

1 Warnings and Errors Troubleshooting Index

1.1 Description

If you encounter an error or warning you don't understand, check this chapter for solutions to common warnings and errors that the mod_perl community has encountered in the last few years.

1.2 General Advice

Perl's warnings mode is immensely helpful in detecting possible problems. Make sure you always turn on warnings while you are developing code. See The Importance of Warnings.

Enabling `use diagnostics;` generally helps you to determine the source of the problem and how to solve it. See diagnostics pragma for more information.

1.3 Building and Installation

See make Troubleshooting and make test Troubleshooting

1.4 Configuration and Startup

This section talks about errors reported when you attempt to start the server.

1.4.1 *SegFaults During Startup*

Possible reasons and solutions:

- You have to build mod_perl with the same compiler as Perl was built with. Otherwise you may see segmentation faults and other weird things happen.
- This could be the XSLoader vs. DynaLoader problem, where the list of dl_handles created by XSLoader is wiped out by DynaLoader. Try adding:

```
use DynaLoader ();
```

to your *startup.pl* before any other module is loaded.

This has been resolved in perl 5.8.0. For earlier versions of Perl you need to comment out:

```
#@dl_librefs      = ();      # things we have loaded
#@dl_modules      = ();      # Modules we have loaded
```

in *DynaLoader.pm*.

However if none of this cases applies and you still experience segmentation faults, please report the problem using the following guidelines.

1.4.2 libexec/libperl.so: open failed: No such file or directory

If when you run the server you get the following error:

```
libexec/libperl.so: open failed: No such file or directory
```

it probably means that Perl was compiled with a shared library. `mod_perl` does detect this and links the Apache executable to the Perl shared library (*libperl.so*).

First of all make sure you have Perl installed on the machine, and that you have *libperl.so* in `<perl-root>/<version>/<architecture>/CORE`. For example in `/usr/local/lib/perl5/5.00503/sun4-solaris/CORE`.

Then make sure that directory is included in the environment variable `LD_LIBRARY_PRELOAD`. Under normal circumstances, Apache should have the path configured at compile time, but this way you can override the library path.

1.4.3 install_driver(Oracle) failed: Can't load '.../DBD/Oracle/Oracle.so' for module DBD::Oracle

```
install_driver(Oracle) failed: Can't load
'/usr/lib/perl5/site_perl/5.005/i386-linux/auto/DBD/Oracle/Oracle.so'
for module DBD::Oracle:
libclntsh.so.8.0: cannot open shared object file:
No such file or directory at
/usr/lib/perl5/5.00503/i386-linux/DynaLoader.pm line 169.
at (eval 27) line 3
Perhaps a required shared
library or dll isn't installed where expected at
/usr/local/apache/perl/tmp.pl line 11
```

On BSD style filesystems `LD_LIBRARY_PATH` is not searched for setuid programs (a.k.a., Apache). This isn't a problem for CGI script since they don't do a setuid (and are forked off), but Apache does, and `mod_perl` is in Apache. Therefore the first solution is to explicitly load the library from the system wide *ldconfig* configuration file:

```
# echo $ORACLE_HOME/lib >> /etc/ld.so.conf
# ldconfig
```

Another solution to this problem is to modify the resulting *Makefile* (after running `perl Makefile.PL`) as follows:

1. search for the line `LD_RUN_PATH=`
2. replace it with `LD_RUN_PATH=(my_oracle_home)/lib`

(`my_oracle_home`) is, of course, the home path to your oracle installation. In particular, the file `libclntsh.so.8.0` should exist in that directory. (If you use CPAN, the build directory for `DBD::Oracle` should be in `~/cpan/build/DBD-Oracle-1.06/` if you're logged in as root.)

Then, just type `make install`, and all should go well.

FYI, setting `LD_RUN_PATH` has the effect of hard-coding the path to *(my_oracle_home)/lib* in the resulting `Oracle.so` file generated by the `DBD::Oracle` so that at run-time, it doesn't have to go searching through `LD_LIBRARY_PATH` or the default directories used by `ld`.

