



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

***Rocky Enterprise Linux 9.2 Manual Pages on command 'ModPerl::Util.3pm'***

***\$ man ModPerl::Util.3pm***

libapache2-mod-perl2-2.0.12::doUserConlibapache2-mod-perl2-2.0.12::docs::api::ModPerl::Util(3pm)

NAME

ModPerl::Util - Helper mod\_perl Functions

Synopsis

```
use ModPerl::Util;

# e.g. PerlResponseHandler

$callback = ModPerl::Util::current_callback;

# exit w/o killing the interpreter

ModPerl::Util::exit();

# untaint a string (do not use it! see the doc)

ModPerl::Util::untaint($string);

# removes a stash (.so, %INC{$stash}, etc.) as best as it can

ModPerl::Util::unload_package($stash);

# current perl's address (0x92ac760 or 0x0 under non-threaded perl)

ModPerl::Util::current_perl_id();
```

Description

"ModPerl::Util" provides mod\_perl utilities API.

API

"ModPerl::Util" provides the following functions and/or methods:

"current\_callback"

Returns the currently running callback name, e.g. 'PerlResponseHandler'.

```
$callback = ModPerl::Util::current_callback();
```

ret: \$callback ( string )

since: 2.0.00

"current\_perl\_id"

Return the memory address of the perl interpreter

```
$perl_id = ModPerl::Util::current_perl_id();
```

ret: \$perl\_id ( string )

Under threaded perl returns something like: 0x92ac760

Under non-thread perl returns 0x0

since: 2.0.00

Mainly useful for debugging applications running under threaded-perl.

"exit"

Terminate the request, but not the current process (or not the current Perl interpreter with threaded mpms).

```
ModPerl::Util::exit($status);
```

opt arg1: \$status ( integer )

The exit status, which as of this writing is ignored. (it's accepted to be compatible with the core "exit" function.)

ret: no return value

since: 2.0.00

Normally you will use the plain "exit()" in your code. You don't need to use

"ModPerl::Util::exit" explicitly, since mod\_perl overrides "exit()" by setting

"CORE::GLOBAL::exit" to "ModPerl::Util::exit". Only if you redefine "CORE::GLOBAL::exit"

once mod\_perl is running, you may want to use this function.

The original "exit()" is still available via "CORE::exit()".

"ModPerl::Util::exit" is implemented as a special "die()" call, therefore if you call it

inside "eval BLOCK" or "eval "STRING"", while an exception is being thrown, it is caught

by "eval". For example:

```
exit;
```

```
print "Still running";
```

will not print anything. But:

```
eval {
```

```
    exit;
```

```
}
```

```
print "Still running";
```

will print Still running. So you either need to check whether the exception is specific to "exit" and call "exit()" again:

```
use ModPerl::Const -compile => 'EXIT';

eval {
    exit;
}

exit if $@ && ref $@ eq 'APR::Error' && $@ == ModPerl::EXIT;

print "Still running";
```

or use "CORE::exit()":

```
eval {
    CORE::exit;
}

print "Still running";
```

and nothing will be printed. The problem with the latter is the current process (or a Perl Interpreter) will be killed; something that you really want to avoid under mod\_perl.

## "unload\_package"

Unloads a stash from the current Perl interpreter in the safest way possible.

```
ModPerl::Util::unload_package($stash);
```

arg1: \$stash ( string )

The Perl stash to unload. e.g. "MyApache2::MyData".

ret: no return value

since: 2.0.00

Unloading a Perl stash (package) is a complicated business. This function tries very hard to do the right thing. After calling this function, it should be safe to "use()" a new version of the module that loads the wiped package.

References to stash elements (functions, variables, etc.) taken from outside the unloaded package will still be valid.

This function may wipe off things loaded by other modules, if the latter have inserted things into the \$stash it was told to unload.

If a stash had a corresponding XS shared object (.so) loaded it will be unloaded as well.

If the stash had a corresponding entry in %INC, it will be removed from there.

"unload\_package()" takes care to leave sub-stashes intact while deleting the requested stash. So for example if "CGI" and "CGI::Carp" are loaded, calling "unload\_package("CGI")"

won't affect "CGI::Carp".

"untaint"

Untaint the variable, by turning its tainted SV flag off (used internally).

```
ModPerl::Util::untaint($tainted_var);
```

arg1: \$tainted\_var (scalar)

ret: no return value

\$tainted\_var is untainted.

since: 2.0.00

Do not use this function unless you know what you are doing. To learn how to properly untaint variables refer to the perlsec manpage.

See Also

mod\_perl 2.0 documentation.

Copyright

mod\_perl 2.0 and its core modules are copyrighted under The Apache Software License, Version 2.0.

Authors

The mod\_perl development team and numerous contributors.

perl v5.34.0

libapache2-mod-perl2-2.0.12::docs::api::ModPerl::Util(3pm)