Apache Module mod_mime

Description:	Associates the requested filename's extensions with the file's behavior (handlers and filters) and content (mime-type, language, character set and encoding)
Status:	Base
Module Identifier:	mime_module
Source File:	mod_mime.c

Summary

This module is used to associate various bits of "meta information" with files by their filename extensions. This information relates the filename of the document to it's mime-type, language, character set and encoding. This information is sent to the browser, and participates in content negotiation, so the user's preferences are respected when choosing one of several possible files to serve. See `mod_negotiation` for more information about content negotiation[1].

The directives `AddCharset`, `AddEncoding`, `AddLanguage` and `AddType` are all used to map file extensions onto the meta-information for that file. Respectively they set the character set, content-encoding, content-language, and MIME-type (content-type) of documents. The directive `TypesConfig` is used to specify a file which also maps extensions onto MIME types.

In addition, `mod_mime` may define the handler[2] and filters[3] that originate and process content. The directives `AddHandler`, `AddOutputFilter`, and `AddInputFilter` control the modules or scripts that serve the document. The `MultiviewsMatch` directive allows `mod_negotiation` to consider these file extensions to be included when testing Multiviews matches.

While `mod_mime` associates meta-information with filename extensions, the `core` server provides directives that are used to associate all the files in a given container (*e.g.*, `<Location>`, `<Directory>`, or `<Files>`) with particular meta-information. These directives include `ForceType`, `SetHandler`, `SetInputFilter`, and `SetOutputFilter`. The core directives override any filename extension mappings defined in `mod_mime`.

Note that changing the meta-information for a file does not change the value of the `Last-Modified` header. Thus, previously cached copies may still be used by a client or proxy, with the previous headers. If you change the meta-information (language, content type, character set or encoding) you may need to 'touch' affected files (updating their last modified date) to ensure that all visitors are receive the corrected content headers.

Topics

Directives

See also

- MimeMagicFile
- AddDefaultCharset
- ForceType
- DefaultType
- SetHandler
- SetInputFilter
- SetOutputFilter

Files with Multiple Extensions

Files can have more than one extension, and the order of the extensions is *normally* irrelevant. For example, if the file `welcome.html.fr` maps onto content type `text/html` and language French then the file `welcome.fr.html` will map onto exactly the same information. If more than one extension is given which maps onto the same type of meta-information, then the one to the right will be used. For example, if `.gif` maps to the MIME-type `image/gif` and `.html` maps to the MIME-type `text/html`, then the file `welcome.gif.html` will be associated with the MIME-type `text/html`.

Care should be taken when a file with multiple extensions gets associated with both a MIME-type and a handler. This will usually result in the request being by the module associated with the handler. For example, if the `.imap` extension is mapped to the handler `imap-file` (from mod_imap) and the `.html` extension is mapped to the MIME-type `text/html`, then the file `world.imap.html` will be associated with both the `imap-file` handler and `text/html` MIME-type. When it is processed, the `imap-file` handler will be used, and so it will be treated as a mod_imap imagemap file.

Content encoding

A file of a particular MIME type can additionally be encoded a particular way to simplify transmission over the Internet. While this usually will refer to compression, such as `gzip`, it can also refer to encryption, such a `pgp` or to an encoding such as UUencoding, which is designed for transmitting a binary file in an ASCII (text) format.

The HTTP/1.1 RFC[4], section 14.11 puts it this way:

> *The Content-Encoding entity-header field is used as a modifier to the media-type. When present, its value indicates what additional content codings have been applied to the entity-body, and thus what decoding mechanisms must be applied in order to obtain the media-type referenced by the Content-Type header field. Content-Encoding is primarily used to allow a document to be compressed without losing the identity of its underlying media type.*

By using more than one file extension (see section above about multiple file extensions), you can indicate that a file is of a particular *type*, and also has a particular *encoding*.

For example, you may have a file which is a Microsoft Word document, which is pkzipped to reduce its size. If the `.doc` extension is associated with the Microsoft Word file type, and the `.zip` extension is associated with the pkzip file encoding, then the file `Resume.doc.zip`would be known to be a pkzip'ed Word document.

Apache sends a `Content-encoding` header with the resource, in order to tell the client browser about the encoding method.

```
Content-encoding: pkzip
```

Character sets and languages

In addition to file type and the file encoding, another important piece of information is what language a particular document is in, and in what character set the file should be displayed. For example, the document might be written in the Vietnamese alphabet, or in Cyrillic, and should be displayed as such. This information, also, is transmitted in HTTP headers.

