Red Hat Secure Web Server

Getting Started Guide

Red Hat Software, Inc.
Research Triangle Park, North Carolina

Revision: SecServ-1.0-Print-RHS (3/98)

Red Hat Software, Inc.
4201 Research Commons, Suite 100
79 T. W. Alexander Drive
P. O. Box 13588
Research Triangle Park, NC 27709
(919) 547-0012
`redhat@redhat.com`
`http://www.redhat.com`

Contents

Introduction

This guide is meant to get you started running your Red Hat Secure Web Server. It is not meant as complete documentation for any of the programs included with this package. However, this guide will point you to the appropriate places for more in-depth documentation where necessary.

This guide will show you how to install the programs included, as well as how to configure your Apache web server. You will also be taken through the steps necessary to get a secure certificate and install it on your server.

After reading and following the steps in this guide, your secure server will be running using a test certificate. If you've followed our instructions for requesting a certificate from the certificate authority of your choice, you'll be ready for secure e-commerce as soon as your certificate arrives.

We Need Feedback!

If you've found a mistake in this manual, or if you've thought of a way to make it better, we'd love to hear from you! Please send mail to:

`docs@redhat.com`

Be sure to mention the manual's identifier:

SecServ-1.0-Print-RHS (3/98)

That way we'll know exactly which version of this manual you have. If you have a suggestion, try to be as specific as possible when describing it. If you've found an error, please include the section number and some of the surrounding text so we can find it easily. We may not be able to respond to every message sent to us, but you can be sure that we'll be reading them all.

1

Installing Your Apache Server

After reading this chapter and following the steps, your web server will be installed and configured. You'll also see how to start your web server and run it without security to test your installation.

Please Note: In order to install the Red Hat Secure Web Server, you *must* already have obtained and be running Red Hat Linux. Red Hat Linux is not included with the Red Hat Secure Web Server product.

1.1 OS and Software Versions

If you are using Red Hat Linux version 3.0.3 or earlier, you are strongly urged to upgrade your system to a more recent version (preferably 5.x) before installing the Red Hat Secure Web Server. If you don't upgrade, the installation won't be handled using RPM. This will make it a lot harder to remove the secure server software at a later time. In addition, the C libraries on your system will also be significantly older than the libraries used to build the server. You may run into unexpected problems because of this.

When you run the installer, you may see a message warning you that your RPM version is old and asking whether you want to upgrade. You should choose to upgrade. If you do not, the installer will use `cpio` to perform the install, making it difficult to uninstall later. Upgrading RPM shouldn't adversely affect your system.

You may also see a message about glibc being too old. You should also allow the installer to upgrade this package. If you don't, the rest of the installation may not go smoothly.

1.2 Mounting the CD-ROM

To begin the installation process, you must first mount the CD-ROM. You must be root to run this command. Type the following command to mount the installation CD:

```
mount -t iso9660 /dev/cdrom /mnt/cdrom
```

Please Note: If you allow users to mount the CD-ROM drive by adding the `user` option the `/etc/fstab` entry, you'll also need to add the `exec` option or the installation program will not work. In other words:

```
/dev/cdrom /mnt/cdrom iso9660 user,noauto,ro 0 0
```

should be changed to:

```
/dev/cdrom /mnt/cdrom iso9660 user,noauto,ro,exec
0 0
```

(All on one line, of course.)

1.3 Optional Packages

When you run the installer, you will be asked what additional packages you would like to install. The following sections give a brief description of each package, as well as the location of the packages' documentation and configuration files (after the installation is complete). Before you start the installation, please review the following packages, and decide which ones you want.

1.3.1 Apache Perl Module

Configuration File: N/A

Documentation: `/usr/doc/mod_perl-1.08/` or the commands:

```
perldoc mod_perl
perldoc Apache
```

Description: mod_perl is an Apache module that integrates Perl and Apache. It allows you to access Apache's C API through Perl, as well as providing a faster alternative to CGI. For more information on this module, refer to section 3.1.

1.3.2 Apache PHP/FI Module

Configuration File: N/A

Documentation: `/usr/doc/mod_php-2.01/`

Description: PHP/FI is a scripting language that is embedded in HTML files. This module allows Apache to handle this embedded language on your web site. It is often used as a faster alternative to CGI programming. Refer to section 3.2 for more information on this module.

1.3.3 Squid

Configuration File: `/etc/squid.conf`

Documentation: `/usr/doc/squid-1.1.21/`

Description: Squid is a caching proxy server. Using squid, you can set your web browsers (this really isn't as useful unless you have multiple people using it) to use your machine as a proxy server. Squid will cache any requests so if you access a site more than once, it will be much faster. When you install squid you can choose whether to install the memory caching version (if you have memory to spare) or the disk caching version (described below). See section 3.3 for more information on configuring Squid.

1.3.4 Squid-novm

Configuration File: `/etc/squid.conf`

Documentation: `/usr/doc/squid-novm-1.1.21/`

Description: Squid-novm is the same as the Squid package except it doesn't use RAM to hold the cache. Instead it uses your disk drive.

1.3.5 Analog

Configuration File: `/etc/analog.cfg`

Documentation: `/usr/docs/analog-2.11/`

Description: Analog is a log parsing program that helps you gain useful statistics from your web server log files. If you choose to install the optional web interface, we'll see how to configure it in section 3.4.

1.3.6 ht://Dig

Configuration File: `/etc/htdig.conf`

Documentation: `/usr/doc/htdig-3.0.8b2/`

Description: ht://Dig is an indexing and search engine meant for smaller websites. See section 3.5 for more information on configuring ht://Dig.

1.3.7 Netscape Navigator

Configuration File: N/A

Documentation: `http://help.netscape.com/`

Description: Netscape Navigator is a popular web browser. Extensive help is available via the 'Help' menu or from Netscape's web site.

1.3.8 Netscape Communicator

Configuration File: N/A

Documentation: `http://help.netscape.com/`

Description: Netscape Communicator is based on Netscape Navigator, but includes additional features, such as a mail and news reader. Extensive help is available via the 'Help' menu or from Netscape's web site.

1.4 Running the Installer

Now you're ready to install your secure server. Change directory to your CD-ROM's mount point:

```
cd /mnt/cdrom
```

And run the installer:

```
./install-webserver
```

For best results, you should run the installer from the console, or from a color xterm (xterm-color). Next, follow these steps to complete the installation process:

1. **Upgrade Required Software**
   If you have older versions of RPM or glibc, you will be notified and asked to upgrade. You should choose to upgrade.
2. **Selecting Packages**
   Now you must select the packages you wish to install. Your screen will appear as in figure 1.1. Use the arrow keys to move between the packages and press the space bar to select or deselect a package. Tab to the "OK" button when you are finished and press Enter.
   (a) **If you selected squid**
   If you chose to install squid, the dialog box in figure 1.2 will appear. If you have a lot of memory available, you should choose `squid` so the cache will be in memory. If you don't, then chose `squid-novm` so the cache will be on disk.
   (b) **If you selected analog**
   If you chose to install analog, the dialog box in figure 1.3 will appear. This allows you to install Analog's web-based interface. This gives you a nice interface from which to run analog.

Figure 1.1: Optional Package Selection

Figure 1.2: Squid Package Options

Figure 1.3: Analog Package Options

3. **Continue with the Install**
   The installer will now show you how much disk space the entire installation will take (see figure 1.4). If it is too much space, select `No`, and re-run the installation selecting fewer optional packages. Otherwise, select `Yes`, and press Enter.

4. **Installing Packages**
   You will now see a series of progress bars similar to figure 1.5. This display will change as each package is installed. When the install completes, you will see a screen similar to figure 1.6. Press Enter to return to your shell prompt. You can now start to configure your new secure web server.

1.5 Configuring Your Server

You shouldn't need to change any of the default configuration settings for your server, but you should know about a few of them. You can't start your server now, because by default it looks for the security files you'll be creating in chapter 2. If you don't want to wait that long, and want to see your server working now, follow the steps in section 1.5.3.

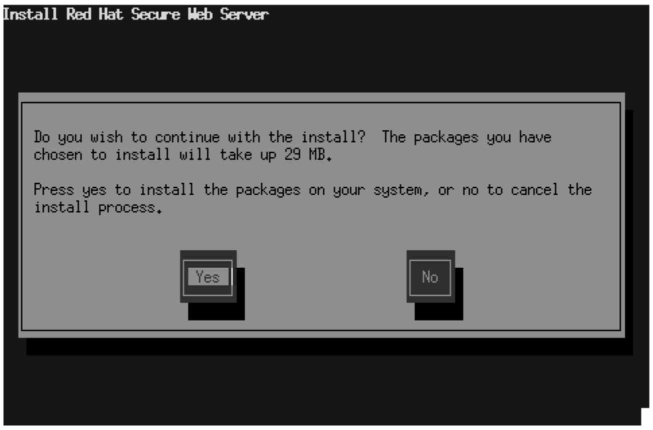

Figure 1.4: Continue with Installation

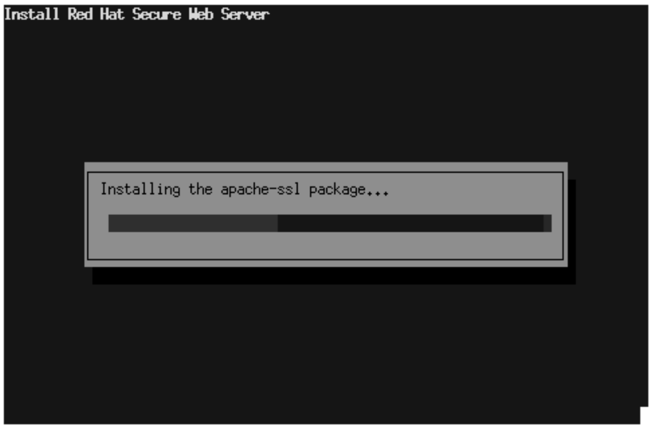

Figure 1.5: Installation Status Bar

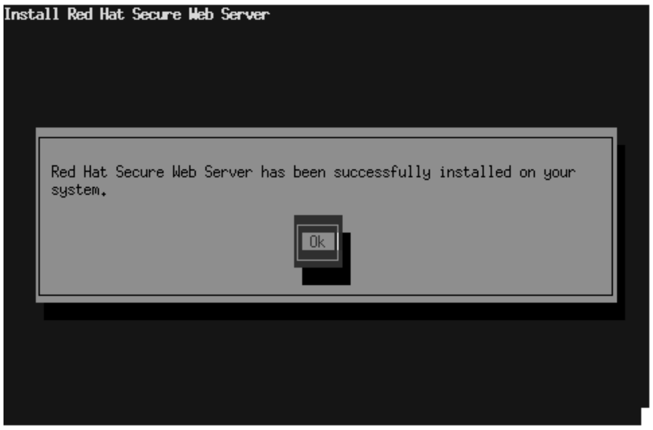

Figure 1.6: Installation Complete

Once you get your server running (either with SSL turned off, or after you get a certificate) you can access the full Apache server documentation at `http://machine.domain.com/manual/`. Look there for a full list of Apache's configuration options.

