user's username, and the field that stores the user's password.

In the last section, you can set the minimum length that the user's password is required to be. If you don't want to impose a minimum password length, then you can set this to 0.

Next, select the Cookie that contains the user's username. This will have been set with another behavior when the user logged in.

You can then set a “General Error Message”, which the user will see if they type their original password incorrectly, the new password is less than the required length, or the new password and confirmation passwords do not match.

Finally, you can set the page that the user is redirected to once their password has been changed.

Again, you can reopen the Server Behavior at any time, and change any of the parameters and then reapply the behavior.

This completes the Encrypted Cookie Server Behaviors, in the next section we look at the Server Behaviors that work with Sessions.
Server Behaviors – Session Variables

The Server Behaviors are divided into two types, allowing you to work with either Cookies or Session Variables to store the logged in user. In this section we look at the Cookie Server Behaviors that are available.

When to use Session Variables

Session Variables are normally used for short term storage, as Session Variables expire between visits, so the user has to log in each time they go to the site.

Plain Text Password or Encrypted Passwords

In the new login suite, there are now two sections containing two types of Server Behaviors for working with Session Variables, shown in the screenshot below.

There are two sets of Server Behaviors, PHPLS01 – PHPLS11 work with plaintext, or are not password specific. PHPLSSE01 – PHPLSSE04 work with encrypted passwords. You must have your passwords stored encrypted in your database using the SQL PASSWORD() function to use these Behaviors.

There is a Server Behavior in the general folder, PHPLSGE01, which can be used by a database administrator to convert plaintext passwords to encrypted passwords.

Problems with Session Variables

When PHP is installed, especially on Windows, Session Variables are not enabled to work by default. If you have problems with Session Variables working on your server, read the Troubleshooting Guide at the end of this manual, which explains how to get session variables working.
PHPLS01 – Login and Set Session

PHPLS01 allows you to check a user’s username and password against those stored in a database table, and if the details are correct, the user is logged in by placing their username in a Session Variable.

Before you apply the extension, you need to first add a form to your page, with two fields, one to allow the user to enter their username, and the other field for them to enter their password. You should then save the page.
The first section of the Dialog Box allows you to select the Form that you wish to work with. Once you have chosen a form, you can select the fields that will hold the users username and password.

In the next section, you first need to select a Dreamweaver MX database connection to work with. Once you have selected a database connection, the table's dropdown will be populated with all the tables that exist in your database. Select the table that is used to hold users details. You then need to select the fields that store the username and password.

Finally, you can choose a name for the Session Variable that will be used to store the user's username. “Error Message” allows you to enter your own custom message that will be displayed if the username or password are incorrect. Finally, you can either enter by hand, or press the “Browse” button and select a page that the user will be redirected to if they have logged in successfully.

Once you have applied the Server Behavior, you can go back and change any of the parameters at any time, by double clicking the Server Behavior name in the Dreamweaver MX “Server Behaviors” tab.

After the Server Behavior has been applied, a session is added to the Dreamweaver MX “Bindings” tab, which represents the session you created, and is shown in the screenshot below.

![Figure 34 – Dreamweaver MX Bindings Tab](image)

This Session Variable can be dragged on to your page, and when the page is viewed in a browser, the contents of the Session Variable will be displayed.
PHPLS02 – Login and Set Level

PHPLS02 allows you to check a user's username and password are correct, and if they are, to log the user in and place the user's username and access level in two Session Variables.

Figure 35 – PHPLS02 Dialog Box
The first section of the dialog box allows you to select the form to use. Once you have selected a form, you can set the form field into which the user will enter their username, and the form field into which the user enters their password.

Next, select the Dreamweaver MX Database Connection that you wish to use. You can then select the table in the database that holds your users details. Finally, select the fields in the database table that hold the username, password and access level for your users.

You can then enter a name for the Session Variable that will hold the users username, and a name for the Session Variable that’s used to store the user’s access level.

Next, you can enter a custom message that is displayed to the user if the user’s username or password is incorrect.

The final option allows you to specify a page that the user is redirected to if their login is successful.

Again, icons representing the Session Variables are added to the Dreamweaver MX “Bindings” tab.

To change the parameters for the extension at any time, you can double click on the extension name in the Dreamweaver MX “Server Behaviors” tab to reopen the extension dialog box. You can then change the parameters and reapply the extension.
PHPLS03 – Access by Session

PHPLS03 is used to secure pages, and ensures that only users who have logged in can view your page. This Server Behavior needs to be applied to every page that you want to be protected.

