

1 Installing mod_perl 2.0

1.1 Description

This chapter provides an in-depth mod_perl 2.0 installation coverage.

1.2 Prerequisites

Before building mod_perl 2.0 you need to have its prerequisites installed. If you don't have them, download and install them first, using the information in the following sections. Otherwise proceed directly to the mod_perl building instructions.

The mod_perl 2.0 prerequisites are:

- **Apache**

Apache 2.0 is required. mod_perl 2.0 **does not** work with Apache 1.3.

Dynamic (DSO) mod_perl build requires Apache 2.0.47 or higher. Static build requires Apache 2.0.51 or higher.

- **Perl**

- **Prefork MPM**

Requires at least Perl version 5.6.1.

You don't need to have threads-support enabled in Perl. If you do have it, it **must** be *ithreads* and not *5005threads*! If you have:

```
% perl5.8.0 -V:use5005threads
use5005threads='define' ;
```

you must rebuild Perl without threads enabled or with `-Dusethreads`. Remember that threads-support slows things down and on some platforms it's unstable (e.g., FreeBSD), so don't enable it unless you really need it.

- **Threaded MPMs**

Require at least Perl version 5.8.0 with *ithreads* support built-in. That means that it should report:

```
% perl5.8.0 -V:useithreads -V:usemultiplicity
useithreads='define' ;
usemultiplicity='define' ;
```

If that's not what you see rebuild Perl with `-Dusethreads`.

- **Static prefork build**

Perl with ithreads support version 5.6.1 or higher

Perl without ithreads support version 5.8.2 or higher

- **Static non-prefork build**

Perl with ithreads support version 5.8.0 or higher

- **threads.pm**

If you want to run applications that take benefit of Perl's *threads.pm* Perl version 5.8.1 or higher w/ithreads enabled is required. Perl 5.8.0's *threads.pm* doesn't work with mod_perl 2.0.

- **CPAN Perl Modules**

The mod_perl 2.0 test suite has several requirements on its own. If you don't satisfy them, the tests depending on these requirements will be skipped, which is OK, but you won't get to run these tests and potential problems, which may exhibit themselves in your own code, could be missed. We don't require them from `Makefile.PL`, which could have been automated the requirements installation, in order to have less dependencies to get mod_perl 2.0 installed.

Also if your code uses any of these modules, chances are that you will need to use at least the version numbers listed here.

- **CGI.pm 3.11**
- **Compress::Zlib 1.09**

Though the easiest way to satisfy all the dependencies is to install `Bundle::Apache2` available from CPAN.

1.2.1 Downloading Stable Release Sources

If you are going to install mod_perl on a production site, you want to use the officially released stable components. Since the latest stable versions change all the time you should check for the latest stable version at the listed below URLs:

- **Perl**

Download from: <http://cpan.org/src/README.html>

This direct link which symlinks to the latest release should work too:
<http://cpan.org/src/stable.tar.gz>.

For the purpose of examples in this chapter we will use the package named *perl-5.8.x.tar.gz*, where *x* should be replaced with the real version number.

- **Apache**

Download from: <http://www.apache.org/dist/httpd/>

For the purpose of examples in this chapter we will use the package named *httpd-2.x.xx.tar.gz*, where *x.xx* should be replaced with the real version number.

1.2.2 Getting Bleeding Edge Sources

If you really know what you are doing you can use the cvs/svn versions of the components. Chances are that you don't want to them on a production site. You have been warned!

● Perl

The cutting edge version of Perl (aka *bleadperl* or *bleedperl*) is only generally available through an rsync repository maintained by ActiveState:

```
# (--delete to ensure a clean state)
% rsync -acvz --delete --force \
  rsync://public.activestate.com/perl-current/ perl-current
```

If you are re-building Perl after rsync-ing, make sure to cleanup first:

```
% make distclean
```

before running `./Configure`.

You'll also want to install (at least) LWP if you want to fully test `mod_perl`. You can install LWP with CPAN `.pm` shell:

```
% perl -MCPAN -e 'install("LWP")'
```

For more details on *bleadperl*, see <http://dev.perl.org/perl5/source.html>.

● Apache

See Development `mod_perl 2.0` Source Distribution.