For more information see the `ld` manpage and an essay on `LD_LIBRARY_PATH`: <http://www.visi.com/~barr/ldpath.html>

1.4.4 Invalid command 'PerlHandler'...

```
Syntax error on line 393 of /etc/httpd/conf/httpd.conf: Invalid
command 'PerlHandler', perhaps mis-spelled or defined by a module
not included in the server configuration [FAILED]
```

This can happen when you have a `mod_perl` enabled Apache compiled with DSO (generally it's an installed RPM or other binary package) but the `mod_perl` module isn't loaded. In this case you have to tell Apache to load `mod_perl` by adding:

```
AddModule mod_perl.c
```

in your `httpd.conf`.

This can also happen when you try to run a non-`mod_perl` Apache server using the configuration from a `mod_perl` server.

1.4.5 symbol `__floatdisf`: referenced symbol not found

This problem is experienced by users on certain Solaris versions. When the server is built with modules that use the `__floatdisf` symbol it can't be started. e.g.:

```
Cannot load /usr/local/apache/libexec/libproxy.so into server:
ld.so.1: ../bin/httpd: fatal: relocation error: file
/usr/local/apache/libexec/libproxy.so: symbol __floatdisf: referenced
symbol not found
```

The missing symbol is in `libgcc.a`. Use

```
% gcc -print-libgcc-file-name
```

to see where that file is. Once found you have to relink the module with that file. You can also look for it in the gcc tree, e.g. under `gcc-3.2.1/gcc`.

First, configure and install Apache. Next, relink `mod_proxy.so` or `mod_negotiation.so`, or whatever the module that reports the problem with `libgcc.a`.

```
% cd apache_1.3.27/src/modules
% ld -G -o mod_proxy.so mod_proxy.lo /pathto/libgcc.a
```

(adjust the `/path/to/` to point to the right file from the gcc output stage.)

You can now verify with `nm` that `mod_proxy.so` includes that symbol.

1.4.6 RegistryLoader: Translation of uri [...] to filename failed

```
RegistryLoader: Translation of uri [/home/httpd/perl/test.pl] to filename
failed [tried: /home/httpd/docs/home/httpd/perl/test.pl]
```

This error shows up when `Apache::RegistryLoader` fails to translate the URI into the corresponding filesystem path. Most failures happen when one passes a file path instead of URI. (A reminder: `/home/httpd/perl/test.pl` is a file path, while `/perl/test.pl` is a URI). In most cases all you have to do is to pass something that `Apache::RegistryLoader` expects to get - the URI, but there are more complex cases. `Apache::RegistryLoader`'s man page shows how to handle these cases as well (look for the `trans()` sub).

1.4.7 "Apache.pm failed to load!"

If your server startup fails with:

```
Apache.pm failed to load!
```

try adding this to `httpd.conf`:

```
PerlModule Apache
```

1.5 Code Parsing and Compilation

1.5.1 Value of \$x will not stay shared at - line 5

`my()` Scoped Variable in Nested Subroutines.

1.5.2 Value of \$x may be unavailable at - line 5.

`my()` Scoped Variable in Nested Subroutines.

1.5.3 Can't locate loadable object for module XXX

There is no object built for this module. e.g. when you see:

```
Can't locate loadable object for module Apache::Util in @INC...
```

make sure to give `mod_perl`'s `Makefile.PL` `PERL_UTIL_API=1`, `EVERYTHING=1` or `DYNAMIC=1` parameters to enable and build all the components of `Apache::Util`.

1.5.4 Can't locate object method "get_handlers"...

Can't locate object method "get_handlers" via package "Apache"

You need to rebuild your mod_perl with stacked handlers, i.e. PERL_STACKED_HANDLERS=1 or more simply EVERYTHING=1.

1.5.5 Missing right bracket at line ...

Most often you will find that you really do have a syntax error. However the other reason might be that a script running under Apache::Registry is using __DATA__ or __END__ tokens. Learn why.

1.5.6 Can't load './auto/DBI/DBI.so' for module DBI

Check that all your modules are compiled with the same Perl that is compiled into mod_perl. Perl 5.005 and 5.004 are not binary compatible by default.

Other known causes of this problem:

OS distributions that ship with a broken binary Perl installation.