![PHPLS03 Dialog Box](image)

This Server Behavior is very easy to use. You simply select the Session Variable that is set with the user's username at login, and a page which the user is redirected to if they are not logged in.

Only people who have logged in correctly will be able to view the page. This Server Behavior must be applied to every page that you want to be protected.

You can change the parameters of the extension at any time by double clicking the Server Behavior name in the Dreamweaver MX “Server Behaviors” tab. You can then change the parameters and reapply the extension.
PHPLS04 – Access by Level

PHPLS04 allows you to only allow certain users to log in, defined by a numerical access level that was stored in a session when the user logged in. It allows you to setup a rule that limits only certain access levels to be able to view the page.

First, select the Session Variable that holds the users access level, which was set when they logged in.

Next, using the Operator and Access level to set a rule which defines who is NOT allowed to view your page. For example, with the settings above, if the user’s access level is less than 2, then the users will not be allowed to see the page.

Finally, you can set the page that users who aren’t allowed to see the page are redirected to.

The parameters can be changed at any time by double clicking on the extension name in the Dreamweaver MX “Server Behaviors” tab. You can then change the parameters and reapply the Server Behavior.
PHPLS05 – Enable Sessions

This Server Behavior is used to add the PHP code to your page that is necessary for session variables to be used. This code is shown below:

```php
<?php
// Buzz inet PHPLS05 - Enable Sessions
session_start();
?>
```

All of the Server Behaviors in the PHP Login Suite that use sessions automatically include this in their code. It only needs to be applied to your own pages where you wish to use Session Variables.

For example, if you create a new page, and drag one of the Session Variables from the Dreamweaver MX “Bindings Menu” onto your page, you will also need to apply this Server Behavior to your page for the Session Variable to work.

This Server Behavior has no dialog box, as soon as it's selected it applied the code to your page.
PHPLS06 – User Details

PHPLS06 allows you to retrieve a user's record from the database, using their username held in a Session which is set when the user logs in.

First, select a Dreamweaver MX Database connection from the “DB Connection: “ menu. Next, select the table that in which users details are stored. Finally, select the username field in your table, and the session variable that contains the username that was set when the user logged in.

When the page is viewed in a browser, the username in the Session Variable specified is compared to the username field in the database, so that the correct record for the user is selected.

This Server Behavior when applied creates a new Dreamweaver MX Recordset, as well as adding the required code to your page, as shown in figure 39 below.

This allows you to drag any of the fields from the Recordset straight onto your page, or the fields can be used to create a dynamic form.
Note that if you remove this Server Behavior from your page, you will see the following message from Dreamweaver MX.

![Dreamweaver MX Alert](image)

This message is shown because the Server Behavior is linked to the recordset that it created. When you remove the Server Behavior, the recordset will be removed as well. As a result, this warning can be safely ignored, and you can click the “OK” button.

You can change the parameters at any time by double clicking the Server Behavior name in the Dreamweaver MX “Server Behaviors” tab, and the dialog box will open. You can then change the desired parameters and reapply the extension.
PHPLS07 – Kill a Session

PHPLS07 allows you to kill a session variable, which effectively logs the user out.

![PHPLS07 Dialog Box](image1)

*Figure 41 – PHPLS07 Dialog Box*

This Server Behavior is very simple, as you only need to select the Session Variable that you want to kill.

When this Server Behavior has been applied, note that the name of the session being killed is shown in the Server Behavior name, as shown in the screenshot below.

![Dreamweaver MX Server Behaviors tab](image2)

*Figure 42 – Dreamweaver MX Server Behaviors tab*
PHPLS08 – Kill all Sessions

PHPLS08 allows you to call all Session Variables that are currently active. This is useful when a username session variable and an access level variable are set.

This Server Behavior has no dialog box. You can tell the extension has been applied by checking the Dreamweaver MX Server Behaviors tab, as shown in the screenshot below.

![Figure 43 – Dreamweaver MX Server Behaviors tab](image-url)
PHPLS09 – Insert Record

PHPLSC09 is used on new User Forms, to add new users to a database. First you need to apply the Dreamweaver MX “Insert Record” Server Behavior to the page, which will insert the user's details into the database. This Server Behavior is then applied, which logs the user in, by storing their username in a session variable.