1.2.3 Configuring and Installing Prerequisites

If you don't have the prerequisites installed yet, install them now.

1.2.3.1 Perl

```
% cd perl-5.8.x
% ./Configure -des
```

If you need the threads support, run:

```
% ./Configure -des -Dusethreads
```

Most likely you don't want perl-support for threads enabled, in which case pass: `-Usethreads` instead of `-Dsethreads`.

If you want to debug mod_perl segmentation faults, add the following `./Configure` options:

```
-Doptimize='-g' -Dusedevel
```

Now build it:

```
% make && make test && make install
```

1.2.3.2 Apache

You need to have Apache built and installed prior to building mod_perl, only if you intend build a DSO mod_perl. If you intend to build a statically linked Apache+mod_perl, you only need to have the Apache source available (mod_perl will build and install Apache for you), you should skip this step.

```
% cd httpd-2.x.xx
% ./configure --prefix=$HOME/httpd/prefork --with-mpm=prefork
% make && make install
```

Starting from 2.0.49, the Apache logging API escapes everything that goes to `error_log`, therefore if you're annoyed by this feature during the development phase (as your error messages will be all messed up) you can disable the escaping during the Apache build time:

```
% CFLAGS="-DAP_UNSAFE_ERROR_LOG_UNESCAPED" ./configure ...
```

Do **not** use that CFLAGS in production unless you know what you are doing.

1.3 Installing mod_perl from Binary Packages

As of this writing only the binaries for the Win32 platform are available, kindly prepared and maintained by Randy Kobes. See the documentation on Win32 binaries for details.

Some RPM packages can be found using rpmfind services, e.g.:

http://www.rpmfind.net/linux/rpm2html/search.php?query=mod_perl&submit=Search+... However if you have problems using them, you have to contact those who have created them.

1.4 Installing mod_perl from Source

Building from source is the best option, because it ensures a binary compatibility with Apache and Perl. However it's possible that your distribution provides a solid binary mod_perl 2.0 package.

For Win32 specific details, see the documentation on Win32 installation.

1.4.1 Downloading the mod_perl Source

First download the mod_perl source.

- **Stable Release**

Download from <http://perl.apache.org/download/> or your favorite CPAN mirror.

This direct link which symlinks to the latest release should work too:
http://perl.apache.org/dist/mod_perl-2.0-current.tar.gz.

For the purpose of examples in this chapter we will use the package named *mod_perl-2.x.x.tar.gz*, where *x.x* should be replaced with the real version number.

Open the package with:

```
% tar -xvzf mod_perl-2.x.x.tar.gz
```

or an equivalent command.

- **Development Version**

See Development mod_perl 2.0 Source Distribution.

1.4.2 Configuring mod_perl

To build mod_perl, you **must** also use the same compiler that Perl was built with. You can find that out by running `perl -V` and looking at the `Compiler:` section.

Like any other Perl module, mod_perl is configured via the *Makefile.PL* file, but requires one or more configuration options:

```
% cd modperl-2.x.x
% perl Makefile.PL <options>
```

where *options* is an optional list of key/value pairs. These options can include all the usual options supported by `ExtUtils::MakeMaker` (e.g., `PREFIX`, `LIB`, etc.).

The following sections give the details about all the available options, but let's mention first an important one.

Configuration options are discussed in Build Options.

1.4.2.1 Dynamic mod_perl

Before you proceed, make sure that Apache 2.0 has been built and installed. mod_perl **cannot** be built before that.

It seems that most users use pre-packaged Apache installation, most of which tend to spread the Apache files across many directories (i.e. not using `--enable-layout=Apache`, which puts all the files under the same directory). If Apache 2.0 files are spread under different directories, you need to use at least the `MP_APXS` option, which should be set to a full path to the `apxs` executable. For example:

```
% perl Makefile.PL MP_APXS=/path/to/apxs
```

For example RedHat Linux system installs the `httpd` binary, the `apxs` and `apr-config` scripts (the latter two are needed to build `mod_perl`) all in different locations, therefore they configure `mod_perl 2.0` as:

```
% perl Makefile.PL MP_APXS=/path/to/apxs \  
MP_APR_CONFIG=/another/path/to/apr-config <other options>
```

However a correctly built Apache shouldn't require the `MP_APR_CONFIG` option, since `MP_APXS` should provide the location of this script.