The 'perl' program and 'libperl.a' library are somehow built with different binary compatibility flags.

The solution to these problems is to rebuild Perl and any extension modules from a fresh source tree. Tip for running Perl's Configure script: use the '-des' flags to accept defaults and '-D' flag to override certain attributes:

```
% ./Configure -des -Dcc=gcc ... && make test && make install
```

Read Perl's INSTALL document for more details.

Solaris OS specific:

"Can't load DBI" or similar error for the IO module or whatever dynamic module mod_perl tries to pull in first. The solution is to re-configure, re-build and re-install Perl and dynamic modules with the following flags when Configure asks for "additional LD flags":

```
-Xlinker --export-dynamic
```

or

```
-Xlinker -E
```

This problem is only known to be caused by installing gnu ld under Solaris.

1.6 Runtime

1.6.1 "exit signal Segmentation fault (11)" with mysql

If you build `mod_perl` and `mod_php` in the same binary, you might get Segmentation fault followed by this error:

```
exit signal Segmentation fault (11)
```

The solution is to not rely on PHP's built-in MySQL support, and instead build `mod_php` with your local MySQL support files by adding `--with-mysql=/path/to/mysql` during `./configure`.

1.6.2 foo ... at /dev/null line 0

Under `mod_perl` you may receive a warning or an error in the `error_log` which specifies `/dev/null` as the source file, and line 0 as an line number where the printing of the message was triggered. This is absolutely normal if the code is executed from within a handler, because there is no actual file associated with the handler. Therefore `$0` is set to `/dev/null` and that's what you see.

1.6.3 Preventing mod_perl Processes From Going Wild

See the sections "Non-Scheduled Emergency Log Rotation" and "All RAM Consumed"

1.6.4 Segfaults when using XML::Parser

If you have some of the processes segfault when using `XML::Parser` you should use

```
--disable-rule=EXPAT
```

during the Apache configuration step.

Starting from `mod_perl` version 1.23 this option is disabled by default.

1.6.5 My CGI/Perl Code Gets Returned as Plain Text Instead of Being Executed by the Webserver

See My CGI/Perl Code Gets Returned as Plain Text Instead of Being Executed by the Webserver.

1.6.6 Incorrect line number reporting in error/warn log messages

See Use of uninitialized value at (eval 80) line 12.

1.6.7 rwrite returned -1

This message happens when the client breaks the connection while your script is trying to write to the client. With Apache 1.3.x, you should only see the rwrite messages if LogLevel is set to debug.

There was a bug that reported this debug message regardless of the value of the LogLevel directive. It was fixed in mod_perl 1.19_01.

Generally LogLevel is either debug or info. debug logs everything, info is the next level, which doesn't include debug messages. You shouldn't use "debug" mode on your production server. At the moment there is no way to prevent users from aborting connections.

1.6.8 Can't upgrade that kind of scalar ...

Fixed in mod_perl 1.23.

1.6.9 caught SIGPIPE in process

```
[modperl] caught SIGPIPE in process 1234
[modperl] process 1234 going to Apache::exit with status...
```

That's the \$SIG{PIPE} handler installed by mod_perl/Apache::SIG, which is called if a connection times out or if the client presses the 'Stop' button. It gives you an opportunity to do cleanups if the script was aborted in the middle of its execution. See Handling the 'User pressed Stop button' case for more info.

If your mod_perl version is earlier than 1.17 you might also get the message in the following section...

1.6.10 Client hit STOP or Netscape bit it!

```
Client hit STOP or Netscape bit it!
Process 2493 going to Apache::exit with status=-2
```

You may see this message in mod_perl versions less than 1.17. See also caught SIGPIPE in process.