![PHPLS09 - Insert Login](image)

*Figure 44 – PHPLS09 Dialog Box*

First, select the form that contains the data you want to use. Next select the field on the form that will hold the users username, so it can be placed in a session.

Finally, enter a name for the Session Variable that will be used to hold the username.

When the page is run, after the Dreamweaver MX Insert record code has run, the user’s username that they entered will be placed in a session variable.

When the Server Behavior is applied, a new Session Variable is created in the Dreamweaver MX “Bindings” tab, which you can drag and drop onto a future page.

You can change the parameters at any time by double clicking on the Server Behavior name in the Dreamweaver MX “Server Behavior” tab.
PHPLS10 – Check Username and Insert

PHPLS10 is a new Server Behavior. It allows you to check a username is unique before inserting it in the database. The username is then stored in a Session Variable, so that you have a link to the new users record. You can then on another page, use the Dreamweaver MX Insert Server Behavior to fill in the rest of their details, such as their name and address if required. It also checks the users password is entered correctly, and that it's the same as, or longer than a password length set by the administrator.
Before you apply the behavior, you need to add a form to your page. Next add a form field so the user can enter their username, a password field in which they can enter their desired password, and finally a confirm password field if you want to ensure they type their password correctly. An example form is shown in the screenshot below:

You will then need to save the page before you apply the Server Behavior.

The first section allows you to select the form that you want to work with. You can specify the form fields that will contain the user’s username, password and a field for the user to confirm the password. This makes sure that the user’s password is typed correctly, as with a password box they can’t see which characters they’ve typed, but their unlikely to make a mistake twice. If you don't want to use a confirmation password field, then just set the two password menus to the same password field.

The next section allows you to first specify the Dreamweaver MX database connection to work with, and to set the table that the users details will be entered into. You can then specify the fields that will hold the users username and password.

Next you can set the minimum length the user’s password should be. If you don't want to impose this restriction, set this value to 0.

You then need to enter a name for the Session Variable that will hold the users username once the details have been inserted into the database.

The “General Message” option allows you to create a custom message that will be displayed to the user if the password and confirm password don’t match, or the passwords are under the minimum length.

The next option allows you to create a custom message that will be displayed if the user’s desired username is already taken, and asking them to select another.

Finally, you can enter or browse to the page the user will be redirected to, once their details have been inserted into the database.

You can change the parameters this Server Behavior uses at any time, by double clicking the Server Behavior name in the Dreamweaver MX “Server Behaviors” tab. You can then change the parameter, and reapply the extension.
PHPLS11 – User Change Password

PHPLS11 allows you to create a page that allows the user to change their password. To do this, the user needs to be logged in, so that their username is stored in a Session Variable.

![PHPLS11 Dialog Box](image)

*Figure 47 – PHPLS11 Dialog Box*
Before you apply the behavior, you need to add a form to the page, with 3 form fields, as shown below.

![Figure 48 – Example Form](image)

The form needs three fields, one for the user to enter their original password, and then two fields for the new password and to allow the user to confirm the new password to make sure that they have entered it correctly. Save the page, and you can then apply the extension.

The first section of the extension dialog box allows you to select the form on your page that you wish to use to get the users new password details. Once you have selected the form, you can select the form field that holds the users original password, the form field that holds the users new password, and a field to confirm the new password to make sure it has been entered correctly. If you don't wish to use a field to confirm the user's password, then you can set the extension options to the same password for both menus.

Next you can select the Dreamweaver MX Database Connection that you wish to use wish to use. You can then select the table in your database that holds the users details. Next, select the field in the database table in which the users username is stored, and the field in which the user’s password is stored.

You can then complete the final section of the dialog box. The first option allows you to set the minimum length that the user's new password can be. If you don't wish to impose this restriction, then you can set the minimum password value to 0. You can then select the Session Variable that holds the users username, and was set when the user logged in. The next option allows you to set a custom error message that is displayed if the users password is incorrect, the new password is less than the required minimum length, or the new password and confirm password do not match.

Finally, you can enter or browse to the page that the user is redirected to once their password has been changed.

Again, like all the extensions in the PHP Login suite, you can change the parameters at any time by double clicking the extension name in the Dreamweaver MX “Server Behaviors” tab to re open the dialog box. You can then change the parameters and reapply the extension.
Sessions and Encrypted Passwords

In this section, we look at a number of Server Behaviors that are new to version 2 of the PHP Login Suite, and allow you to work with Encrypted Passwords.