If however all Apache 2.0 files were installed under the same directory, `mod_perl 2.0`'s build only needs to know the path to that directory, passed via the `MP_AP_PREFIX` option:

```
% perl Makefile.PL MP_AP_PREFIX=$HOME/httpd/prefork
```

1.4.2.2 Static mod_perl

Before you proceed make sure that Apache 2.0 has been downloaded and extracted. `mod_perl` **cannot** be built before that.

If this is an `svn` checkout and not an official distribution tarball, you need to first run:

```
% cd httpd-2.0  
% ./buildconf
```

To enable statically linking `mod_perl` into Apache, use the `MP_USE_STATIC` flag like this:

```
% perl Makefile.PL MP_USE_STATIC=1 \  
MP_AP_PREFIX=$HOME/src/httpd-2.x \  
MP_AP_CONFIGURE="--with-mpm=prefork"
```

`MP_AP_PREFIX` **must** point to an extracted Apache 2.0 source tree.

This will configure Apache by passing `MP_AP_CONFIGURE` to Apache's `./configure` script.

Here is an example:

```
% cd ~/src  
% tar -xvzf perl-5.8.x.tar.gz  
% cd perl-5.8.x  
% ./Configure -des  
% make install  
% cd ..  
% tar -xvzf httpd-2.0.xx.tar.gz  
% tar -xvzf mod_perl-2.x.x.tar.gz
```

```
% perl5.8.x Makefile.PL \  
MP_USE_STATIC=1 \  
MP_AP_PREFIX="$HOME/src/httpd-2.0.xx" \  
MP_AP_CONFIGURE="--with-mpm=prefork" \  
% make \  
% make test \  
% make install \  
% ./httpd -l | grep perl \  
mod_perl.c
```

1.4.3 mod_perl Build Options

1.4.3.1 Boolean Build Options

The following options are boolean and can be set with `MP_XXX=1` or unset with `MP_XXX=0`, where `XXX` is the name of the option.

1.4.3.1.1 MP_PROMPT_DEFAULT

Accept default values for all would-be prompts.

1.4.3.1.2 MP_GENERATE_XS

Generate XS code from parsed source headers in `xs/tables/$httpd_version`. Default is 1, set to 0 to disable.

1.4.3.1.3 MP_USE_DSO

Build `mod_perl` as a DSO (`mod_perl.so`). This is the default.

1.4.3.1.4 MP_USE_STATIC

Build static `mod_perl` (`mod_perl.a`).

1.4.3.1.5 MP_STATIC_EXTS

Build `Apache2::*.xs` as static extensions.

1.4.3.1.6 MP_USE_GTOP

Link with `libgtop` and enable `libgtop` reporting.

1.4.3.1.7 MP_COMPAT_1X

`MP_COMPAT_1X=1` or a lack of it enables several `mod_perl` 1.0 back-compatibility features, which are deprecated in `mod_perl` 2.0. It's enabled by default, but can be disabled with `MP_COMPAT_1X=0` during the build process.

When this option is disabled, the following things will happen:

- Deprecated special variable, `$Apache2::__T` won't be available. Use `${^TAINT}` instead.
- `$ServerRoot` and `$ServerRoot/lib/perl` won't be appended to `@INC`. Instead use:

```
PerlSwitches -I/path/to/server -I/path/to/server/lib/perl
```

in *httpd.conf* or:

```
use Apache2::ServerUtil ();
use File::Spec::Functions qw(catfile);
push @INC, catfile Apache2::ServerUtil::server_root, "";
push @INC, catfile Apache2::ServerUtil::server_root, "lib/perl";
```

in *startup.pl*.

- The following deprecated configuration directives won't be recognized by Apache:

```
PerlSendHeader
PerlSetupEnv
PerlHandler
PerlTaintCheck
PerlWarn
```

Use their 2.0 equivalents instead.

1.4.3.1.8 MP_DEBUG

Turn on debugging (`-g -lperl`) and tracing.