1.6.11 Global symbol "\$foo" requires explicit package name

The script below will print a warning like that above, moreover it will print the whole script as a part of the warning message:

```
#!/usr/bin/perl -w
use strict;
print "Content-type: text/html\n\n";
print "Hello $undefined";
```

The warning:


```

Global symbol "$undefined" requires
explicit package name at /usr/apps/foo/cgi/tmp.pl line 4.
    eval 'package Apache::ROOT::perl::tmp_2epl;
use Apache qw(exit);sub handler {
#line 1 /usr/apps/foo/cgi/tmp.pl
BEGIN {$^W = 1;}#!/usr/bin/perl -w
use strict;
print "Content-type: text/html\\n\\n";
print "Hello $undefined";

}
;' called at
/usr/apps/lib/perl5/site_perl/5.005/aix/Apache/Registry.pm
line 168
    Apache::Registry::compile('package
Apache::ROOT::perl::tmp_2epl;use Apache qw(exit);sub han...')
called at
/usr/apps/lib/perl5/site_perl/5.005/aix/Apache/Registry.pm
line 121
    Apache::Registry::handler('Apache=SCALAR(0x205026c0)')
called at /usr/apps/foo/cgi/tmp.pl line 4
    eval {...} called at /usr/apps/foo/cgi/tmp.pl line 4
[Sun Nov 15 15:15:30 1998] [error] Undefined subroutine
&Apache::ROOT::perl::tmp_2epl::handler called at
/usr/apps/lib/perl5/site_perl/5.005/aix/Apache/Registry.pm
line 135.

[Sun Nov 15 15:15:30 1998] [error] Goto undefined subroutine
&Apache::Constants::SERVER_ERROR at
/usr/apps/lib/perl5/site_perl/5.005/aix/Apache/Constants.pm
line 23.

```

The error is simple to fix. When you use the `use strict;` pragma (and you should...), Perl will insist that all variables are defined before being used, so the error will not arise.

The bad thing is that sometimes the whole script (possibly, thousands of lines) is printed to the *error_log* file as code that the server has tried to `eval()` uate.

May be you have a `$SIG{__DIE__}` handler installed (`Carp::confess()`?). If so that's what's expected.

You might wish to try something more terse such as `"local $SIG{__WARN__} = \&Carp::cluck;"` The `confess` method is *very* verbose and will tell you more than you might wish to know including full source.

1.6.12 Use of uninitialized value at (eval 80) line 12.

Your code includes some undefined variable that you have used as if it was already defined and initialized. For example:

```

$params = $q->param('test');
print $params;

```

1.6.13 Undefined subroutine &Apache::ROOT::perl::test_2epl::some_function called at

vs.

```
$param = $q->param('test') || '';  
print $param;
```

In the second case, `$param` will always be *defined*, either with `$q->param('test')`'s return value or the default value (`''` empty string in our example).

Also read about Finding the Line Which Triggered the Error or Warning.

1.6.13 Undefined subroutine &Apache::ROOT::perl::test_2epl::some_function called at

See Names collisions with Modules and libs.

1.6.14 Callback called exit

Callback called exit is just a generic message when some unrecoverable error occurs inside Perl during `perl_call_sv()` (which `mod_perl` uses to invoke all handler subroutines. Such problems seem to occur far less with 5.005_03 than 5.004.

Sometimes you discover that your server is not responding and its `error_log` has filled up the remaining space on the file system. When you get to see the contents of the `error_log` -- it includes millions of lines, like:

```
Callback called exit at -e line 33, <HTML> chunk 1.
```

Why the looping?

Perl can get *very* confused inside an infinite loop in your code. It doesn't necessarily mean that your code did call `exit()`. Perl's malloc went haywire and called `croak()`, but no memory is left to properly report the error, so Perl is stuck in a loop writing that same message to `stderr`.

Perl 5.005+ plus is recommended for its improved `malloc.c` and other features that improve `mod_perl` and are turned on by default.

See also Out of memory

1.6.15 Out of memory!

If something goes really wrong with your code, Perl may die with an "Out of memory!" message and/or "Callback called exit". Common causes of this are never-ending loops, deep recursion, or calling an undefined subroutine. Here's one way to catch the problem: See Perl's INSTALL document for this item:

```
=item -DPERL_EMERGENCY_SBRK
```

If PERL_EMERGENCY_SBRK is defined, running out of memory need not be a fatal error: a memory pool can allocated by assigning to the special variable `$_M`. See `perlvar(1)` for more details.