![Server Behaviors for Session Variables and Encrypted Passwords]

Note that you can only use the Encrypted Passwords Server Behaviors if you have encrypted passwords stored in your database. The Passwords should be encrypted using the SQL `Password()` function.

If you currently have plaintext passwords stored in your database, i.e. passwords that aren't encrypted, you can use the Server Behavior, `PHPLSGE01`, which can be found in the "Encrypted Passwords" section of the "General" menu. This Server Behavior can be used to convert plain text passwords into encrypted passwords. You will then need to make sure that you only use the Encrypted Passwords Server Behaviors.
PHPLSSE01 – Login and Set Session

PHPLSSE01 allows you to check a user’s username and password against those stored in a database table, and if the details are correct, the user is logged in by placing their username in a Session Variable. This behavior works with encrypted passwords.

Figure 50 – PHPLSSE01 Dialog Box

Before you apply the extension, you need to first add a form to your page, with two fields, one to allow the user to enter their username, and the other field for them to enter their password. You should then save the page.
The first section of the Dialog Box allows you to select the Form that you wish to work with. Once you have chosen a form, you can select the fields that will hold the users username and password.

In the next section, you first need to select a Dreamweaver MX database connection to work with. Once you have selected a database connection, the table’s dropdown will be populated with all the tables that exist in your database. Select the table that is used to hold users details. You then need to select the fields that store the username and password.

Finally, you can choose a name for the Session Variable that will be used to store the user’s username. “Error Message” allows you to enter your own custom message that will be displayed if the username or password are incorrect. Finally, you can either enter by hand, or press the “Browse” button and select a page that the user will be redirected to if they have logged in successfully.

Once you have applied the Server Behavior, you can go back and change any of the parameters at any time, by double clicking the Server Behavior name in the Dreamweaver MX “Server Behaviors” tab.

After the Server Behavior has been applied, a session is added to the Dreamweaver MX “Bindings” tab, which represents the session you created, and is shown in the screenshot below.

![Image of Dreamweaver MX Bindings Tab]

Figure 51 – Dreamweaver MX Bindings Tab

This Session Variable can be dragged on to your page, and when the page is viewed in a browser, the contents of the Session Variable will be displayed.
PHPLSSE02 – Login and Set Level

PHPLSSE02 allows you to check a user's username and password are correct, and if they are to log the user in and place the user's username and access level in two Session Variables. This Server Behavior works with encrypted passwords.

![PHPLSSE02 Dialog Box]

*Figure 52 – PHPLSSE02 Dialog Box*
The first section of the dialog box allows you to select the form to use. Once you have selected a form, you can set the form field into which the user will enter their username, and the form field into which the user enters their password.

Next, select the Dreamweaver MX Database Connection that you wish to use. You can then select the table in the database that holds your users details. Finally, select the fields in the database table that hold the username, password and access level for your users.

You can then enter a name for the Session Variable that will hold the users username, and a name for the Session Variable that’s used to store the user’s access level.

Next, you can enter a custom message that is displayed to the user if the user’s username or password is incorrect.

The final option allows you to specify a page that the user is redirected to if their login is successful.

Again, icons representing the Session Variables are added to the Dreamweaver MX “Bindings” tab.

To change the parameters for the extension at any time, you can double click on the extension name in the Dreamweaver MX “Server Behaviors” tab to reopen the extension dialog box. You can then change the parameters and reapply the extension.
PHPLSSE03 – Check Username & Insert

PHPLSSE03 is used to check that a new user's username is unique before inserting it into the database. It also checks that the password is greater than a minimum length that you can specify, and that the password field and confirm password fields match. This Server Behavior works with encrypted passwords.

You can use this behavior to check that a user's username is unique, if it is unique, the user's username and password are inserted into the database, and their username is stored in a cookie. You can then use another page to fill in the rest of the user's details, and update the user's record using their username stored in a cookie.

![PHPLSSE03 - Check Username and Insert](image)

- Form: form1
- Form User Field: username
- Form Password Field 1: password1
- Form Password Field 2: password2
- DB Connection: phpls
- DB Table: eusers
- DB Username Field: username
- DB Password Field: password
- Minimum Password Length: 6
- Session User Name: userSession
- General Error Message: Password Details Incorrect
- Username Not Unique Message: Username is not Unique
- Redirect To: phplsse03.php

PHP Login Suite © Buzz inet Ltd 2003
First, create a form with a username field and two password fields. In this example the username field is named *username* and the two password fields are named *password1* and *password2*. You create two password fields, so that the user has type the password twice, to make sure they have typed the correct password, as they can't see what they have typed because data entered into a password field is hidden.