1.4.3.1.9 MP_MAINTAINER

Enable maintainer compile mode, which sets `MP_DEBUG=1` and adds the following `gcc` flags:

```
-DAP_DEBUG -Wall -Wmissing-prototypes -Wstrict-prototypes \
-Wmissing-declarations \
```

If `gcc` version 3.3.2+ is found, not compiling on OpenBSD, and `-Wdeclaration-after-statement` is not already part of the `gcc` flags add it.

To use this mode Apache must be build with `--enable-maintainer-mode`.

1.4.3.1.10 MP_TRACE

Enable tracing

1.4.3.2 Non-Boolean Build Options

set the non-boolean options with `MP_XXX=value`.

1.4.3.2.1 *MP_APXS*

Path to `apxs`. For example if you've installed Apache 2.0 under `/home/httpd/httpd-2.0` as DSO, the default location would be `/home/httpd/httpd-2.0/bin/apxs`.

1.4.3.2.2 *MP_AP_CONFIGURE*

The command-line arguments to pass to `httpd`'s configure script.

1.4.3.2.3 *MP_AP_PREFIX*

Apache installation prefix, under which the `include/` directory with Apache C header files can be found. For example if you've installed Apache 2.0 in directory `\Apache2` on Win32, you should use:

```
MP_AP_PREFIX=\Apache2
```

If Apache is not installed yet, you can point to the Apache 2.0 source directory, but only after you've built or configured Apache in it. For example:

```
MP_AP_PREFIX=/home/stas/apache.org/httpd-2.0
```

Though in this case `make test` won't automatically find `httpd`, therefore you should run `t/TEST` instead and pass the location of `apxs` or `httpd`, e.g.:

```
% t/TEST -apxs /home/stas/httpd/prefork/bin/apxs
```

or

```
% t/TEST -httpd /home/stas/httpd/prefork/bin/httpd
```

1.4.3.2.4 *MP_AP_DESTDIR*

This option exists to make the lives of package maintainers easier. If you aren't a package manager you should not need to use this option.

Apache installation destination directory. This path will be prefixed to the installation paths for all Apache-specific files during `make install`. For instance, if Apache modules are normally installed into `/path/to/httpd-2.0/modules/` and `MP_AP_DESTDIR` is set to `/tmp/foo`, the `mod_perl.so` will be installed in:

```
/tmp/foo/path/to/httpd-2.0/modules/mod_perl.so
```

1.4.3.2.5 *MP_APR_CONFIG*

If APR wasn't installed under the same file tree as httpd, you may need to tell the build process where it can find the executable `apr-config`, which can then be used to figure out where the `apr` and `aprutil` *include/* and *lib/* directories can be found.

1.4.3.2.6 *MP_CCOPTS*

Add to compiler flags, e.g.:

```
MP_CCOPTS=-Werror
```

(Notice that `-Werror` will work only with the Perl version 5.7 and higher.)

1.4.3.2.7 *MP_OPTIONS_FILE*

Read build options from given file. e.g.:

```
MP_OPTIONS_FILE=~/.my_mod_perl2_opts
```

1.4.3.2.8 *MP_APR_LIB*

On Win32, in order to build the APR and APR::* modules so as to be independent of mod_perl.so, a static library is first built containing the needed functions these modules link into. The option

```
MP_APR_LIB=aprext
```

specifies the name that this library has. The default used is `aprext`. This option has no effect on platforms other than Win32, as they use a different mechanism to accomplish the decoupling of APR and APR::* from mod_perl.so.

1.4.3.3 mod_perl-specific Compiler Options

1.4.3.3.1 *-DMP_IOBUFSIZE*

Change the default mod_perl's 8K IO buffer size, e.g. to 16K:

```
MP_CCOPTS=-DMP_IOBUFSIZE=16384
```

1.4.3.4 mod_perl Options File

Options can also be specified in the file `makepl_args.mod_perl2` or `.makepl_args.mod_perl2`. The file can be placed under `$ENV{HOME}`, the root of the source package or its parent directory. So if you unpack the mod_perl source into `/tmp/mod_perl-2.x/` and your home is `/home/foo/`, the file will be searched in:

```
/tmp/mod_perl-2.x/makepl_args.mod_perl2
/tmp/makepl_args.mod_perl2
/home/foo/makepl_args.mod_perl2
/tmp/mod_perl-2.x/.makepl_args.mod_perl2
/tmp/.makepl_args.mod_perl2
/home/foo/.makepl_args.mod_perl2
```

If the file specified in `MP_OPTIONS_FILE` is found the *makepl_args.mod_perl2* will be ignored.