1.5.1 Apache Configuration Options

The default location for your server configuration files is `/etc/httpsd/conf/`. There you'll find the following files:

The httpd.conf File

This is the main configuration file. It determines how your server will run and how logs are kept. The more important options in this file are:

HostnameLookups This option can be set to `on` or `off`. It determines whether your server will look up the hostnames of people connecting to your web site. It is generally a better idea to leave this set to `off`, because it may slow down your server to do a lookup for each connection.

User This option sets the user that will be running the server process. It determines what access the server will have while it's running. Any HTML files that aren't accessible to this user will not be accessible to your web site's visitors. It also determines the owner of any CGI processes that the server spawns. The default is `nobody`.

Group This is similar to the `User` setting, except it sets the group of the server process. The default is `nobody`.

ServerAdmin This option defines the email address of the server's administrator. It will appear by default on error pages from your server, so that people can email the administrator when problems happen. It defaults to `root@localhost`, but you may want to change it to your webmaster's email address.

There are also a few options that you don't really need to change, but the information they contain is important.

ErrorLog This option defines where any server errors are logged. The default location is `/var/log/httpsd/error_log`. By default, all server logs will be in `/var/log/httpsd/`.

TransferLog This option defines where accesses to your server will be logged. You will need this file if you want to generate any server statistics (with Analog, for instance). The default is `/var/log/httpsd/access_log`.

The srm.conf File

This configuration file contains information about the documents and programs running on your server. The most important options are:

DocumentRoot This option defines where the server will look for HTML files. The default is `/home/httpsd/html`.

UserDir This option defines the name of the sub-directory within each users home directory where they can place personal HTML files. By default, this is `public_html` so if someone accesses

http://www.domain.com/~username/, they will receive the default file in
/home/username/public_html (assuming /home/username/ is username's home directory).

DirectoryIndex This option defines the default page in a directory. The default is index.html index.shtml index.cgi. This means that the server will try to find any one of these three files, and will return the first one it finds.

Alias This option allows you to create a directory that can be accessed from your web server, but isn't within the DocumentRoot. By default, there's an icons directory that you can access by requesting /icons/. This directory is in /home/httpsd/icons/ instead of /home/httpsd/html/icons/.

ScriptAlias This option allows you to define directories where CGI programs can run. You generally don't want these directories within the DocumentRoot so people won't be able to see how your scripts work. The default directory can be accessed at /cgi-bin/, and is in /home/httpsd/cgi-bin/.

The access.conf File

This file contains information about restricting access to your web pages. You won't need to change any of the default settings to get started.

1.5.2 Starting and Stopping Your Server

During the installation process, a server startup file was installed into /etc/rc.d/init.d, and named httpsd. You can manually stop, start and restart the server by running this command, and passing what you want to do as the argument. For instance, if you've already changed directory into /etc/rc.d/init.d, you can stop and start your server by issuing the following commands:

```
./httpsd stop
./httpsd start
```

If you want to have your server start when your machine boots, you should make a symlink to this file from the `rc.3` and `rc.0` directories. The following commands will do this for you:

```
cd /etc/rc.d/rc3.d
ln -s ../init.d/httpsd S99httpsd
cd /etc/rc.d/rc0.d
ln -s ../init.d/httpsd K99httpsd
```

Note that if you are running X, you can use `tksysv` to do this graphically.

1.5.3 Running Without Security

If you want to get your server running before getting or creating a digital certificate, you need to change two settings in the `httpd.conf` file.

Uncomment the line that reads:

```
#SSLDisable
```

so that it reads:

```
SSLDisable
```

The next step isn't strictly necessary, but it really should be done. The default port for a non-secure server is 80, so you should change the line:

```
Port 443
```

to:

```
Port 80
```

Now save the file and start your server. You can now access it at `http://machine.domain.com/`.

2

Securing Your Server

If you want to engage in electronic commerce using your web site, you may not get a lot of business without first securing your server. Having a secure server instructs any web browser accessing your site to encrypt all the data passing back and forth, so that the data won't be intelligible to a third party. Your Apache-SSL web server allows you to conduct secure transactions so your customers will feel more comfortable when making purchases from your site.

This chapter will show you how to create a secure certificate and get it signed by a Certificate Authority (CA). The two CAs discussed will be VeriSign (`www.verisign.com`) and Thawte (`www.thawte.com`). Once you have your certificate we will show you how to install it on your Apache-SSL server.

In addition to ensuring confidentiality, secure servers also guarantee the identity of the organization running the server. If the certificate says the site is Red Hat Software's, and the user trusts the CA, then there is no doubt that any files or programs downloaded from that site really are from Red Hat Software, Incorporated.

Secure servers aren't just for e-commerce; you may also use a secure server to transmit sensitive data such as sales figures to sales people on the road or to your business partners over your extranet.

2.1 How Server Security Works

When a web browser connects to a secure site, it requests that the server authenticate itself. The authentication process involves private keys, public keys, and a digital certificate. The certificate is digitally signed by a CA. Before signing the certificate, the CA has verified that the organization requesting the certificate is who they claim to be. This means that anyone accessing the site can be sure of who they are connecting to.

The process of getting a certificate is relatively easy, and we'll go into more detail later, but here is a quick overview. You first create an encryption key, and then a certificate request based on that key. The certificate request contains information about your server and the company hosting it. You send this certificate request, along with documents proving your identity, to a CA. Once the CA is satisfied that you are indeed who you claim to be, you will be sent the digital certificate. You can then install this certificate on your web server, and begin handling secure transactions.

Most web browsers that support SSL have a list of CAs whose certificates they will accept. Any certificates authorized by a CA that isn't in this list will be shown to the user, who can then choose to accept or decline them.

2.2 Deciding on a Certificate Authority

There isn't a whole lot of difference between the various certificate authorities. Your decision may be based on past experience, or the experiences of your friends or colleagues. It may also be a purely monetary decision. This guide will discuss using VeriSign or Thawte as your CA. At the time this was written, VeriSign charges $349 for the first server digital certificate you request, and $249 for each additional server. Thawte charges $125 for each server.

Please Note: Through a special arrangement with Thawte, purchasers of the Red Hat Secure Web Server boxed set can obtain signed certificates for only $100 per server. Instructions for obtaining this discount can be found in section 2.6.2.

If you wish to use VeriSign, you can simply use your Dun & Bradstreet (D-U-N-S) Number to prove you have the right to use your organization's name. If you do not have this number, you can request one from their website at www.dnb.com. You can also provide any of the following documentation to VeriSign to verify your company information:

1. Articles of Incorporation
2. Partnership Papers
3. Business License
4. Fictitious Business License
5. Federal Tax ID Confirmation

Thawte requires the following three documents to prove your identity:

1. Letter of Authorization
2. Proof of Organizational Name
3. Proof of right to use Domain Name

See section 2.6 for more information on D-U-N-S numbers and these documents, as well as complete steps for requesting a certificate from these certificate authorities.

2.3 Creating Your Key and Certificate Request

To create your certificate request, you need to use the SSLeay program. This program has been installed with your secure server.

2.3.1 Generating a Key

First you need to generate an encryption key for use when creating your certificate request. To do this, you need to generate a random key. You can use the special system file `/dev/urandom` to achieve the randomness desired. Type the following command to generate a 1024 bit key encrypted with the triple-DES cipher:[1]

```
ssleay genrsa -des3 -rand /dev/urandom 1024 >
httpsd.key
```

(All on one line, of course.)

You will see something similar to:

```
1049776 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
..........................................+++++
..............+++++
e is 65537 (0x10001)
Enter PEM pass phrase:
```

Now you should select a password to encrypt the key with.[2] You will need to remember this password every time you start your secure web server. This password will not be printed on the screen, so you will be asked to verify the password to make sure you didn't make any typos entering it. You will then have a file called `httpsd.key` which contains your key. This file should be owned by the root user on your system and not accessible to any other user. You should also keep a backup of this file

[1] If you see an error message such as "No such file or directory" then you need to make sure you have `/usr/sbin` in your `PATH` environment variable. You can also just type `/usr/sbin/ssleay` plus the rest of the command.

[2] Your password should be at least 8 characters, and should include numbers or punctuation to make it harder to crack. Your password is case sensitive.

because if you lose it, you will have to request a new digital certificate and pay all over again.

If you don't want to have to type in a password every time you start your web server, and you're absolutely certain that no untrusted eyes will ever be able to see your key file, you can use the same command as above without the `-des3` option and your key won't be encrypted. It is generally not a good idea to do this, however.