![Example Form](image)

Once you have the form created as above, you can then save the page, and then apply the extension.

The first section of the dialog box allows you to select the form that you are using to gather the users username and password. Once you have selected a form, you can select the form field that holds the users username. Next, you can select the form field that contains the user’s password, and a second password field to hold the password again as confirmation. If you don’t want to use have an extra password field to confirm the password, you can set both options to the same password field.

The next section concerns the database used to store the user’s information. First select a database connection to use to connect to your database. Next, you can select the table that will contain user's information. You can then select the database field that will contain the user’s username, and the database field that will hold the users password.

In the final section, the first option to set is the minimum length of the user’s password. If you don't want to impose a minimum password length, you can set this option to 0. Otherwise, set this to 6 for example, which means the users password must be 6 characters or more, otherwise they will be shown the general error message.

You can then choose the Session Variable that will be used to store the user’s user name.

Next, you can set the “General Error Message”, this is a message which is shown to the user if there is an error with their details, such as the passwords not matching, or being less than the minimum password length.

You can then set a message which is shown to the user, if their desired username is already in use, so you can ask them to try a different username.
Lastly, you can set the page that the user is redirected to, once everything is correct and their details are inserted into the database.

It's suggested that you redirect to another page, where the user can enter their name and address details etc if needed. You can then insert these details into the correct database record by using an SQL Update operation, using the users username stored in a Cookie to update the correct record.

Finally, click “OK” to apply the Server Behavior.

To change any of the parameters at any time, double click on the Server Behavior name in the Dreamweaver MX “Server Behaviors” tab, and you can then change a parameters and reapply the extension.
PHPLSSE04 – User Change Password

PHPLSSE04 is used after a user has logged in, and allows you to create a page so that the user can change their password. This behavior works with encrypted passwords.

![PHPLSSE04 Dialog Box](image)

*Figure 55 – PHPLSSE04 Dialog Box*
Before you apply the extension, you need to add a form to your page similar to the one below:

![Figure 56 – Example Change Password Form](image)

This form has 3 fields, one called `originalPassword` that is used for the user to enter their original password. Next, there are two fields called `newPassword1` and `newPassword2`. These allow the user to enter their new password, and then to enter it again to confirm it to make sure it has been entered correctly.

Once you have created the form, you need to save the page before you apply the Server Behavior.

The first section of the Server Behavior allows you to tell the Server Behavior which form you want to work with. You can then select the form field in which the username is entered, and the two password fields for the new and confirmation password. If you don’t want to use a confirmation password, set the password and confirmation passwords to the same password form field.

Next, you need to select the Database connection to use, and the database table that you use to store users details. Next select the field in your database table that will store the user’s username, and the field that stores the user’s password.

In the last section, you can set the minimum length that the user’s password is required to be. If you don’t want to impose a minimum password length, then you can set this to 0.

Next, select the Session Variable that contains the user’s username. This will have been set with another behavior when the user logged in.

You can then set a “General Error Message”, which the user will see if they type their original password incorrectly, the new password is less than the required length, or the new password and confirmation passwords do not match.

Finally, you can set the page that the user is redirected to once their password has been changed.

Again, you can reopen the Server Behavior at any time, and change any of the parameters and then reapply the behavior.
PHP Login Suite – General Behaviors

In the General section of the PHP Login Suite, there are a number of Server Behaviors which don't fit into the other categories.

In this section, there are also a number of Behaviors new to version 2 of the PHP Login Suite.
PHPLSG01 – Turn off Notices

This Server Behavior is used if you see messages similar to the ones shown below, when using the PHP Login Suite.

Warning: Undefined index: action in C:\Inetpub\wwwroot\PHPLogin\login.php on line 25

Warning: Undefined variable: errorMessage in C:\Inetpub\wwwroot\PHPLogin\login.php on line 52

These messages are classed as Notices in PHP, and only occur when the PHP error reporting level is set to E_ALL, which means to display all Errors, Warnings and Notices.

The Notices occur because the variables have not been specifically defined, which is not important, and does not stop the code from working.