If you compile with that option and add `'use Apache::Debug level => 4;'` to your Perl script, it will allocate the `$_M` emergency pool and the `$SIG{__DIE__}` handler will call `Carp::confess`, giving you a stack trace which should reveal where the problem is. See the `Apache::Resource` module for ways to control httpd processes.

Note that Perl 5.005 and later have PERL_EMERGENCY_SBRK turned on by default.

The other trick is to have a startup script initialize `Carp::confess`, like so:

```
use Carp ();
eval { Carp::confess("init") };
```

this way, when the real problem happens, `Carp::confess` doesn't eat memory in the emergency pool (`$_M`).

1.6.16 server reached MaxClients setting, consider raising the MaxClients setting

See Choosing MaxClients.

1.6.17 syntax error at /dev/null line 1, near "line arguments:"

```
syntax error at /dev/null line 1, near "line arguments:"
Execution of /dev/null aborted due to compilation errors.
parse: Undefined error: 0
```

There is a chance that your `/dev/null` device is broken. Try:

```
% echo > /dev/null
```

Alternatively you should try to remove this special file and recreate it:

```
# rm /dev/null
# mknod /dev/null c 1 3
# chmod a+rw /dev/null
```

1.6.18 Can't call method "register_cleanup" (CGI.pm)

```
Can't call method "register_cleanup" on an
undefined value at /usr/lib/perl5/5.00503/CGI.pm line 263.
```

caused by this code snippet in *CGI.pm*:

```
if ($MOD_PERL) {
    Apache->request->register_cleanup(\&CGI::_reset_globals);
    undef $NPH;
}
```

One solution is to add to *httpd.conf*:

```
PerlPostReadRequestHandler 'sub { Apache->request(shift) }'
```

But even better, switch to `Apache::Cookie`:

```
use Apache;
use Apache::Cookie;

sub handler {
    my $r = shift;
    my $cookies = Apache::Cookie->new($r)->parse;
    my %bar = $cookies->{foo}->value;
}
```

1.6.19 *readdir() not working*

If `readdir()` either fails with an exception, or in the list context it returns the correct number of items but each item as an empty string, you have a binary compatibility between `mod_perl` and Perl problem. Most likely the two have been built against different *glibc* versions, which have incompatible `struct dirent`.

To solve this problem rebuild `mod_perl` and Perl against the same *glibc* version or get new binary packages built against the same *glibc* version.

1.7 Shutdown and Restart

1.7.1 *Evil things might happen when using PerlFreshRestart*

`PerlFreshRestart` affects the graceful restart process for non-DSO `mod_perl` builds. If you have `mod_perl` enabled Apache built as DSO and you restart it, the whole Perl interpreter is completely torn down (`perl_destruct()`) and restarted. The value of `PerlFreshRestart` is irrelevant at this point.

Unfortunately, not all perl modules are robust enough to survive reload. For them this is an unusual situation. `PerlFreshRestart` does not much more than:

```
while (my($k,$v) = each %INC) {
    delete $INC{$k};
    require $k;
}
```

Besides that, it flushes the `Apache::Registry` cache, and empties any dynamic stacked handlers (e.g. `PerlChildInitHandler`).

Some users, who had turned `PerlFreshRestart` **On**, reported having segfaults, others have seen no problems starting the server, no errors written to the logs, but no server running after a restart. Most of the problems have gone away when it was turned **Off**.

It doesn't mean that you shouldn't use it, if it works for you. Just beware of the dragons...

1.7.2 Constant subroutine XXX redefined

That's a mandatory warning inside Perl which happens only if you modify your script and Apache::Registry reloads it. Perl is warning you that the subroutine(s) were redefined. It is mostly harmless. If you don't like seeing these warnings, just kill -USR1 (graceful restart) Apache when you modify your scripts.

You aren't supposed to see these warnings if you don't modify the code with perl 5.004_05 or 5.005+ and higher. If you still experience a problem with code within a CGI script, moving all the code into a module (or a library) and require()ing it should solve the problem.

