# Apache Module mod_ext_filter

| | |
|---|---|
| **Description:** | Pass the response body through an external program before delivery to the client |
| **Status:** | Extension |
| **Module Identifier:** | ext_filter_module |
| **Source File:** | mod_ext_filter.c |

## Summary

`mod_ext_filter` presents a simple and familiar programming model for filters[1]. With this module, a program which reads from stdin and writes to stdout (i.e., a Unix-style filter command) can be a filter for Apache. This filtering mechanism is much slower than using a filter which is specially written for the Apache API and runs inside of the Apache server process, but it does have the following benefits:

- the programming model is much simpler
- any programming/scripting language can be used, provided that it allows the program to read from standard input and write to standard output
- existing programs can be used unmodified as Apache filters

Even when the performance characteristics are not suitable for production use, `mod_ext_filter` can be used as a prototype environment for filters.

## Topics



## Directives



## See also

- Filters[1]

# Examples

## Generating HTML from some other type of response

```
# mod_ext_filter directive to define a filter
# to HTML-ize text/c files using the external
# program /usr/bin/enscript, with the type of
# the result set to text/html
ExtFilterDefine c-to-html mode=output \
  intype=text/c outtype=text/html \
  cmd="/usr/bin/enscript --color -W html -Ec -o - -"

<Directory "/export/home/trawick/apacheinst/htdocs/c">
  # core directive to cause the new filter to
  # be run on output
  SetOutputFilter c-to-html

  # mod_mime directive to set the type of .c
  # files to text/c
  AddType text/c .c

  # mod_ext_filter directive to set the debug
```

```
  # level just high enough to see a log message
  # per request showing the configuration in force
  ExtFilterOptions DebugLevel=1
</Directory>
```

## Implementing a content encoding filter

Note: this gzip example is just for the purposes of illustration. Please refer to `mod_deflate` for a practical implementation.

```
# mod_ext_filter directive to define the external filter
ExtFilterDefine gzip mode=output cmd=/bin/gzip

<Location /gzipped>
  # core directive to cause the gzip filter to be
  # run on output
  SetOutputFilter gzip

  # mod_header directive to add
  # "Content-Encoding: gzip" header field
  Header set Content-Encoding gzip
</Location>
```

## Slowing down the server

```
# mod_ext_filter directive to define a filter
# which runs everything through cat; cat doesn't
# modify anything; it just introduces extra pathlength
# and consumes more resources
ExtFilterDefine slowdown mode=output cmd=/bin/cat \
  preservescontentlength

<Location />
  # core directive to cause the slowdown filter to
  # be run several times on output
  #
  SetOutputFilter slowdown;slowdown;slowdown
</Location>
```

## Using sed to replace text in the response

```
# mod_ext_filter directive to define a filter which
# replaces text in the response
#
ExtFilterDefine fixtext mode=output intype=text/html \
  cmd="/bin/sed s/verdana/arial/g"

<Location />
  # core directive to cause the fixtext filter to
  # be run on output
  SetOutputFilter fixtext
</Location>
```

## Tracing another filter

```
# Trace the data read and written by mod_deflate
# for a particular client (IP 192.168.1.31)
# experiencing compression problems.
# This filter will trace what goes into mod_deflate.
ExtFilterDefine tracebefore \
```

```
  cmd="/bin/tracefilter.pl /tmp/tracebefore" \
  EnableEnv=trace_this_client

# This filter will trace what goes after mod_deflate.
# Note that without the ftype parameter, the default
# filter type of AP_FTYPE_RESOURCE would cause the
# filter to be placed *before* mod_deflate in the filter
# chain. Giving it a numeric value slightly higher than
# AP_FTYPE_CONTENT_SET will ensure that it is placed
# after mod_deflate.
ExtFilterDefine traceafter \
  cmd="/bin/tracefilter.pl /tmp/traceafter" \
  EnableEnv=trace_this_client ftype=21

<Directory /usr/local/docs>
  SetEnvIf Remote_Addr 192.168.1.31 trace_this_client
  SetOutputFilter tracebefore;deflate;traceafter
</Directory>
```

**Here is the filter which traces the data:**

```
#!/usr/local/bin/perl -w
use strict;

open(SAVE, ">$ARGV[0]")
  or die "can't open $ARGV[0]: $?";

while (<STDIN>) {
  print SAVE $_;
  print $_;
}

close(SAVE);
```

## ExtFilterDefine Directive

| | |
|---|---|
| **Description:** | Define an external filter |
| **Syntax:** | `ExtFilterDefine` *`filtername parameters`* |
| **Context:** | server config |
| **Status:** | Extension |
| **Module:** | mod_ext_filter |

The `ExtFilterDefine` directive defines the characteristics of an external filter, including the program to run and its arguments.