If you get the messages shown above, you can apply this Server Behavior to any affected page, and it will stop the notices from being displayed.

Note that you can stop the messages appearing permanently, by editing you php.ini file and changing the following line:

```
error_reporting = E_ALL;
```

To

```
error_reporting = E_ALL & ~E_NOTICE
```

You can then save your php.ini file, and restart your Web Server.

This will turn off notices on a permanent basis, and will stop the messages from appearing.
PHPLSG02 – PHP Info

This server behavior allows you to quickly and easily add the `phpinfo()` command to your page, which when viewed in a browser will show information about your PHP setup.

To apply this behavior, first place the cursor at the position on the page in which you want the command to be inserted. You can then apply the behavior.

When the page is viewed in a browser, you will see the PHP info screen displayed, as shown in the following screenshot.

![PHP Info Screen](image)

**Figure 58 – Example of the PHP Info Screen**

This screen is extremely useful, as it shows all the PHP settings on your system, and is useful for troubleshooting, and to get a list of the PHP Server Variables that are present on your PHP server, along with their values.
PHPLSG03 – Mail Forgotten Password

This Server Behavior allows you to create a page, that allows the user to enter their email address that they signed up with, and emails their username and password to that address. This is used if the user has forgotten their details.

![PHPLSG03 Dialog Box](image)

*Figure 59 – PHPLSG03 Dialog Box*
Before you apply the extension, you need to create a form such as the one shown below. You should then save your page.

![Example Form](image)

*Figure 60 – Example Form*

The first section of the Dialog Box allows you to choose a Database Connection to work with, the database table to use, and the fields in the table that holds the users username, password and email address.

Next, you need to select the form, and form field that the users enters their email address into.

You than then create a custom message that is shown to the user if their email address is not found, and you can enter or browse to a page that the user is redirected to once their user details have been sent.

The final section of the dialog box allows you to enter an email address that appears as the from field in the email that’s sent, as well as the subject for the email, and a custom message. The user’s username and password are automatically added to the bottom of the email.

When the extension is applied, the user can enter the email address they used to sign up with, and their username and password will be sent to that address.
PHPLSGE01 – Convert Plain Text Password to Encrypted Passwords

This Server Behavior is for use by the Site Administrator Only, and needs to be applied carefully.

This Server Behavior creates code that will work through a selected table, read in the plain text passwords, and write them back to the table in an encrypted form.

You MUST make sure that the page created is only run ONCE. If the page is run more than once, the encrypted password will be read and encrypted again, effectively changing the user’s password to a random value. To make sure the page is only run once, the behavior is started by submitting a form, so that you can easily control the behavior.

It’s recommended that you take a backup of your database table before applying this extension.

Before you start, add a Form to your page. There is no need for any form fields, just add an ordinary form and save the page. You can then apply the Server Behavior.

![PHPLSGE01 Dialog Box]

First, select the Dreamweaver MX Database Connection that you wish to use. You can then select the table in your database that holds the users details that need converting. Next, select a form field that is unique for each user, allowing the Server Behavior to...
change the correct record. This can be the username field if you don't allow duplicated usernames. Next select the password field that contains the plaintext password that needs converting to an encrypted password, and the form on your page that the extension can use.

When the Server Behavior is applied, the page will be changed so that it looks like the screenshot below:

![Figure 62 – Example of Page](image)

You can now run the page in a browser, and click the “Convert Passwords” button to start the conversion process.

Make sure that you only click the button once.

Once the conversion process has finished, all plaintext passwords will be converted to encrypted passwords, and can be used with the encrypted passwords Server Behaviors.
PHPLSGE02 – Reset Password

This Server Behavior is used to create a page that allows a user to reset their encrypted password. If the user forgets their password, you can't email their details to them, as there is no way to get the original password from their encrypted password.

To solve this situation, if the user forgets their password, the password is reset to a password created at random. This is then emailed to the user, and they can use it to log in and change their password to a password of their choice.

Figure 63 – PHPLSGE02 Dialog Box
Before this Server Behavior is applied, you need to add a form to your page, with a field where the user can enter their email address, as shown below:

![Figure 64 – Example Form](image)

You can then save your page, and apply the Server Behavior.

The first section of the dialog box allows you to set the Dreamweaver MX Database connection to work with, and the table in the database that holds your users details. You can then set the fields which hold the users username, password and email address.

Next you can select the form on your page that you wish to use, and the fieldname of the form field that the user enters their email address into.