The character set, language, encoding and mime type are all used in the process of content negotiation (See `mod_negotiation`) to determine which document to give to the client, when there are alternative documents in more than one character set, language, encoding or mime type. All filename extensions associations created with `AddCharset`, `AddEncoding`, `AddLanguage` and `AddType` directives (and extensions listed in the `MimeMagicFile`) participate in this select process. Filename extensions that are only associated using the `AddHandler`, `AddInputFilter` or `AddOutputFilter` directives may be included or excluded from matching by using the `MultiviewsMatch` directive.

Charset

To convey this further information, Apache optionally sends a `Content-Language` header, to specify the language that the document is in, and can append additional information onto the `Content-Type` header to indicate the particular character set that should be used to correctly render the information.

```
Content-Language: en, fr
Content-Type: text/plain; charset=ISO-8859-1
```

The language specification is the two-letter abbreviation for the language. The `charset` is the name of the particular character set which should be used.

AddCharset Directive

Description:	Maps the given filename extensions to the specified content charset
Syntax:	`AddCharset` *`charset extension`* `[`*`extension`*`] ...`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

The `AddCharset` directive maps the given filename extensions to the specified content charset. *charset* is the MIME charset parameter of filenames containing *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*.

Example

```
AddLanguage ja .ja
AddCharset EUC-JP .euc
AddCharset ISO-2022-JP .jis
AddCharset SHIFT_JIS .sjis
```

Then the document `xxxx.ja.jis` will be treated as being a Japanese document whose charset is `ISO-2022-JP` (as will the document `xxxx.jis.ja`). The `AddCharset` directive is useful for both to

inform the client about the character encoding of the document so that the document can be interpreted and displayed appropriately, and for content negotiation [1], where the server returns one from several documents based on the client's charset preference.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

See also

- mod_negotiation
- AddDefaultCharset

AddEncoding Directive

Description:	Maps the given filename extensions to the specified encoding type
Syntax:	`AddEncoding` *`MIME-enc extension`* `[`*`extension`*`] ...`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

The AddEncoding directive maps the given filename extensions to the specified encoding type. *MIME-enc* is the MIME encoding to use for documents containing the *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*.

Example

```
AddEncoding x-gzip .gz
AddEncoding x-compress .Z
```

This will cause filenames containing the `.gz` extension to be marked as encoded using the `x-gzip` encoding, and filenames containing the `.Z` extension to be marked as encoded with `x-compress`.

Old clients expect `x-gzip` and `x-compress`, however the standard dictates that they're equivalent to `gzip` and `compress` respectively. Apache does content encoding comparisons by ignoring any leading `x-`. When responding with an encoding Apache will use whatever form (*i.e.*, `x-foo` or `foo`) the client requested. If the client didn't specifically request a particular form Apache will use the form given by the `AddEncoding` directive. To make this long story short, you should always use `x-gzip` and `x-compress` for these two specific encodings. More recent encodings, such as `deflate` should be specified without the `x-`.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

AddHandler Directive

Description:	Maps the filename extensions to the specified handler
Syntax:	`AddHandler` *`handler-name extension`* `[`*`extension`*`] ...`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

Files having the name *extension* will be served by the specified *handler-name*[2]. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*. For example, to activate CGI scripts with the file extension `.cgi`, you might use:

```
AddHandler cgi-script .cgi
```

Once that has been put into your httpd.conf file, any file containing the `.cgi` extension will be treated as a CGI program.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

See also

- `SetHandler`

AddInputFilter Directive

Description:	Maps filename extensions to the filters that will process client requests
Syntax:	`AddInputFilter` *`filter`*`[;`*`filter`*`...] `*`extension`*` [`*`extension`*`] ...`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	AddInputFilter is only available in Apache 2.0.26 and later.

`AddInputFilter` maps the filename extension *extension* to the filters[3] which will process client requests and POST input when they are received by the server. This is in addition to any filters defined elsewhere, including the `SetInputFilter` directive. This mapping is merged over any already in force, overriding any mappings that already exist for the same *extension*.

If more than one filter is specified, they must be separated by semicolons in the order in which they should process the content. Both the filter and *extension* arguments are case-insensitive, and the extension may be specified with or without a leading dot.

AddLanguage Directive

Description:	Maps the given filename extension to the specified content language
Syntax:	`AddLanguage` *`MIME-lang extension`* `[`*`extension`*`] ...`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

The `AddLanguage` directive maps the given filename extension to the specified content language. *MIME-lang* is the MIME language of filenames containing *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*.