*filtername* specifies the name of the filter being defined. This name can then be used in SetOutputFilter directives. It must be unique among all registered filters. *At the present time, no error is reported by the register-filter API, so a problem with duplicate names isn't reported to the user.*

Subsequent parameters can appear in any order and define the external command to run and certain other characteristics. The only required parameter is `cmd=`. These parameters are:

**`cmd=`*`cmdline`***

The `cmd=` keyword allows you to specify the external command to run. If there are arguments after the program name, the command line should be surrounded in quotation marks (*e.g.*, `cmd="`*`/bin/mypgm arg1 arg2`*`"`. Normal shell quoting is not necessary since the program is run directly, bypassing the shell. Program arguments are blank-delimited. A backslash can be

used to escape blanks which should be part of a program argument. Any backslashes which are part of the argument must be escaped with backslash themselves. In addition to the standard CGI environment variables, DOCUMENT_URI, DOCUMENT_PATH_INFO, and QUERY_STRING_UNESCAPED will also be set for the program.

**`mode=`*`mode`***
`mode` should be `output` for now (the default). In the future, `mode=input` will be used to specify a filter for request bodies.

**`intype=`*`imt`***
This parameter specifies the internet media type (*i.e.*, MIME type) of documents which should be filtered. By default, all documents are filtered. If `intype=` is specified, the filter will be disabled for documents of other types.

**`outtype=`*`imt`***
This parameter specifies the internet media type (*i.e.*, MIME type) of filtered documents. It is useful when the filter changes the internet media type as part of the filtering operation. By default, the internet media type is unchanged.

**`PreservesContentLength`**
The `PreservesContentLength` keyword specifies that the filter preserves the content length. This is not the default, as most filters change the content length. In the event that the filter doesn't modify the length, this keyword should be specified.

**`ftype=`*`filtertype`***
This parameter specifies the numeric value for filter type that the filter should be registered as. The default value, AP_FTYPE_RESOURCE, is sufficient in most cases. If the filter needs to operate at a different point in the filter chain than resource filters, then this parameter will be necessary. See the AP_FTYPE_foo definitions in util_filter.h for appropriate values.

**`disableenv=`*`env`***
This parameter specifies the name of an environment variable which, if set, will disable the filter.

**`enableenv=`*`env`***
This parameter specifies the name of an environment variable which must be set, or the filter will be disabled.

## ExtFilterOptions Directive

| | |
|---|---|
| **Description:** | Configure mod_ext_filter options |
| **Syntax:** | `ExtFilterOptions` *`option`* `[`*`option`*`] ...` |
| **Default:** | `ExtFilterOptions DebugLevel=0 NoLogStderr` |
| **Context:** | directory |
| **Status:** | Extension |
| **Module:** | mod_ext_filter |

The `ExtFilterOptions` directive specifies special processing options for `mod_ext_filter`. *Option* can be one of

**`DebugLevel=`*`n`***
The `DebugLevel` keyword allows you to specify the level of debug messages generated by `mod_ext_filter`. By default, no debug messages are generated. This is equivalent to `DebugLevel=0`. With higher numbers, more debug messages are generated, and server performance will be degraded. The actual meanings of the numeric values are described with the definitions of the DBGLVL_ constants near the beginning of `mod_ext_filter.c`.

Note: The core directive `LogLevel` should be used to cause debug messages to be stored in the

Apache error log.

**`LogStderr | NoLogStderr`**
The `LogStderr` keyword specifies that messages written to standard error by the external filter program will be saved in the Apache error log. `NoLogStderr` disables this feature.

**Example**
```
ExtFilterOptions LogStderr DebugLevel=0
```

Messages written to the filter's standard error will be stored in the Apache error log. No debug messages will be generated by `mod_ext_filter`.

## URI References

[1] http://httpd.apache.org/docs-2.1/filter.html