1.7.3 Can't undef active subroutine

```
Can't undef active subroutine at
/usr/apps/lib/perl5/site_perl/5.005/aix/Apache/Registry.pm line 102.
Called from package Apache::Registry, filename
/usr/apps/lib/perl5/site_perl/5.005/aix/Apache/Registry.pm, line 102
```

This problem is caused when a client drops the connection while httpd is in the middle of a write. httpd times out, sending a SIGPIPE, and Perl (in that child) is stuck in the middle of its eval context. This is fixed by the Apache::SIG module which is called by default. This should not happen unless you have code that is messing with \$SIG{PIPE}. It's also triggered only when you've changed your script on disk and mod_perl is trying to reload it.

1.7.4 [warn] child process 30388 did not exit, sending another SIGHUP

From mod_perl.pod: With Apache versions 1.3.0 and higher, mod_perl will call the perl_destruct() Perl API function during the child exit phase. This will cause proper execution of END blocks found during server startup as well as invoking the DESTROY method on global objects which are still alive. It is possible that this operation may take a long time to finish, causing problems during a restart. If your code does not contain any END blocks or DESTROY methods which need to be run during child server shutdown, this destruction can be avoided by setting the PERL_DESTRUCT_LEVEL environment variable to -1. Be aware that 'your code' includes any modules you use and *they* may well have DESTROY and END blocks...

1.7.5 Processes Get Stuck on Graceful Restart

If you see a process stuck in "G" (Gracefully finishing) after a doing a graceful restart (sending kill -SIGUSR1) it means that the process is hanging in perl_destruct() while trying to cleanup. This cleanup normally isn't a requirement, you can disable it by setting the PERL_DESTRUCT_LEVEL environment variable to -1. See the section "Speeding up the Apache Termination and Restart" for more information.

1.7.6 httpd keeps on growing after each restart

See the *HUP Signal* explanation at the section: Server Stopping and Restarting

1.8 OS Specific Notes

1.8.1 RedHat Linux

1.8.1.1 RH 8 and 9 Locale Issues

RedHat 8 and 9 ship Perl with a locale setting that breaks perl. To solve the problem either change the locale to en_US.ISO8859-1 or simply remove the UTF8 part.

For more information refer to: <http://twiki.org/cgi-bin/view/Codev/UsingPerl58OnRedHat8>
http://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=87682

1.8.2 OpenBSD

1.8.2.1 Issues Caused by httpd being chroot'ed

Since OpenBSD 3.2, the Apache `httpd(8)` server has been `chroot(2)`ed by default.

A `mod_perl` enabled OpenBSD `httpd` will NOT work as is. A proper `chroot(2)` environment must be setup for the default chrooted behavior to work. The `-u` option to `httpd` disables the chroot behavior, and returns the `httpd` to the expanded "unsecure" behavior. See the OpenBSD `httpd(8)` man page.

For more information see *section 10.16 of the OpenBSD FAQ*.
<http://www.openbsd.org/faq/faq10.html#httpdchroot>.

1.8.3 Win FU

1.8.3.1 Apache::DBI

`Apache::DBI` causes the server to exit when it starts up, with:

```
[Mon Oct 25 15:06:11 1999] file .\main\http_main.c, line 5890,  
assertion "start_mutex" failed
```

Solution: build `mod_perl` with `PERL_STARTUP_DONE_CHECK` set (e.g. insert

```
#define PERL_STARTUP_DONE_CHECK 1
```

at the top of `mod_perl.h` or add it to the defines in the MSVC++ and similar applications' Options dialog).

Apache loads all Apache modules twice, to make sure the server will successfully restart when asked to. This flag disables all `PerlRequire` and `PerlModule` statements on the first load, so they can succeed on the second load. Without that flag, the second load fails.

1.8.4 HP-UX

1.8.4.1 `Apache::Status` Dumps Core Under HP-UX 10.20

HP-UX 10.20 may dump core when accessing `perl-status?sig`. This issue is known to happen when perl's `./Configure` doesn't detect that `SIGRTMAX` is defined, but not implemented on that platform.

One solution is to upgrade to a recent version of Perl (5.8.0?) that properly detects the implementation of that signal.