Example

```
AddEncoding x-compress .Z
```

```
AddLanguage en .en
AddLanguage fr .fr
```

Then the document `xxxx.en.Z` will be treated as being a compressed English document (as will the document `xxxx.Z.en`). Although the content language is reported to the client, the browser is unlikely to use this information. The `AddLanguage` directive is more useful for content negotiation[1], where the server returns one from several documents based on the client's language preference.

If multiple language assignments are made for the same extension, the last one encountered is the one that is used. That is, for the case of:

```
AddLanguage en .en
AddLanguage en-uk .en
AddLanguage en-us .en
```

documents with the extension `.en` would be treated as being `en-us`.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

See also

- `mod_negotiation`

AddOutputFilter Directive

Description:	Maps filename extensions to the filters that will process responses from the server
Syntax:	`AddOutputFilter` *`filter`*`[;`*`filter`*`...] `*`extension`*` [`*`extension`*`] ...`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	AddOutputFilter is only available in Apache 2.0.26 and later.

The `AddOutputFilter` directive maps the filename extension *extension* to the filters[3] which will process responses from the server before they are sent to the client. This is in addition to any filters defined elsewhere, including `SetOutputFilter` and `AddOutputFilterByType` directive. This mapping is merged over any already in force, overriding any mappings that already exist for the same *extension*.

For example, the following configuration will process all `.shtml` files for server-side includes and will then compress the output using `mod_deflate`.

```
AddOutputFilter INCLUDES;DEFLATE shtml
```

If more than one filter is specified, they must be separated by semicolons in the order in which they should process the content. Both the *filter* and *extension* arguments are case-insensitive, and the extension may be specified with or without a leading dot.

See also

- `RemoveOutputFilter`

AddType Directive

Description:	Maps the given filename extensions onto the specified content type
Syntax:	`AddType` *`MIME-type extension`* `[`*`extension`*`] ...`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

The `AddType` directive maps the given filename extensions onto the specified content type. *MIME-type* is the MIME type to use for filenames containing *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*. This directive can be used to add mappings not listed in the MIME types file (see the `TypesConfig` directive).

Example

```
AddType image/gif .gif
```

It is recommended that new MIME types be added using the `AddType` directive rather than changing the `TypesConfig` file.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

See also

- `DefaultType`
- `ForceType`

DefaultLanguage Directive

Description:	Sets all files in the given scope to the specified language
Syntax:	`DefaultLanguage` *`MIME-lang`*
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

The `DefaultLanguage` directive tells Apache that all files in the directive's scope (*e.g.*, all files covered by the current `<Directory>` container) that don't have an explicit language extension (such as `.fr` or `.de` as configured by `AddLanguage`) should be considered to be in the specified *MIME-lang* language. This allows entire directories to be marked as containing Dutch content, for instance, without having to rename each file. Note that unlike using extensions to specify languages, `DefaultLanguage` can only specify a single language.

If no `DefaultLanguage` directive is in force, and a file does not have any language extensions as configured by `AddLanguage`, then that file will be considered to have no language attribute.

Example

```
DefaultLanguage en
```

See also

- mod_negotiation

ModMimeUsePathInfo Directive

Description:	Tells mod_mime to treat path_info components as part of the filename
Syntax:	`ModMimeUsePathInfo On|Off`
Default:	`ModMimeUsePathInfo Off`
Context:	directory
Status:	Base
Module:	mod_mime
Compatibility:	Available in Apache 2.0.41 and later

The ModMimeUsePathInfo directive is used to combine the filename with the `path_info` URL component to apply mod_mime's directives to the request. The default value is `Off` - therefore, the `path_info` component is ignored.

This directive is recommended when you have a virtual filesystem.

Example

```
ModMimeUsePathInfo On
```

If you have a request for `/bar/foo.shtml` where `/bar` is a Location and ModMimeUsePathInfo is `On`, mod_mime will treat the incoming request as `/bar/foo.shtml` and directives like `AddOutputFilter INCLUDES .shtml` will add the `INCLUDES` filter to the request. If ModMimeUsePathInfo is not set, the `INCLUDES` filter will not be added.

See also

- AcceptPathInfo

MultiviewsMatch Directive

Description:	The types of files that will be included when searching for a matching file with MultiViews
Syntax:	`MultiviewsMatch Any|NegotiatedOnly|Filters|Handlers [Handlers|Filters]`
Default:	`MultiviewsMatch NegotiatedOnly`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	Available in Apache 2.0.26 and later.