You can then set a custom message that is displayed to the user if their email address is not found, and the page that the user is redirected to if their details are successfully sent.

Finally, you can set the length of the random password generated for the user, the email address from which the email is sent, the email subject, and a custom message that is included in the body of the email.

The user’s username and password are added to the bottom of this message when the email is sent.
Special Techniques

Redirecting a user to a certain page when they log in depending on their Access Level.

One question that was frequently asked with the original login suite, was how to redirect users to a specific page when they log in, which depends on their access level.

Imagine, for this example that you have 4 levels of Access, 1,2,3 and 4. When the user logs in, you want them to be redirected to a page matching their access level.

First, you need to create 4 pages, one for each access level. Start with the page which requires the highest access level i.e. 4.

Save this page as access4.php. You can then apply PHPLS04 – Access by Level (sessions) or PHPLSC04 – Access by Level (cookies), depending on whether you use sessions or cookies to store the user’s access level.

You can then set a rule such as:

![PHPLS04 - Access by Level](image)

**Figure 65 – Access Level 4 Example**

This means that any user with an access level LESS THAN 4 will be sent to the page access3.php instead of being allowed to see the page.

Next, create a page called access3.php, and apply the behavior again, as shown below:
This means that any user with an access level LESS THAN 3 will be redirected to the page access2.php, instead of being allowed to view the page.

Create two more pages, access2.php and access1.php, with rules similar to the above, so only the appropriate access level can view the page.

Once you have created all four access level pages, you can create the login page.

Save the page as login.php, and apply the Server Behavior PHPLS02 or PHPLSC02 – Login and Set Level, depending on where you are using session variables or cookies to store the user’s access level, as shown in figure 67 below.

Redirect the user to the Highest Access Level Page, which in this example is access4.php.

If, for example the user’s access level is 2 when they log in, the following process will happen.

The user is first redirected to page access4.php. As they do not have an access level of 4, they are redirected to the next page, access3.php.

On access3.php, again they do not have the required access level, so they will be redirected to the next page, access2.php.

On access2.php, their user level meets the requirements specified, and they are allowed to view the page.

By cascading down from the highest access level to the lowest access level, the user will stop on the page that matches their access level, and they will be allowed to see the page.
Although this may sound like a slow process, in practice the redirections happen instantly, and the user will not realise they have been sent from one page to another, instead they will just see the page load that matches their access level.
Troubleshooting

In this last section of the manual, we look at a few of the problems frequently encountered by users, and how to solve them.

**Notices**

This section explains what to do if you receive messages on your page similar to the ones shown below:

```
Warning: Undefined index: action in C:\Inetpub\wwwroot\PHPLogin\login.php on line 25
Warning: Undefined variable: errorMessage in C:\Inetpub\wwwroot\PHPLogin\login.php on line 52
```

These messages are classed as Notices in PHP, and only occur when the PHP error reporting level is set to E_ALL, which means to display all Errors, Warnings and Notices.

The Notices occur because the variables have not been specifically defined, which is not important, and does not stop the code from working.

If you get the messages shown above, refer to the section of this manual for Server Behavior PHPLSG01 – Turn off Notices, which will stop these notices from being shown.

Note that you can stop the messages appearing permanently, by editing your php.ini file and changing the following line:

```
error_reporting = E_ALL;
```

to

```
error_reporting = E_ALL & ~E_NOTICE
```

You can then save your php.ini file, and restart your Web Server.

This will turn off notices on a permanent basis, and will stop the messages from appearing.
Error: Cannot add header information - headers already sent

This error is very common when you use the PHP header command. It occurs when data is output to the browser, before the header command which redirects users to another page for example.

Usually the data that is sent is difficult to see on the page, as usually it consists of spaces. If a single line break of space is sent to the browser before the header command, it will cause the command to fail.

The way to stop the error is to go through your code, and remove any spaces or line breaks which are outside of PHP tags.

Everything inside the PHP tags is fine, as long as it doesn’t output anything using an echo command for example.

Once anything that is output to the browser before the header command is deleted, the code will work as intended.