Options specified on the command line override those from *makepl_args.mod_perl2* and those from `MP_OPTIONS_FILE`.

If your terminal supports colored text you may want to set the environment variable `APACHE_TEST_COLOR` to 1 to enable the colored tracing which makes it easier to tell the reported errors and warnings, from the rest of the notifications.

1.4.4 Re-using Configure Options

Since `mod_perl` remembers what build options were used to build it if first place, you can use this knowledge to rebuild itself using the same options. Simply `chdir(1)` to the `mod_perl` source directory and run:

```
% cd modperl-2.x.
% perl -MApache2::Build -e rebuild
```

1.4.5 Compiling mod_perl

Next stage is to build `mod_perl`:

```
% make
```

1.4.6 Testing mod_perl

When `mod_perl` has been built, it's very important to test that everything works on your machine:

```
% make test
```

If something goes wrong with the test phase and want to figure out how to run individual tests and pass various options to the test suite, see the corresponding sections of the bug reporting guidelines or the `Apache::Test Framework` tutorial.

1.4.7 Installing mod_perl

Once the test suite has passed, it's a time to install `mod_perl`.

```
% make install
```

If you install mod_perl system wide, you probably need to become *root* prior to doing the installation:

```
% su
# make install
```

1.5 If Something Goes Wrong

If something goes wrong during the installation, try to repeat the installation process from scratch, while verifying all the steps with this document.

If the problem persists report the problem.

1.6 Maintainers

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

- Stas Bekman [<http://stason.org/>]

1.7 Authors

- Stas Bekman [<http://stason.org/>]
- Doug MacEachern <dougm (at) covalent.net>

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

Table of Contents:

1	Installing mod_perl 2.0	1
1.1	Description	2
1.2	Prerequisites	2
1.2.1	Downloading Stable Release Sources	3
1.2.2	Getting Bleeding Edge Sources	4
1.2.3	Configuring and Installing Prerequisites	4
1.2.3.1	Perl	4
1.2.3.2	Apache	5
1.3	Installing mod_perl from Binary Packages	5
1.4	Installing mod_perl from Source	5
1.4.1	Downloading the mod_perl Source	6
1.4.2	Configuring mod_perl	6
1.4.2.1	Dynamic mod_perl	6
1.4.2.2	Static mod_perl	7
1.4.3	mod_perl Build Options	8
1.4.3.1	Boolean Build Options	8
1.4.3.1.1	MP_PROMPT_DEFAULT	8
1.4.3.1.2	MP_GENERATE_XS	8
1.4.3.1.3	MP_USE_DSO	8
1.4.3.1.4	MP_USE_STATIC	8
1.4.3.1.5	MP_STATIC_EXTS	8
1.4.3.1.6	MP_USE_GTOP	8
1.4.3.1.7	MP_COMPAT_1X	8
1.4.3.1.8	MP_DEBUG	9
1.4.3.1.9	MP_MAINTAINER	9
1.4.3.1.10	MP_TRACE	9
1.4.3.2	Non-Boolean Build Options	10
1.4.3.2.1	MP_APXS	10
1.4.3.2.2	MP_AP_CONFIGURE	10
1.4.3.2.3	MP_AP_PREFIX	10
1.4.3.2.4	MP_AP_DESTDIR	10
1.4.3.2.5	MP_APR_CONFIG	11
1.4.3.2.6	MP_CCOPTS	11
1.4.3.2.7	MP_OPTIONS_FILE	11
1.4.3.2.8	MP_APR_LIB	11
1.4.3.3	mod_perl-specific Compiler Options	11
1.4.3.3.1	-DMP_IOBUFSIZE	11
1.4.3.4	mod_perl Options File	11
1.4.4	Re-using Configure Options	12
1.4.5	Compiling mod_perl	12
1.4.6	Testing mod_perl	12
1.4.7	Installing mod_perl	12
1.5	If Something Goes Wrong	13
1.6	Maintainers	13

Table of Contents:

1.7 Authors	13
-------------	----