2.3.2 Generating a Certificate Request

Now you must generate your certificate request. This is what you will send to the CA of your choice so that they can sign your certificate. Enter the following command:

```
ssleay req -new -key httpsd.key > httpsd.csr
```

You will see the following output and questions that you must answer. The default answers to the questions will appear in brackets (`[]`) You can simply press enter if you want to use the default for your answer. Make sure that you use the name of the server as it will be accessed and not any aliases that it may have (i.e. you may wish to have your secure server named `secure.mydomain.com` instead of `www.mydomain.com` even though it may be a virtual host on the same machine). You just need to make sure that any links to your secure server reference the name you put here. When it asks for the password, type in the password you selected when generating your key. You don't need to worry about any of the optional settings, so hit Enter to accept the blank default.

```
Using configuration from /etc/ssl/lib/ssleay.cnf
Enter PEM pass phrase:
You are about to be asked to enter information
that will be incorporated into your certificate
request.
```

```
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave
some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name []:NC
Locality (City) Name []:Research Triangle Park
Company (Organization) Name []:Test Company
Department Name []:Information Systems
Server Host Name []:www.mydomain.com
Administrators E-mail address []:admin@mydomain.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

You will now have your certificate request in a file named `httpsd.csr` ready to send to your CA. In the next section we'll see how to get a test certificate from VeriSign and Thawte, as well as how to create one yourself.

2.4 Getting a Test Certificate

Both VeriSign and Thawte provide the ability to request a temporary signed certificate from them for testing purposes. VeriSign's test certificate lasts for 2 weeks while Thawte's lasts for 30 days. You can also create your own test certificate using the SSLeay library.

2.4.1 Creating a Test Certificate On Your Own

You can create your own test certificate by running the following command:

```
ssleay req -new -x509 -key httpsd.key > httpsd.crt
```

You will see the following output with questions that you must answer:

```
Using configuration from /etc/ssl/lib/ssleay.cnf
Enter PEM pass phrase:
You are about to be asked to enter information
that will be incorporated into your certificate
request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave
some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name []:NC
Locality (City) Name []:Research Triangle Park
Company (Organization) Name []:Test Company
Department Name []:Information Systems
Server Host Name []:www.mydomain.com
Administrators E-mail address []:admin@mydomain.com
```

You will now have a certificate in the file named `httpsd.crt` that you can use to test your secure server.

See section 2.5 to install and test this certificate.

2.4.2 Getting a Test Certificate from VeriSign

To request a test certificate from VeriSign go to `http://digitalid.verisign.com/test_server_ids.html` and follow these steps:

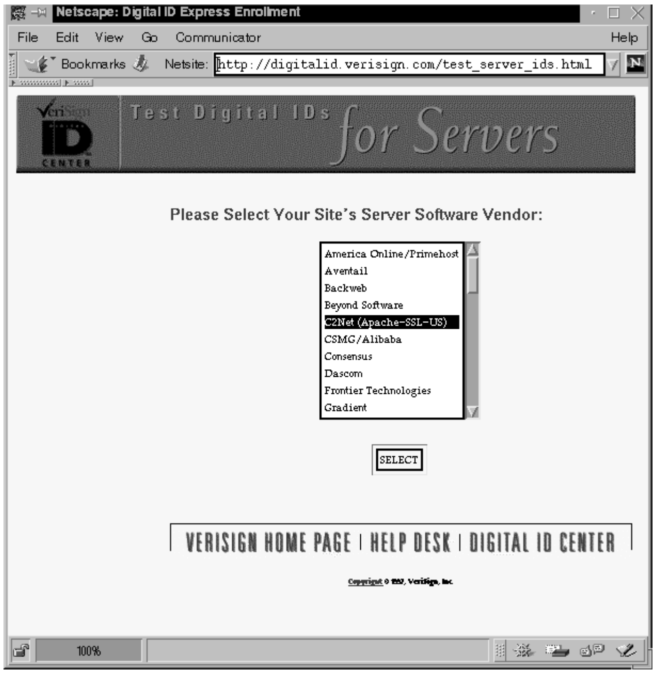

Figure 2.1: VeriSign's Server Selection Page

1. Select C2Net (Apache-SSL-US) from the selection box and click "SELECT" as in figure 2.1.

2. You will see a page explaining the differences between the test certificate and a real certificate. Click the "BEGIN" button at the bottom of the page.

3. The next page will ask for the certificate request that you generated. Cut and paste the `httpsd.csr` file you've created and then click "CONTINUE" as in figure 2.2.

4. The next page will allow you to verify the information you placed into the certificate request. You will also need to provide contact information. Then click the "ACCEPT" button if you accept the test certificate agreement as in figure 2.3.

Figure 2.2: VeriSign's CSR Page.

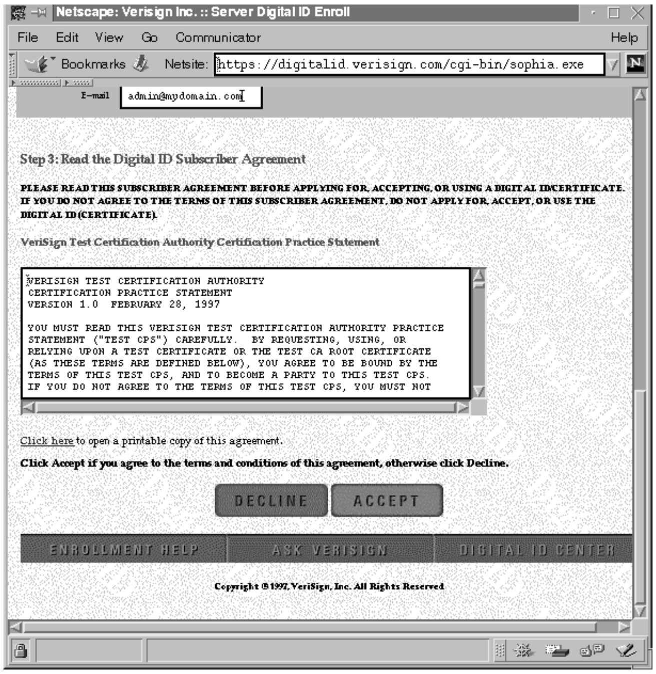

Figure 2.3: VeriSign's Final Verification Page

5. You will be emailed the test certificate very soon. You should save this certificate in a file named `httpsd.crt`.

VeriSign also has steps to install a Test Certificate Authority certificate in your browser, so when it accesses a site using a test certificate, the certificate will be accepted by the browser. If you're using Netscape, you will be given the option to accept the test certificate, so this step isn't absolutely necessary unless you're using an old version of Netscape.

See section 2.5 to install and test this certificate.

2.4.3 Getting a Test Certificate from Thawte

To request a test certificate from Thawte you need to go to `https://www.thawte.com/cgi/server/test.exe` (figure 2.4) and follow these steps:

1. Cut and paste your certificate request into the "Certificate Signing Request" portion of the page.
2. Select "Use most basic format" in the Certificate Format Options section of the page.
3. Click the "Generate Test Certificate" button.
4. The return page will include your test certificate (figure 2.5) which you can then cut and paste into a file named `httpsd.crt`.

See section 2.5 to install and test this certificate.

2.5 Installing and Testing Your Certificate

You should now have a file named `httpsd.crt` containing your test certificate. Copy the `httpsd.key` and `httpsd.crt` files to the directory `/etc/httpsd/conf/`, which is the default location for your Apache-SSL

Figure 2.4: Thawte's Test Certificate Page

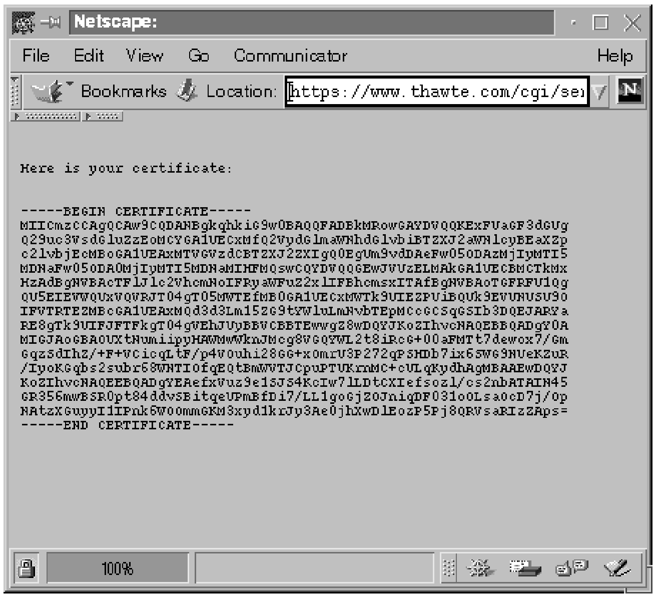

Figure 2.5: Thawte's Test Certificate Return Page

Figure 2.6: Netscape's New Site Certificate Dialog Box

server. If you changed the location or names of the files in your Apache config files, you should name and place the files appropriately.

Now start your Apache server. If your key file is encrypted, you will be asked for the password. Your server will now be running.

Access your server's home page. You should see a dialog box as in figure 2.6.

Follow the instructions to accept the test certificate (you can just accept the defaults and click "Next" until the dialogs have finished). You can then browse your site securely. The default home page will look like figure 2.7.

2.6 Buying a Certificate

Now you're ready to purchase a certificate. Once you've received it, simply follow the steps in section 2.5 to install your new certificate.

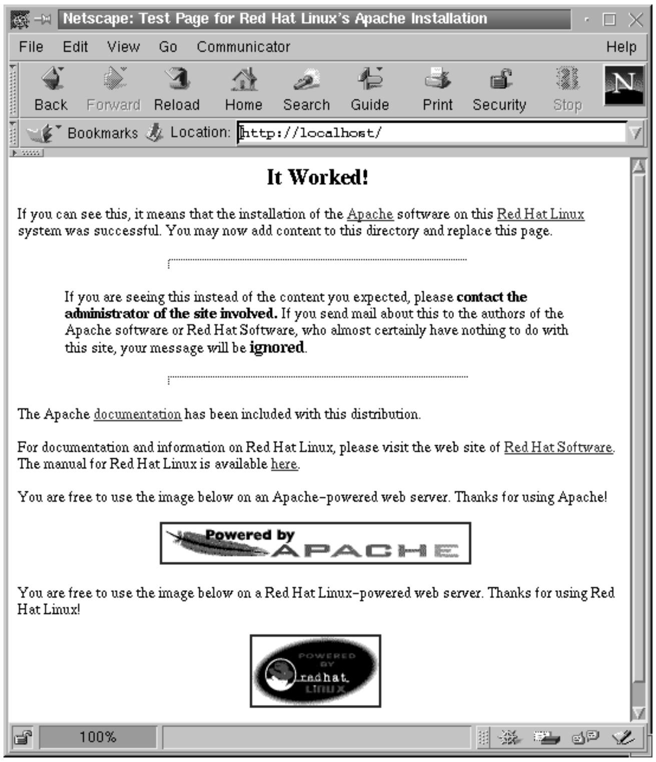

Figure 2.7: The Red Hat Secure Web Server Default Home Page

2.6.1 Purchasing a Certificate From VeriSign

Purchasing a Certificate from VeriSign is similar to requesting the free test certificate. You start out by going to `http://digitalid.verisign.com/server_ids.html` and then following these steps:

1. Select C2Net (Apache-SSL-US) from the selection box and click "SELECT". This page is very similar to figure 2.1.

2. You will then see a page explaining the process of purchasing a certificate. Click "BEGIN" to continue.

3. The first page is where you cut and paste the certificate request you created (which should be in a file named `httpsd.csr`) and click "CONTINUE". This page is very similar to figure 2.2.

4. The following page includes 5 steps for you to complete:

   (a) Double check the information placed into the certificate request.

   (b) Select the software vendor (C2Net (Apache-SSL-US)) from the list and select a challenge phrase. This is a security check in case VeriSign needs to contact you. You must also enter contact information for the certificate. The technical contact is the person who will be installing the certificate. The organizational contact is the person who can speak for the company requesting the certificate. The billing contact is the person who will be paying the bill for the certificate.

   (c) Now you must select how you will pay for your certificate. First select what type of certificate you are getting (whether it's a renewal or a new certificate) then how you will pay. If you use a credit card you will save $20 off the price.

   (d) Enter your D-U-N-S number now. If you don't have this number you can fax your articles of incorporation, business license or other charter documents to VeriSign. Your certificate will be processed faster if you have a D-U-N-S number however. See section 2.6.3 for more information about getting a D-U-N-S number or what documents you should fax.

(e) Click on the “ACCEPT” button to agree to the Digital ID Server Agreement.

5. You will receive a PIN number you can use to check the status of your certificate at any time by visiting `https://digitalid.verisign.com/getstatus.html`.

6. Your request will now be processed and you will be emailed your certificate. Replace the test certificate installed in section 2.5, restart your server, and you’re ready for electronic commerce.

2.6.2 Purchasing a Certificate From Thawte

To purchase a certificate from Thawte, follow these steps:

1. Collect the following three documents to be faxed or mailed to Thawte (see section 2.6.3 for more information on these documents):

   (a) Proof of your right to use the Organization name.

   (b) Proof of your right to use the given domain name.

   (c) An authorization letter.

2. Generate your certificate request. This is the `httpsd.csr` file we created in section 2.3.2.

3. Go to `https://www.thawte.com/cgi/server/step1.exe` and fill out the forms as follows:

   (a) Select “Web Server Certificate” and click “Continue”.

   (b) The next page has quite a bit of information to be added. First, cut and paste your certificate request into the Server CSR section. Then select Apache SSL in the WebServer Software section. Select how you will be paying.

   Please Note: Since Red Hat is a Hosting Partner with Thawte, you will receive $25 off the certificate price, so make sure you enter “redhat” in the Hosting Partner section.

   Now click “Process page 1”.

(c) You will see a page that verifies the information in your certificate request. Make sure this information is correct. Then fill in the "Company and Payment" information as requested. Now click "Process page 2".

(d) The final page includes more information about your company and how you wish to pay. Fill this out and click the "Submit Application" button to finish your request.

4. Fax (or mail) the necessary documents to Thawte. When your application has been processed, you will receive your certificate by email. Replace the certificate used in section 2.5 with this one and restart your Apache-SSL server.

2.6.3 D-U-N-S Numbers and Other Business Documents

If you don't have a D-U-N-S number, you can request one by filling out the form at `http://www.dnb.com/aboutdb/dunsform.htm`. You can also find out from VeriSign if you already have a D-U-N-S number by going to `https://digitalid.verisign.com/dnb_query.htm`.

If you don't wish to get a D-U-N-S number, you can also fax any of the following documents to VeriSign to verify your business identity:

1. Articles of Incorporation
2. Partnership Papers
3. Business License
4. Fictitious Business License
5. Federal Tax ID Confirmation

If you're using Thawte for your CA, you need the following documents. Please visit Thawte's web site to make sure these requirements haven't changed. This list is meant as a general overview and is not guaranteed to be complete.

Proof of Right to Use Organization Name

Different types of organizations require different types of proof:

Corporations, companies, partnerships, proprietorships require a company registration document or the articles of incorporation or partnership declaration.

DBAs and Individuals require a copy of the DBA registration, or any official correspondence with the given name.

Government Departments require an original signed letter from the head of the department, on the appropriate letterhead.

Non-Government Organizations require an original signed letter from the CEO, Chairman, or Managing Director of the Organization, on the appropriate letterhead.

Universities and University Departments require an original signed letter from the Dean or Vice-Chancellor of the department, on the appropriate letterhead.

Special Interest Groups must contact Thawte to discuss verification.

Proof of Right to Use Domain Name

This documentation is not necessary if your domain ends in `.com`, `.edu`, `.org`, `.se`, or `.za`. Otherwise, the following information is needed. Note that this information is provided in the `whois` output of the domain.

1. The domain agency the domain was registered with.
2. The name and contact information for your domain's administrative contact.
3. The name and contact information for your domain's technical contact.

Authorization Letter

This document is a signed letter from a high-ranking individual in the organization. The person must be able to speak for the organization in an official capacity. This is to prevent non-authorized people from requesting an electronic authenticated presence without first going through the proper channels.

3

Configuring Optional Packages

3.1 mod_perl

mod_perl is a large package, and it would be impossible to include all it can do in this guide. This section will explain how to set up mod_perl in its most commonly used role: as a replacement for CGI.

The recommended way to do this is to edit the `srm.conf` file, and add the following directives:

```
Alias /perl/ /home/httpsd/perl/
<Location /perl>
SetHandler perl-script
PerlHandler Apache::Registry
Options +ExecCGI
</Location>
```

Actually, /home/httpsd/perl/ can be any directory where you want to store your Perl scripts. This will be like your cgi-bin directory, so /perl/ is the directory where you'll access the scripts on your server such as http://www.domain.com/perl/.

If you want to have mod_perl handle scripts based on file extension you can add this instead:

```
<Files *.pl>
SetHandler perl-script
PerlHandler Apache::Registry
Options ExecCGI
</Files>
```

You'll need to restart your server for these modifications to take affect:

```
/etc/rc.d/init.d/httpsd restart
```

For more information on using mod_perl as a CGI replacement refer to the cgi_to_mod_perl Perl documentation by running the command:

```
perldoc cgi_to_mod_perl
```

3.2 mod_php

Like mod_perl, mod_php is very large. It is a scripting language in itself. For more information about how to use mod_php, open the file /usr/doc/mod_php-2.0.1/doc.html using your web browser. To configure your Apache server to access the module, you need to add the following line to the srm.conf file:

```
AddType application/x-httpd-php .phtml
```

Now restart your server for this change to take affect:

```
/etc/rc.d/init.d/httpsd restart
```

To test the module, you can create a simple .phtml file called `test.phtml` (in `/home/httpsd/html/`) containing the following text:

```
<?phpinfo()>
```

Now open `http://www.domain.com/test.phtml`. Information about your PHP configuration should now be displayed.

3.3 Configuring Squid

To get started running Squid, you should start with the `QUICKSTART` file in `/usr/docs/squid-1.1.21` (or `/usr/doc/squid-novm-1.1.21`, if you installed squid-novm). For more complete information, you should open the `FAQ.html` file (in the same directory as the `QUICKSTART` file) using your web browser.

You will also need to run the following commands to copy the Squid CGI program to the proper CGI directory:

```
cd /home/httpd/cgi-bin
cp cachemgr.cgi ../../httpsd/cgi-bin
```

To get Squid to start and stop with your machine you should add the following symlinks to your startup directory:

```
cd /etc/rc.d/rc3.d
ln -s ../init.d/squid S99squid
cd /etc/rc.d/rc0.d
ln -s ../init.d/squid K99squid
```

Note that if you're using X, you can also do this by running the `tksysv` command.

The main configuration settings you need to use to get Squid running are outlined below. You'll need to edit the configuration file `/etc/squid.conf`, and set the following parameters:

1. Find the line that begins with:

   ```
   #cache_mem 8
   ```

   Uncomment this line and replace `8` with the amount of RAM (in megabytes) you want to allocate to Squid.

2. Find the line that begins with:

   ```
   #cache_swap 100
   ```

   Uncomment this line and replace `100` with the amount of disk space (in megabytes) you want to allocate to Squid.

3. Find the section with the lines:

   ```
   acl manager proto cache_object
   acl localhost src 127.0.0.1/255.255.255.255
   acl all src 0.0.0.0/0.0.0.0
   ```

Add the following line replacing the Xs with your domain subnet. Make sure your domain's subnet mask is properly entered after the "/":

```
acl allowed_hosts src XXX.XXX.XXX.0/255.255.255.0
```

4. Find the section with the line:

```
http_access allow all
```

Replace it with:

```
http_access allow allowed_hosts
http_access deny all
```

5. Find the section with the line

```
icp_access allow all
```

Replace it with:

```
icp_access allow allowed_hosts
icp_access deny all
```

6. Find the line that begins with:

```
#cache_mgr webmaster
```

Uncomment this line and replace `webmaster` with the email address of the Squid adminstrator.

7. Save the `squid.conf` file.
8. You can now start squid using the start script in `/etc/rc.d/init.d/`:

   ```
   /etc/rc.d/init.d/squid start
   ```

   If this is the first time you've run Squid, it will create the swap directories it needs upon startup.

Squid should now be running on port 3128 of your machine. You can now set up Netscape (from any machine you chose to allow access to Squid) to use your machine as its proxy server. Some people have also changed their routers to forward any requests on port 80 to redirect through Squid; this saves them from having to set up the proxy on all the machines on their network.

3.4 Configuring Analog

Analog is a complex program. Please refer to the file `/usr/doc/analog-2.11/Readme.html` for information on running it. If you installed the form interface, then you will need to copy the files into your server's `DocumentRoot` and `CGI-bin` directory. You will also need to copy a few icon files. The following commands will do this:

```
cd /home/httpd/html
cp analogform.html ../../httpsd/html
cd ../cgi-bin
cp analform.cgi ../../httpsd/cgi-bin
cd ../icons
cp *.gif ../../httpsd/icons
```

Now you need to tell Analog where to find your server's log files. You do this by editing the configuration file `/etc/analog.cfg`. Add the following two lines anywhere in the file:

```
LOGFILE /var/log/httpsd/access_log
ERRLOG /var/log/httpsd/error_log
```

Now you can access the form by pointing your browser to `http://www.domain.com/analogform.html`.

You'll probably want to put this in a separate directory, and restrict access to it using the `access.conf` file (or a `.htaccess` file). You don't want just anyone seeing your site's access log. Please refer to the web server documentation for more information on restricting access to pages or directories on your server.

3.5 Configuring ht://Dig

For complete documentation on configuring and running ht://Dig, open the file `/usr/doc/htdig-3.0.8b2/index.html` using your web browser. You will also need to copy the following files, puting them in your server's `DocumentRoot`:

```
cd /home/httpd/html
cp -R htdig ../../httpsd/html
cd ../cgi-bin
cp htsearch ../../httpsd/cgi-bin
```

To get started you will need to edit the `/etc/htdig.conf` file as follows:

1. Change the line that begins with:

   ```
   start_url: http://htdig.sdsu.edu/
   ```

   to include your domain (such as `http://www.mydomain.com/`.)

2. Save the `htdig.conf` file. You may wish to read through this file and make other changes, but this is the only required change.

3. Type the following commands to set up your search database:

```
htdig
htmerge
```

4. Now go to `http://www.domain.com/htdig/` to search your web site.

5. You may also want to customize the following files to display information specific to your site:

   /home/httpsd/htdig/search.html This is the page that you should link to for your site search. You may wish to make this file a symlink to `index.html` in the same directory.

   /var/lib/htdig/header.html This page will be displayed at the top of any search results.

   /var/lib/htdig/footer.html This page will be displayed at the bottom of any search results.

   /var/lib/htdig/nomatch.html This page will be displayed when no search results have been found.

   /var/lib/htdig/syntax.html This page will be displayed when a syntax error has occurred in a boolean expression.

6. Now all you need to do is make a link to the `search.html` page. If your site changes often, you'll need to set up `htdig` and `htmerge` to run automatically using `cron`. See your system documentation for more information on `cron`.

A

mod_php Functions

This is a list of the functions that are provided in the mod_php package. For more detailed information on the PHP/FI scripting language, please read the documentation file from /usr/doc/mod_php*/.

Abs

Syntax: Abs(variable)

Usage: Abs returns the absolute value of arg.

Parameters: integer,real

Returns: integer

AddSlashes

Syntax: AddSlashes(variable)

Usage: Escapes any $ or ′ (if MAGIC_QUOTES is set) with a backslash.

Parameters: string

Returns: string

See Also: StripSlashes()

ASort

Syntax: ASort(array)

Usage: Sort is used to sort a PHP associative array in ascending order. Use ARSort() for descending order. Unlike the Sort() function, ASort() maintains index-value pairings. It understands the three variable types and will sort alphabetically if the array contains strings, and numerically if the array contains numbers. In the case of an array which contains a mixture of types, the first type in the array will specify the sort method. Note that if you are going to sort a non-associative array, you should use the Sort() function.

Parameters: associative array

Returns: associative array

See Also: Sort()

BinDec

Syntax: BinDec(binary_string)

Usage: BinDec returns the decimal equivalent of the binary number represented by the binary_string argument. The largest number that can be converted is 31 bits long or 4294967295 in decimal.

Parameters: binary string

Returns: decimal

See Also: DecBin()

Ceil

Syntax: Ceil(value)

Usage Ceil() rounds a floating point value up to the next integer. The return value is of type double (floating point) such that it can be used properly in complex equations. To get an integer type back, use: `$new = IntVal(Ceil($value));`.

Parameters: float

Returns: double

See Also: Floor()

ChDir

Syntax: ChDir(dir)

Usage: ChDir changes the current working directory to the directory specified in the argument.

Parameters: string

Returns: boolean

ChGrp

Syntax: ChGrp(file,group)

Usage: ChGrp changes the group id of the specified file.

Parameters: string

Returns: boolean

ChMod

Syntax: ChMod(file,perms)

Usage: ChMod changes the file permissions of the specified file. The perms parameter must be specified in octal notation.

Parameters: string, octal (integer)

Returns: boolean

Example: `ChMod($filename,0755)`

Chop

Syntax: Chop(string)

Usage: Chop removes all trailing whitespaces including new-lines, tabs and spaces and returns the new string.

Parameters: string

Returns: string

Chr

Syntax: Chr(arg)

Usage: Chr returns the ASCII character represented by the integer argument.

Parameters: integer

Returns: character

ClearStack

Syntax: ClearStack()

Usage: The ClearStack() function is a hack/workaround for a deficiency in the PHP parser. This defficiency is that PHP only has a single expression stack. When inside a user-defined function this expression stack is never cleared because its contents may be needed within a complex expression in the context from which the user-defined function was called. This means that if you have a while loop with a lot of iterations inside a user-defined function, you may be chewing up a lot of stack space. You may even be hitting the max data space limit. You can put a call to ClearStack() inside your while loop to prevent this memory consumption, but the trade-off is that you cannot use your function within any sort of context. You have to assign the output of the function to a temporary variable and then use this temporary variable in whatever context you need.

Returns: Stack

ClearStatCache

Syntax: ClearStatCache()

Usage: The stat() system call is normally an expensive operation on most operating systems. In order to make sure that repeated calls to the various File* functions like FilePerms(), FileInode(), etc. The result of the last stat() call is always cached. If one of the File* functions is called again with the same argument, the cached stat() result will be used. In order to force a new stat() system call the ClearStatCache() function may be called to clear the cached stat() result.

closeDir

Syntax: closeDir()

Usage: closeDir closes a directory opened using the openDir function.

CloseLog

Syntax: CloseLog()

Usage: CloseLog() closes the descriptor Syslog() uses to write to the system logger. See the `closelog(3)` UNIX man page for more details.

See Also: Syslog(), OpenLog() and InitSyslog().

Cos

Syntax: Cos(arg)

Usage: Cos returns the cosine of arg in radians. See also Sin() and Tan()

Parameters: float

Returns: float

Count

Syntax: Count(array)

Usage: Count returns the number of items in an array variable. If the variable is not an array, the return value will be 1 (because a normal variable is similar to an array with only one item). If the variable is not defined, the return value will be 0.

Parameters: array

Returns: integer

Crypt

Syntax: Crypt(string,[salt])

Usage: Crypt will encrypt a string using the standard Unix DES encryption method. Arguments are a string to be encrypted and an optional two-character salt string to base the encryption on. See the Unix man page for your crypt function for more information. If you do not have a crypt function on your Unix system, you can use Michael Glad's public domain UFC-Crypt package which was developed in Denmark and hence not restricted by US export laws as long as you ftp it from an non-US site.

Parameters: string, 2 character string

Returns: string

Date

Syntax: Date(format,[time])

Usage: The Date function is used to display times and dates in various ways. The function takes a format string and a time as arguments. If the time argument is left off, the current time and date will be used. The time argument is specified as an integer in number of seconds since the Unix Epoch on Jan.1 1970. The format string is used to indicate which date/time components should be displayed and how they should be formatted. The following characters are recognized within the format string. Any unrecognized character is printed verbosely:

- `Y` - Year eg. 1995
- `y` - Year eg. 95
- `M` - Month eg. Oct
- `m` - Month eg. 10
- `F` - Month eg. October
- `D` - Day eg. Fri

- `l` - Day eg. Friday
- `d` - Day eg. 27
- `z` - Day of the year eg. 299
- `H` - Hours in 24 hour format eg. 13
- `h` - Hours in 12 hour format eg. 1
- `i` - Minutes eg. 5
- `s` - Seconds eg. 40
- `U` - Seconds since epoch eg. 814807830
- `A` - AM/PM
- `a` - am/pm

Example: This example prints today's date in the format "MM/DD/YYYY":

```
<?
 $today = Date("m/d/Y");
 echo "$today<p>\n";
>
```

Parameters: string, time

See Also: MkTime()

dbList

Syntax: dbList()

Usage: dbList outputs information about the db support compiled into PHP.

dbmClose

Syntax: dbmClose(filename)

Usage: dbmClose simply closes the specified dbm file. It will also unlock any lock files, so it is important to close any dbm files that have been opened.

Parameters: string

dbmDelete

Syntax: dbmDelete(filename,key)

Usage: dbmDelete will delete the key/content pair specified by the given key argument.

Parameters: string, key element

dbmExists

Syntax: dbmExists(filename,key)

Usage: dbmExists will return 1 if the key exists and 0 otherwise.

Parameters: string, key element

Returns: boolean

dbmFetch

Syntax: dbmFetch(filename,key)

Usage: dbmFetch will return the content string associated with the given key.

Parameters: string, key element

Returns: string

dbmFirstKey

Syntax: dbmFirstKey(filename)

Usage: dbmFirstKey returns the first key in the dbm file. Note that no particular order is guaranteed since the order depends on hash table values calculated within the dbm implementation. You may use the Sort function to sort arrays of data from the dbm file if necessary.

Parameters: string

Returns: key element

dbmInsert

Syntax: dbmInsert(filename,key,content)

Usage: dbmInsert inserts a new key/content data pair into a dbm file. If the key already exists, the insert will fail.

Parameters: string, key element, string

Returns: boolean

dbmNextKey

Syntax: dbmNextKey(filename,key)

Usage: dbmNextKey returns the next key after the specified key. By calling dbmfirstkey() followed by successive calls to dbmnextkey() it is possible to visit every key/content pair in the dbm file.

Parameters: string, key element

Returns: key element

dbmOpen

Syntax: dbmOpen(filename,mode)

Usage: dbmOpen() opens a dbm file. The first argument is the full-path filename of the dbm file to be opened and the second is the file open mode which is one of "r", "n" or "w" for read, new (implies write) and write respectively. If ndbm support is used, ndbm will actually create filename.dir and filename.pag files. gdbm only uses one file, as does the internal flat ASCII file support, and Berkeley's libdb create a filename.db file. Note that PHP does its own file locking in addition to any file locking that may be done by the dbm library itself. PHP does not delete the .lck files it creates. It uses these files simply as fixed inodes on which to do the file locking. For more information on dbm files, see your Unix man pages, or obtain GNU's gdbm from `ftp://prep.ai.mit.edu/pub/gnu`.

Parameters: string, character

Returns: boolean

dbmReplace

Syntax: dbmReplace(filename,key,content)

Usage: dbmReplace is similar to the dbminsert() function, the only difference being that if the key already exists, the old content string will be replaced with the new.

Parameters: string, key element, string

DecBin

Syntax: DecBin(number)

Usage: DecBin returns a string containing a binary representation of the given number argument. The largest number that can be converted is 31 bits long or 4294967295 in decimal.

Parameters: integer or float

Returns: binary

See Also: BinDec()

DecHex

Syntax: DecHex(number)

Usage: DecHex converts a decimal number to a hexadecimal string.

Parameters: integer

Returns: string

See Also: HexDec()

DecOct

Syntax: DecOct(number)

Usage: DecOct converts a decimal number to an octal number.

Parameters: integer

Returns: string

See Also: OctDec()

doubleval

Syntax: doubleval(variable)

Usage: doubleval returns the double (floating point) value of the variable.

Returns: double

See Also: strval() and intval()

Echo

Syntax: Echo [format_string] expression [, expression [,...]]

Usage: Echo is not a function. ie. you do not put brackets around the arguments to it. It is used to display results of PHP functions or PHP variables. See the Escape Character Section for a list of special characters supported. The format_string is optional and if not present, no output formatting will be done. The format string is similar to the format string of the C printf function. See the man page for printf for more details. Up to 5 expressions can be printed with a single echo command. If you try to print more you will get a parser error. Note that the types of the expressions are not relevant. The expressions are automagically converted to the appropriate types as specified by the format string if one is present. If you want to format something and assign the formatted string to a variable instead of displaying it, use the sprintf() function.

The following conversions are supported:

- `%d %i` - Print a signed decimal number.
- `%o` - Print an octal number.
- `%u` - Print an unsigned decimal number.
- `%x %X` - Print a hexadecimal number.
- `%f` - Print a floating-point number.
- `%e %E` - Print a floating-point number in scientific notation.

- `%g %G` - Print a floating-point number in scientific notation or normal notation, as appropriate.
- `%c` - Print a single character.
- `%s` - Print a string of characters.
- `%%` - Print a literal percent-sign.

The following flags are accepted.

- `'-'` Left-justify the output within the field-width.
- `'+'` - Ensure that all integers are signed (with a plus/minus sign).
- `' '` - Similar to '+', but uses a space instead of a plus-sign.
- `'#'` - Print prefixes in front of hex and octal numbers designating them as such.
- `'''` - Separate the digits into groups (usually comma-separated groups of three).
- `'0'` - Pad the field-width with zeros.

All of these flags are dependent upon whether or not your C library's printf function supports them (the "', for example, is a GNU extension).

Most conversions will accept a field width and a precision, as shown in the demo_echo.html file in the directory /examples. It is not necessary to specify any type modifiers, and, in fact, PHP will complain if the type modifier does not make sense (which is almost always the case). PHP will complain about (and refuse to accept) anything that it does not recognize. Any extra arguments given that are not required by the format-string are ignored.

End

Syntax: End(variable)

Usage: End moves the internal array pointer for the given variable to the last item of the array and returns the value of this item. This is useful for traversing an associative array in reverse order.

Example: The following example would traverse an associative array in reverse order:

```
<?
 Reset($array);
 $first_key = key($array);
 End($array);
 $k = key($array);
 while($k != $first_key);
    echo $array[$k];
    prev($array);
    $k = key($array);
 endwhile;
 echo $array[$k];
>
```

Parameters: array

Returns: item value

See Also: Reset() and Prev()

ereg

Syntax: ereg(expr,arg[,regs])

Usage: ereg returns non-zero if the regular expression is matched in the argument string. For example, the condition:

```
<?if (ereg("^This.*", "This is an example string")>
```

would be true since the:

```
 "^This.*"
```

expression says to match the word This at the beginning of the string and then match any characters afterwards. If the regs argument is present, then match registers are filled into positions 0-10 in the array named by the regs argument. Register 0 will always contain the full matched string. For more information on regular expressions, see the regular expression section of the full PHP/FI documentation.

Parameters: regex, string

Returns: boolean

eregi

Syntax: eregi(expr,arg[,regs])

Usage: eregi is identical to the ereg() function except for the fact that the regular expression is applied such that upper/lower case is ignored.

See Also: ereg()

ereg_replace

Syntax: ereg_replace(expr,replace,arg)

Usage: ereg_Replace scans the entire argument string and replaces any portions of the string matched by the given expression with the replacement string. For example, in the string, "This is an example string" we could very easily replace every space with a dash with the command:

```
ereg_replace("","-","This is an example string")
```

For more information on regular expressions, see the regular expression section of the full PHP/FI documentation.

See Also: ereg()

eregi_replace

Syntax: eregi_replace(expr,replace,arg)

Usage: eregi_replace is identical to the ereg_replace() function except for the fact that the regular expression is applied such that upper/lower case is ignored.

See Also: ereg_replace(), ereg()

EscapeShellCmd

Syntax: EscapeShellCmd(string)

Usage: EscapeShellCmd escapes any characters in a string that might be used to trick a shell command into executing arbitrary commands. This function should be used to make sure that any data coming from user input is escaped before this data is passed to the Exec() or System() functions.

Example: `<?system(EscapeShellCmd($cmd))>`

Parameters: string

Returns: string

Eval

Syntax: Eval(string)

Usage: Eval takes the contents of the string argument and treats it like a mini PHP/FI script. It will execute it as a separate PHP/FI script. Any variables set or accessed from inside the eval will be from the global reference frame in the current context of the eval statement in the script. Variable substitution is done on the string arguments, so if variables are to be used in the string expression they should be escaped.

Example: The following example illustrates how eval() is used:

```
$a = "echo phpversion();";
eval($a);

eval("echo phpversion();");

eval("\$a=1; echo \$a;");
```

Parameters: string

Returns: string

Exec

Syntax: Exec(command_string [, array [,return_var]])

Usage: Exec executes the given unix command, however it does not output anything. It simply returns the last line from the result of the command. If you need to execute a command and have all the data from the command passed directly back without any interference, use the PassThru() function. If the array argument is present, then the specified array will be filled with every line of output from the unix command starting at the end of the array. Make sure you UnSet the array before the call if your array already contains elements and you want to start filling it at array element 0. If the return_var argument is present along with the array argument, then the return status of the executed unix command will be written to this variable. Note that if you are going to allow data coming from user input to be passed to this Exec function, then you should be using the EscapeShellCmd() function to make sure that users cannot trick the system into executing arbitrary commands.

Parameters: string

Returns: optional array

See Also: EscapeShellCmd(), system()

Exit

Syntax: Exit

Usage: The Exit command is used to terminate parsing right away as soon as this tag is parsed.

Exp

Syntax: Exp(arg)

Usage: Exp returns e raised to the power of arg.

Parameters: number

See Also: pow()

fclose

Syntax: fclose($fd)

Usage: fclose() closes a file opened by fopen(). The argument is a file pointer index as returned by the fopen() call.

Parameters: file pointer

Returns: file pointer index

feof

Syntax: feof($fd)

Usage: Feof returns true if the file referred to by the file pointer index argument has hit end-of-file.

Parameters: file pointer

Returns: file pointer index

fgets

Syntax: fgets($fd,bytes)

Usage: fgets() reads a line from a file opened by fopen(). Arguments are a file pointer index as returned by fopen() and the max number of bytes to read. Reading ends when max number of bytes have been read, or on an end of line. This is similar to the C fgets() call.

Parameters: file pointer, integer

Returns: binary data

See Also: fputs()

fgetss

Syntax: fgetss($fd,bytes)

Usage: Identical to the fgets() function, except this one tries to strip off any HTML tags or PHP/FI script tags as it is reading the file.

See Also: fgets()

File

Syntax: $array = File(filename)

Usage: File reads the entire file and returns an array with each array element containing a line of the file starting with array index 0.

Parameters: string

Returns: array

FileAtime

Syntax: FileAtime(filename)

Usage: fileAtime returns the time of last data access. If the file does not exist, or if it for some other reason could not be accessed, this function returns -1. If repeated calls to fileAtime and the rest ofthe file* functions are to be made and the file being accessed might change or disappear, the ClearStatCache() should be called before the call to the file* function.

Parameters: string

Returns: time

See Also: ClearStatCache()

fileCtime

Syntax: fileCtime(filename)

Usage: fileCtime returns the time of last status change. If the file does not exist, or if it for some other reason could not be accessed, this function returns -1. If repeated calls to fileCtime and the rest ofthe file* functions are to be made and the file being accessed might change or disappear, the ClearStatCache() should be called before the call to the file* function.

Parameters: string

Returns: time

See Also: ClearStatCache()

fileGroup

Syntax: fileGroup(filename)

Usage: fileGroup returns the group id of the owner of the file. If the file does not exist, or if it for some other reason could not be accessed, this function returns -1. If repeated calls to fileGroup and the rest ofthe file* functions are to be made and the file being accessed might change or disappear, the ClearStatCache() should be called before the call to the file* function.

Parameters: string

Returns: string

See Also: ClearStatCache()

fileOwner

Syntax: fileOwner(filename)

Usage: fileOwner returns the uid of the owner of the file. If the file does not exist, or if it for some other reason could not be accessed, this function returns -1. If repeated calls to fileOwner and the rest ofthe file* functions are to be made and the file being accessed might change or disappear, the ClearStatCache() should be called before the call to the file* function.

Parameters: string

Returns: string

See Also: ClearStatCache()

filePerms

Syntax: filePerms(filename)

Usage: filePerms returns the permission bits of the file. This is the st_mode field of the Unix C stat structure. If the file does not exist, or if it for some other reason could not be accessed, this function returns -1. If repeated calls to filePerms and the rest ofthe file* functions are to be made and the file being accessed might change or disappear, the ClearStatCache() should be called before the call to the file* function.

Parameters: string

Returns: string

See Also: ClearStatCache()

fileType

Syntax: fileType(filename)

Usage: fileType returns the type of the file. The return values are one of: "dir", "file","fifo","char", "block", or "link". These are for, directory, regular file, fifo special, character special, block special and symbolic link, respectively.

Parameters: string

Returns: string

See Also: ClearStatCache()

Floor

Syntax: Floor(value)

Usage: Floor() rounds a floating point value down to the previous integer. The return value is of type double (floating point) such that it can be used properly in complex equations. To get an integer type back, use: `$new = IntVal(Floor($value));`

Parameters: number

Returns: double

See Also: Ceil()

Flush

Syntax: Flush()

Usage: The Flush() function is used to Flush the output buffer. For the Apache module, it flushes Apache's output buffer, and for the CGI version it simply flushes stdout. When running as a CGI under Apache, the server will buffer the CGI script's output, so this Flush() function won't help much there. Look at running your script as an nph- script if you are running the CGI version of PHP under the Apache web server. Or, alternatively, run the Apache module version of PHP.

fopen

Syntax: $fp = fopen(filename,mode)

Usage: fopen() opens a file and returns a file pointer index. If the file could not be opened the function returns -1. It is similar to the C fopen() call. The filename argument is the relative or absolute path to the file to be opened, and the mode argument is one of, "r", "r+", "w", "w+", "a", "a+". See the Unix man page on the fopen() call for more information.

Example: The following example illustrates how fopen() is used:

```
$fp = fopen("/home/rasmus/file.txt","r");
```

Parameters: string, octal

Returns: boolean

See Also: popen(), close()

fputs

Syntax: fputs(fp,string)

Usage: fputs() writes a line to a file opened by fopen(). Arguments are a file pointer index as returned by fopen() and the string to write. Note that the string argument may contain the special escape characters:

```
\n, \r and \t
```

to output newlines, carriage returns and tabs respectively.

See Also: gets()

FPassThru

Syntax: FPassThru($fp)

Usage: FPassThru() outputs all remaining data on fp directly. It is different from ReadFile() in that it can also handle files opened with fsockopen(). It differs from PassThru() in that it does not handle commands, but opened files. FPassThru() returns the number of bytes read and written.

Parameters: file pointer

fseek

Syntax: fseek(fp,pos)

Usage: fseek() positions a file pointer identified by the $fd argument which is the return value of the fopen() call. The file pointer is positioned at the beginning of the file plus the offset specified by the pos argument.

Parameters: file pointer, integer

See Also: ftell(), rewind()

fsockopen

Syntax: fp = fsockopen(hostname,port)

Usage: fsockopen() opens a socket connection and returns a file pointer index. This file pointer index can be used by fgets, fputs and fclose. Arguments are a hostname and a port number. Return values are: -3 if the socket couldn't be created, -4 if the dns lookup on the hostname failed, -5 if the connection was refused or it timed out, -6 if the actual fdopen() call failed or -7 if the setvbuf() call failed. If the port number is 0, then the hostname argument will be treated as a filename of a Unix domain socket if your operating system support Unix domain sockets.

Parameters: string, integer

Returns: integer

ftell

Syntax: $pos = ftell(fp)

Usage: ftell() returns the position of a file pointer identified by the fp argument which is the return value of the fopen() call. The position can later be used as an argument to fseek().

Parameters: file pointer

See Also: fseek(), rewind()

getAccDir

Syntax: getAccDir()

Usage: getAccDir returns the directory where PHP access configuration files are kept. The access configuration filenames come from the numerical user id of the user whose access configurations they represent.

GetEnv

Syntax: GetEnv(string)

Usage: GetEnv returns the value of the environment value specified by string. Normally this function is not used because environment variables are available to PHP/FI directly. If a reference is made to a variable which is not found in the internal symbol table, then the environment space is automatically searched. GetEnv should be used when it is necessary to ensure that an environment variable has not been overwritten by normal PHP/FI variable. Security mechanisms that rely on http server-defined variables like REMOTE_ADDR and REMOTE_HOST should load these variables using GetEnv as opposed to referencing them directly as $REMOTE_ADDR to avoid someone making up a fake form and posting the data to your server and thereby bypassing whatever security mechanism you might have.

Parameters: string

Returns: string

getHostByName

Syntax: getHostByName(domain_name)

Usage: getHostByName converts the given domain name into an IP address in nnn.nnn.nnn.nnn format.

Parameters: string

Returns: string

getHostByAddr

Syntax: getHostByAddr(ip_address)

Usage: getHostByAddr converts the given IP address in nnn.nnn.nnn.nnn format into a fully qualified domain name.

Parameters: string

Returns: string

GetImageSize

Syntax: GetImageSize(filename)

Usage: The GetImageSize() function takes either a full path filename, or a relative path relative to the location of the calling script. It returns a 3 element array consisting of width, height and type. Width and height are in pixels, and a type of 1 means GIF, a 2 indicates a JPG file and a 3 indicates a PNG file. Other file types are not supported. The fourth element in the returned array is a string containing, "width=x height=y" that is suitable for using directly in an IMG tag. It is important to note that the GD image library is not needed to use this function.

Example: The following example illustrates how GetImageSize() is used:

```
<?
 $result = GetImageSize("img/flag.jpg");
>
<IMG SRC="img/flag.jpg" <?echo $result[3]> >
```

Parameters: string

Returns: string

getMyPid

Syntax: getMyPid()

Usage: getMyPid() returns the current process id of the PHP parsing process.

getMyUid

Syntax: getMyUid()

Usage: getMyUid returns the numerical user id of the owner of the current HTML file.

getRandMax

Syntax: getRandMax()

Usage: getRandMax returns the maximum random number the Rand function will return. If the value returned does not seem to be accurate, have a look in the php.h source file in the PHP distribution for more information.

GetType

Syntax: GetType(variable)

Usage: GetType returns the type of the variable. The return value is a string and it is one of, "integer", "double" or "string".

Parameters: any variable

Returns: string

See Also: SetType()

gmDate

Syntax: gmDate(format,time)

Usage: gmDate is identical to the Date function except for the fact that it uses Greenwich Mean Time instead of the current local time.

See Also: Date()

Header

Syntax: Header(header_string)

Usage: The Header command is used at the top of an HTML file to send raw HTTP header strings. See the HTTP Specification for more information on raw http headers. Remember that the Header() command must be used before any actual output is sent either by normal HTML tags or by PHP echo commands. Usage examples can be found in the HTTP Authentication section of the full PHP/FI documentation.

HexDec

Syntax: HexDec(hex_string)

Usage: HexDec converts a hexadecimal string to a decimal number.

Parameters: hexadecimal string

Returns: decimal

See Also: DecHex()

HtmlSpecialChars

Syntax: HtmlSpecialChars(string)

Usage: HtmlSpecialChars converts any characters with ASCII codes in the string argument between 160 and 255 inclusive to their corresponding HTML Entity names. The function returns the converted string. The:

```
<, >, & and "
```

characters are also converted.

Parameters: string

Returns: string

Include

Syntax: Include(filename)

Usage: The Include command can be used to insert other files into the current html file. This is extremely handy for headers and footers which may need to be included in hundreds of HTML files. By using an include command you then only need to modify the header or footer file in one place when it needs to be changed. Since full PHP parsing is done on the included file, you can also use the include command to include common PHP scripts you may have written. Sort of like having a primitive shared library of scripts you can call from your HTML file. You can place such common library files in one directory and set PHP's include path and not have to refer to the files with pathnames. For Apache module users this can be configured with the phpIncludePath directive, for CGI users with the PHP_INCLUDE_PATH environment variable. This path is colon-separated just like $PATH is in your UNIX shell.

Example: `<?include("/path/filename.txt")>`

InitSyslog

Syntax: InitSyslog()

Usage: InitSyslog() defines some PHP variables that you need when using OpenLog() and Syslog(). These variables are not defined by default for efficiency reasons. The variables are named in the same way as in the syslog.h C include file (such as $LOG_LOCAL0). See your syslog(3) UNIX manual page for more details. See also

See Also: InitSyslog(), Syslog(), CloseLog()

intval

Syntax: intval(variable)

Usage: intval returns the long integer value of the variable.

See Also: strval(), doubleval()

Key

Syntax: Key(variable)

Usage: Key returns the key of the current array item. The current item is determined by the position of the array pointer for the given variable. This array pointer may be manipulated with the Reset(), End(), Next(), and Prev() functions. This function is mainly used for determining the key value for an item in an associative array, although it will work for normal array as well.

Parameters: array

Returns: array pointer

Link

Syntax: Link(target,link)

Usage: Link() creates a hard link on the filesystem.

See Also: Symlink(), ReadLink() LinkInfo()

LinkInfo

Syntax: LinkInfo(path)

Usage: LinkInfo returns the st_dev field of the UNIX C stat structure returned by the lstat system call. This function is used to verify if a link (pointed to by path) really exists (using the same method as the S_ISLNK macro defined in stat.h). Returns -1 in case of error.

Log

Syntax: Log(arg)

Usage: Log returns the natural logarithm of arg.

Log10

Syntax: Log10(arg)

Usage: Log10 returns the base-10 logarithm of arg.

LogAs

Syntax: LogAs(filename)

Usage: The LogAs() function will treat the hit on the current page as if it as actually received on the argument filename.

Mail

Syntax: Mail(to,subject,message[,headers])

Usage: Mail automatically mails the message specified in the message argument to the receiver specified in the to argument. Multiple recipients can be specified by spaces between them in the to argument.

Example: The following example illustrates how mail() is used:

```
<?
mail("rasmus@lerdorf.on.ca",
"My Subject",
"Line 1\nLine 2\nLine 3");
>
```

If a fourth string argument is passed, this string is inserted at the end of the header, example:

```
<?
mail("ssb@guardian.no", "the subject", $message,
"X-Mailer: PHP/FI " + phpversion());
>
```

Max

Syntax: Max(array)

Usage; Max returns the maximum value of a PHP array. ie. it will search through the entire array looking for the max element. If it is an array of strings, the returned string is the string which would be alphabetically last in the array if it were sorted.

Microtime

Syntax: Microtime()

Usage: Microtime() returns a string "msec sec" where sec is number of seconds since 00:00 GMT, Jan 1, 1970, and msec is the microseconds part (as fraction of seconds). Ex "0.87633900 825010464". This function is only available on operating systems that support the gettimeofday() system call.

Min

Syntax: Min(array)

Usage: Min returns the minimum value of a PHP array. ie. it will search through the entire array looking for the min element. If it is an array of strings, the returned string is the string which would be alphabetically first in the array if it were sorted.

MkDir

Syntax: MkDir(dir,mode)

Usage: MkDir creates a directory. The mode parameter must be given in octal notation.

Example: `MkDir("DirName",0755);`

MkTime

Syntax: MkTime(hour,min,sec,mon,day,year)

Usage: MkTime returns a time in Unix timestamp (long integer) format which corresponds to the date and time specified by the arguments. Arguments may be left out in which case the given component is set to the current value according to the current local time and date. These left out arguments may only be left out from right to left. ie. MkTime(hour,min,sec) is valid, but MkTime(mon,day,year) is not valid. Note that this function can be very handy as a tool for doing both date arithmetic and date validation. You can feed it invalid parameters, such as a month greater than 12, or a day greater than 31 and it will figure out the right date anyway. It will also generate an error message if any of the parameters are outside the normal values. Use the SetErrorReporting(0) function to turn this error reporting off before calling the function and you may then check the $phperrmsg for any errors that may have occurred.

Example: The following example illustrates how MkTime() is used:

```
<?
SetErrorReporting(0);
$a = MkTime(0,0,0,13,1,1997);
SetErrorReporting(1);
echo $phperrmsg;
>
```

Next

Syntax: Next(variable)

Usage: Next moves the internal array pointer to the next item in the array. This happens automatically when an array is accessed using the non-indexed method ($array[]). The function returns the value of the new item. This function can be used to move the pointer forward without having to access the array explicitly. One use would be to traverse an associative array and only printing out the keys of the array and not the actual contents.

Example: The following example illustrates how next() is used:

```
<?
 Reset($array);
 $i=0;
 while($i < count($array));
  echo key($array);
  next($array);
  $i++;
 endwhile;
>
```

OctDec

Syntax: OctDec(octal_number)

Usage: OctDec converts an octal number to a decimal number.

See Also: DecOct()

openDir

Syntax: openDir(directory)

Usage: openDir opens the specified directory and places an internal pointer to the beginning of the directory. Directory entries are read using the readDir function, and an opened directory should be closed with the closeDir function.

OpenLog

Syntax: OpenLog(ident,options,facility)

Usage: OpenLog() initializes the system for further Syslog() calls. See the openlog(3) UNIX man page for more details.

See Also: InitSyslog(), Syslog(), CloseLog()

Ord

Syntax: Ord(arg)

Usage: Ord returns the ASCII value of the first character of arg.

Parse_Str

Syntax: Parse_Str(arg)

Usage: Parse_str takes a string identical to a regular URL encoded string and extracts variables and their values.

Example: The following example illustrates how parse_str() is used:

```
<? parse_str("a[]=hello+world&a[]=second+variable");
 echo $a[],"<br>";
 echo $a[],"<br>";
>
```

produces:

```
 hello world
 second variable
```

PassThru

Syntax: PassThru(command_string [,return_var])

Usage: The PassThru() function is similar to the Exec() function in that it executes a Unix command. If the return_var argument is present, the return status of the Unix command will be placed here. This command should be used in place of Exec or System when the output from the Unix command is binary data which needs to be passed directly back to the browser. A common use for this is to execute something like the pbmplus utilities that can output an image stream directly. By setting the content-type to image/gif and then calling a pbmplus program to output a gif, you can create PHP/FI scripts that output images directly.

pclose

Syntax: pclose(fp)

Usage: Pclose closes a pipe opened using the popen() function.

See Also: popen()

phpInfo

Syntax: phpInfo()

Usage: phpInfo prints the same page you get when adding "?info" to a PHP/FI parsed URL or when you run the php.cgi binary by itself. It is especially useful for debugging scripts in the Apache module version since it displays a number of useful internal data.

phpVersion

Syntax: phpVersion()

Usage: phpVersion returns the version number of PHP/FI currently running.

popen

Syntax: fp = popen(command,mode)

Usage: Popen opens a pipe to a command and returns a file pointer index. This file pointer index can be used by fgets, fputs and fclose. Arguments are the command to run and the mode. The mode can be either "r" for read or "w" for write. See the UNIX C library popen man page for more details. Any file opened with popen() should be closed using the pclose() function.

See Also: pclose()

Pos

Syntax: Pos(var)

Usage: The Pos() function returns the numerical position of an array element within that array. This is not very useful for normal arrays, but for associative arrays it can be handy.

pow

Syntax: pow(x,y)

Usage: Evaluates x raised to the power of y.

See Also: Exp()

Prev

Syntax: Prev(variable)

Usage: Prev moves the internal array pointer for the given variable to the previous item in the array. If already at the beginning of the list, the pointer will point to the first item. The function returns the value of the new item. This function is useful for traversing an associative array in reverse order.

See Also: End(), Next()

PutEnv

Syntax: PutEnv(string)

Usage: PutEnv puts the given string in the environment. Not extremely useful since the local environment variables are wiped out when PHP is done with a page, but in some cases it is useful if other things called from within a PHP script checks environment variables. For example, if you want to run multiple mSQL daemon processes, you will need to use PutEnv to switch back and forth between the different sockets.

QuoteMeta

Syntax: QuoteMeta(arg)

Usage: QuoteMeta returns a string composed of arg with any regular expression special characters escaped with a backslash.

Rand

Syntax: Rand()

Usage: Rand returns a random number between 0 and RANDMAX. Remember to seed your random number generator with a call to srand() before calling rand(). You only need to seed the random number generator once. RANDMAX can be determined with the getRandMax function. Normally a specific range is chosen by simply using the modulus operator on the result.

readDir

Syntax: readDir()

Usage: readDir reads the next entry from the currently open directory structure. Once an entry is read, the pointer is advanced to the next entry in the directory and the next call to this function will return the next entry in the directory. Use the openDir function to open a directory before calling this function.

See Also: openDir()

ReadFile

Syntax: $size = ReadFile(filename)

Usage: Reads the file filename and simply outputs it directly. It returns the number of bytes actually read. It is different from the File() command in that it does not store the file in memory, and it is safe for use on binary files. This function is generally used where one might otherwise do a PassThru("cat filename"). Using ReadFile is more efficient.

Reset

Syntax: Reset(variable)

Usage: Reset moves the internal array pointer for the given array variable to the first item of the array and returns the value of this item. This is useful for traversing associative and non-indexed arrays.

Example: The following example traverses an associative array:

```
<?
 Reset($array);
 $i=0;
 while($i < count($array));
   echo $array[]; /* pointer automatically moves
                     ahead one */
   $i++;
 endwhile;
>
```

See Also: End(), Next()

SetCookie

Syntax: SetCookie(name,value,expire,path,domain,secure)

Usage: SetCookie() defines a cookie to be sent along with the rest of the header information. All the arguments except the name argument are optional. If only the name argument is present, the cookie by that name will be deleted from the remote client. You may also replace any argument with an empty string ("") in order to skip that argument. The expire and secure arguments are integers and cannot be skipped with an empty string. Use a zero (0) instead. The expire argument is a regular Unix time integer as returned by the time() or mktime() functions.

Example: The following example illustrates how SetCookie() is used:

```
SetCookie("TestCookie","Test Value");
SetCookie("TestCookie",$value,time()+3600);
   /* expire in 1 hour */
SetCookie("TestCookie",$value,time()+3600,
   "/~rasmus/",".utoronto.ca",1);
```

SetErrorReporting

Syntax: SetErrorReporting(arg)

Description: SetErrorReporting sets the current error reporting state to the value of arg. If non-zero, errors will be printed, and if 0 they won't be. The function returns the previous error reporting state. This is a more general way of disabling error reporting than by preceding individual functions with a '@' character. See the section on Suppressing Errors from function calls for more information.

Sleep

Syntax: Sleep(secs)

Usage: Sleep will delay for secs seconds. Similar to the Unix C sleep() function.

See Also: USleep()

Sort

Syntax: Sort(array)

Usage: Sort is used to sort a PHP array in ascending order. To sort in descending order, use the RSort() function. It understands the three variable types and will sort alphabetically if the array contains strings, and numerically if the array contains numbers. In the case of an array which contains a mixture of types, the first type in the array will specify the sort method. Note that if you are going to sort an associative array, you should use the ASort() function.

See Also: ASort()

Sprintf

Syntax: Sprintf(format,arg [,arg,arg,arg,arg])

Usage: Sprintf returns the string created by the formatted output defined by the format argument and arg. It is similar to the formatted version of the echo command, except this just returns the string while echo displays it. It is also similar to the C function by the same name. The difference being that this version does not accept more than 5 arg arguments. If you need to format more than 5 arguments into a single string, simply call sprintf() multiple times for each group of arguments. Note that the type of the argument doesn't affect the output. The argument type is automagically converted to match the type specified in the format string.

Srand

Syntax: Srand(integer)

Usage: Srand seeds the random number generator. This function takes any integer as an argument. One choice for a seed value is to use the date function to give you the current number of seconds past the minute. Note that this function does not return a value! This function simply seeds the random number generator for subsequent calls to the rand() function.

Example: `<?srand(date("s"))>`

See also: rand()

strchr

Syntax: strchr(string,arg)

Usage: strchr and strstr are actually identical functions. They can be used interchangeably and both are included for completeness sake. They will return the portion of the string argument starting at the point where the given sub-string is found.

Example: In the string, "This is an example string" above, the call:

```
<?echo strchr(\$string,"an ")>
```

would return the string: "an example string".

strtr

Syntax: strtr(input,set1,set2)

Usage: strtr() translates each character of "string" that is in "set1" to the corresponding character in "set2". Characters not in set1 are passed through unchanged. When a character appears more than once in ”set1” and the corresponding characters in ”set2” are not all the same, only the final one is used. When one of ”set1” or ”set2” is longer, longer ”set?” is truncated to length of shorter ”set?”.

StripSlashes

Syntax: StripSlashes(arg)

Usage: StripSlashes unescapes the string argument.

See Also: AddSlashes()

strlen

Syntax: strlen(string)

Usage: strlen returns the length of the string.

strrchr

Syntax: strrchr(string,arg)

Usage: strrchr will search for a single character starting at the end of the argument string and working its way backwards. It returns the string starting with the search character if the character was found and an empty string if it wasn't.

strstr

Syntax: strstr(string,arg)

See Also: strchr()

strtolower

Syntax: strtolower(string)

Usage: strtolower converts the string argument to all lower case characters.

strtoupper

Syntax: strtoupper(string)

Usage: strtoupper converts the string argument to all upper case characters.

strval

Syntax: strval(variable)

Usage: strval returns the string value of the variable.

See Also: intval(), doubleval()

substr

Syntax: substr(string, start, length)

Usage: substr returns a part of the given string. The start position is given by the start argument. The first position in a string is position 0. And the length argument specifies the number of characters to return from the start position.

System

Syntax: System(command_string [,return_var])

Usage: System is just like the C system() command in that it executes the given unix command and outputs the result. If a variable is provided as the second argument, then the return status code of the executed unix command will be written to this variable. Note, that if you are going to allow data coming from user input to be passed to this System function, then you should be using the EscapeShellCmd() function to make sure that users cannot trick the system into executing arbitrary commands. The System() call also automatically flushes the Apache output buffer after each line of output if PHP is running as an Apache module. If you need to execute a command and have all the data from the command passed directly back without any interference, use the PassThru() function.

See Also: PassThru()

Time

Syntax: Time()

Usage: Time simply returns the current local time in seconds since Unix epoch (00:00:00 Jan. 1 1970). It is equivalent to calling Date("U"). If you need better than per-second granularity, use the Microtime function.

USleep

Syntax: USleep(microsecs)

Usage: Sleep will delay for the given number of microseconds. Similar to the Unix C usleep() function.

See Also: Sleep()

Index

D

E

F

Q

R

S

T

U

V