An example of code that will create this error message is shown in the screenshot below:

```php
<?php
session_start();

if($_SESSION['userSession'] == "Fred"){
    header("Location: admin.php");
}

<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
</body>
</html>
```

Figure 68 – Example Code that causes the Error

The spaces in the code above, shown in black, need to removed as they are sent to the users browser before the PHP header() command, which will cause the header() command to fail. As soon as the spaces are removed, as shown in the screenshot below, the code should work correctly.
Figure 69 – Example code after Correction

In the code above, you can see the spaces between the PHP tags have been removed, and the code will now work correctly, without causing a header error.
Session Variable Problems

Problems with Session Variables are fairly common with new users, especially those on a Windows machine running PHP. Session data is actually stored in files on the server by PHP, and these files are stored in a temporary directory which is specified in the PHP configuration file, php.ini.

Unfortunately, the path specified by default in the php.ini file is not automatically created during installation, and this stops Session Variables from working, as PHP has nowhere to store the temporary data.

Testing Session Variables on your System

To check that Session Variables are working on your system, create a new PHP page, and add the following code:

```php
<?php
session_start();
session_register("testSession");
$HTTP_SESSION_VARS['testSession'] = "Session Variables are Working!";
?>
<html>
<head>
<title>Set Session</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
Session Variable has been set
</body>
</html>
```

Save this as `setsession.php`. This page sets a session variable called "testSession". Next, create another PHP page, and add the following code:

```php
<?php
session_start();
?>
<html>
<head>
<title>Show Session</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<?php
echo $HTTP_SESSION_VARS['testSession'];
?>
</body>
```
Save this page as readsession.php.

First, open the page setsession.php in your browser, which will cause the session variable to be set. Once the page has loaded, you can then open readsession.php, and if session variables are working on your system, you should see the message “Session Variables are Working!”.

**If Sessions are NOT working on your System**

If the code above shows that Session Variables aren't working on your system, then you can enable them using the instructions below.

First, create a directory on your Server which PHP can use to store the session information. It makes sense to place this directory in your PHP installation directory, and to name this directory, temp, or something similar. So, for example, if you installed PHP to c:\php, then the location of the new directory will be:

```
c:\php\temp
```

Next, open your php.ini file, which on windows will probably be in your windows directory at the following location:

```
c:\windows\php.ini
```

Once you have opened php.ini in a text editor, scroll down until you find the following setting:

```
session.save_path = \tmp
```

The path here needs to be changed to the directory you created above, for example:

```
session.save_path = C:\php\temp
```

Once you have made this change, save php.ini, and you will then need to restart your web server to read in the new settings.

You can then test the Session Variables again, using the code shown above, and you should find that sessions are now working.

Although the above instructions should enable Session Variables, if you still have problems your next stop should be the Session Variable section of the online PHP manual at:

```
```
There are a number of online PHP resources available, which we look at in this section.

**The PHP Login Suite Newsgroup**

This newsgroup is private, and only for people who have purchased the PHP login suite, and is becoming a nice community for discussing questions relating to the PHP login suite, and PHP code in general.

Many queries that you may have will have already been asked, and answered, in this group, so it's worth downloading all the messages in the group and having a browse through, as the group contains a lot of useful information.

Feel free to use the group to ask questions, whether they're about the PHP login suite or PHP in general.

The News Server address is:

```
nntp.buzzinet.co.uk
```

Note that when you set up a news account for the above server, you need to specify that your news reader sends authorization details when it logs on, and enter the username and password you received when you purchased the PHP Login Suite.

**Dreamweaver MX Support Website**

The Dreamweaver MX support website, is run by myself, and is not connected to Macromedia, and can be found at the following address:

```
http://www.dreamweavermxsupport.com/index.php
```

This website contains a large number of tutorials, FAQ’s, and is becoming a fast growing resource for Dreamweaver MX related information, with a strong bias towards the PHP server model.

**phploginsuite.dreamweavermxsupport.com**

The above is a new section of the Dreamweaver MX Support website, that I’ve set up with the introduction of the second version of the PHP Login Suite.

This website is private, and is only for registered users of the PHP login suite. You will need to log in to this website, which you can do with the same username and password that you use to connect to the PHP Login Suite Newsgroup.

This website has been set up, to allow PHP Login Suite users to easily download the latest version of the PHP Login Suite.
PHP.net

http://www.php.net/

PHP.net is the home of PHP, and contains a huge amount of PHP related information. You can download the latest version of PHP, or browse through the online PHP manual, which is an extremely useful resource.

The online PHP manual pages also allow user comments at the bottom, and these are well worth reading for extra information, as many of the common PHP problems and solutions are discussed, and there are a number of useful PHP code examples.