Another solution is to modify `Apache/Status.pm` to skip that broken signal by replacing the line:

```
sort keys %SIG),
```

with:

```
sort grep { $_ ne 'RTMAX' } keys %SIG),
```

1.9 Problematic Perl Modules

Here is the list of Perl Modules that are known to have problems under `mod_perl` 1.0 and possible workarounds, solutions.

- **`IPC::Open*`**

use `IPC::Run` instead. It provides the same functionality as the `IPC::Open*` family and more... and it works fine with `mod_perl`.

1.10 Maintainers

Maintainer is the person(s) you should contact with updates, corrections and patches.

- Stas Bekman <stas (at) stason.org>

1.11 Authors

- Stas Bekman <stas (at) stason.org>

Only the major authors are listed above. For contributors see the `Changes` file.

Table of Contents:

1	Warnings and Errors Troubleshooting Index	1
1.1	Description	2
1.2	General Advice	2
1.3	Building and Installation	2
1.4	Configuration and Startup	2
1.4.1	SegFaults During Startup	2
1.4.2	libexec/libperl.so: open failed: No such file or directory	3
1.4.3	install_driver(Oracle) failed: Can't load '.../DBD/Oracle/Oracle.so' for module DBD::Oracle	3
1.4.4	Invalid command 'PerlHandler'...	4
1.4.5	symbol __floatdisf: referenced symbol not found	4
1.4.6	RegistryLoader: Translation of uri [...] to filename failed	5
1.4.7	"Apache.pm failed to load!"	5
1.5	Code Parsing and Compilation	5
1.5.1	Value of \$x will not stay shared at - line 5	5
1.5.2	Value of \$x may be unavailable at - line 5.	5
1.5.3	Can't locate loadable object for module XXX	5
1.5.4	Can't locate object method "get_handlers"...	6
1.5.5	Missing right bracket at line ...	6
1.5.6	Can't load '.../auto/DBI/DBI.so' for module DBI	6
1.6	Runtime	7
1.6.1	"exit signal Segmentation fault (11)" with mysql	7
1.6.2	foo ... at /dev/null line 0	7
1.6.3	Preventing mod_perl Processes From Going Wild	7
1.6.4	Segfaults when using XML::Parser	7
1.6.5	My CGI/Perl Code Gets Returned as Plain Text Instead of Being Executed by the Webserver	7
1.6.6	Incorrect line number reporting in error/warn log messages	7
1.6.7	rwrite returned -1	8
1.6.8	Can't upgrade that kind of scalar ...	8
1.6.9	caught SIGPIPE in process	8
1.6.10	Client hit STOP or Netscape bit it!	8
1.6.11	Global symbol "\$foo" requires explicit package name	8
1.6.12	Use of uninitialized value at (eval 80) line 12.	9
1.6.13	Undefined subroutine &Apache::ROOT::perl::test_2epl::some_function called at	10
1.6.14	Callback called exit	10
1.6.15	Out of memory!	10
1.6.16	server reached MaxClients setting, consider raising the MaxClients setting	11
1.6.17	syntax error at /dev/null line 1, near "line arguments:"	11
1.6.18	Can't call method "register_cleanup" (CGI.pm)	11
1.6.19	readdir() not working	12
1.7	Shutdown and Restart	12
1.7.1	Evil things might happen when using PerlFreshRestart	12
1.7.2	Constant subroutine XXX redefined	13

Table of Contents:

1.7.3 Can't undef active subroutine	13
1.7.4 [warn] child process 30388 did not exit, sending another SIGHUP	13
1.7.5 Processes Get Stuck on Graceful Restart	13
1.7.6 httpd keeps on growing after each restart	14
1.8 OS Specific Notes	14
1.8.1 RedHat Linux	14
1.8.1.1 RH 8 and 9 Locale Issues	14
1.8.2 OpenBSD	14
1.8.2.1 Issues Caused by httpd being chroot'ed	14
1.8.3 Win FU	14
1.8.3.1 Apache::DBI	14
1.8.4 HP-UX	15
1.8.4.1 Apache::Status Dumps Core Under HP-UX 10.20	15
1.9 Problematic Perl Modules	15
1.10 Maintainers	15
1.11 Authors	15