MultiviewsMatch permits three different behaviors for mod_negotiation [5]'s Multiviews feature. Multiviews allows a request for a file, *e.g.* `index.html`, to match any negotiated extensions

following the base request, *e.g.* `index.html.en`, `index.html.fr`, or `index.html.gz`.

The `NegotiatedOnly` option provides that every extension following the base name must correlate to a recognized `mod_mime` extension for content negotation, *e.g.* Charset, Content-Type, Language, or Encoding. This is the strictest implementation with the fewest unexpected side effects, and is the default behavior.

To include extensions associated with Handlers and/or Filters, set the `MultiviewsMatch` directive to either `Handlers`, `Filters`, or both option keywords. If all other factors are equal, the smallest file will be served, *e.g.* in deciding between `index.html.cgi` of 500 bytes and `index.html.pl` of 1000 bytes, the `.cgi` file would win in this example. Users of `.asis` files might prefer to use the Handler option, if `.asis` files are associated with the `asis-handler`.

You may finally allow `Any` extensions to match, even if `mod_mime` doesn't recognize the extension. This was the behavior in Apache 1.3, and can cause unpredicatable results, such as serving .old or .bak files the webmaster never expected to be served.

For example, the following configuration will allow handlers and filters to participate in Multviews, but will exclude unknown files:

```
MultiviewsMatch Handlers Filters
```

See also

- `Options`
- `mod_negotiation`

RemoveCharset Directive

Description:	Removes any character set associations for a set of file extensions
Syntax:	`RemoveCharset` *`extension`* `[`*`extension`*`] ...`
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	RemoveCharset is only available in Apache 2.0.24 and later.

The `RemoveCharset` directive removes any character set associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

Example

```
RemoveCharset .html .shtml
```

RemoveEncoding Directive

Description:	Removes any content encoding associations for a set of file extensions
Syntax:	`RemoveEncoding` *`extension`* `[`*`extension`*`] ...`

Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

The `RemoveEncoding` directive removes any encoding associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server config files. An example of its use might be:

/foo/.htaccess:

```
AddEncoding x-gzip .gz
AddType text/plain .asc
<Files *.gz.asc>
  RemoveEncoding .gz
</Files>
```

This will cause `foo.gz` to be marked as being encoded with the gzip method, but `foo.gz.asc` as an unencoded plaintext file.

Note

`RemoveEncoding` directives are processed *after* any `AddEncoding` directives, so it is possible they may undo the effects of the latter if both occur within the same directory configuration.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

RemoveHandler Directive

Description:	Removes any handler associations for a set of file extensions
Syntax:	`RemoveHandler` *`extension`* `[`*`extension`*`] ...`
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

The `RemoveHandler` directive removes any handler associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server config files. An example of its use might be:

/foo/.htaccess:

```
AddHandler server-parsed .html
```

/foo/bar/.htaccess:

```
RemoveHandler .html
```

This has the effect of returning `.html` files in the `/foo/bar` directory to being treated as normal files, rather than as candidates for parsing (see the `mod_include` module).

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

RemoveInputFilter Directive

Description:	Removes any input filter associations for a set of file extensions
Syntax:	`RemoveInputFilter` *`extension`* `[`*`extension`*`] ...`
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	RemoveInputFilter is only available in Apache 2.0.26 and later.

The `RemoveInputFilter` directive removes any input filter associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

RemoveLanguage Directive

Description:	Removes any language associations for a set of file extensions
Syntax:	`RemoveLanguage` *`extension`* `[`*`extension`*`] ...`
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	RemoveLanguage is only available in Apache 2.0.24 and later.

The `RemoveLanguage` directive removes any language associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

RemoveOutputFilter Directive

Description:	Removes any output filter associations for a set of file extensions
Syntax:	`RemoveOutputFilter` *`extension`* `[`*`extension`*`] ...`
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	RemoveOutputFilter is only available in Apache 2.0.26 and later.

The `RemoveOutputFilter` directive removes any output filter associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

Example

```
RemoveOutputFilter shtml
```

See also

- AddOutputFilter

RemoveType Directive

Description:	Removes any content type associations for a set of file extensions
Syntax:	`RemoveType` *`extension`* `[`*`extension`*`] ...`
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

The RemoveType directive removes any MIME type associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server config files. An example of its use might be:

/foo/.htaccess:

```
RemoveType .cgi
```

This will remove any special handling of `.cgi` files in the `/foo/` directory and any beneath it, causing the files to be treated as being of the DefaultType.

Note

RemoveType directives are processed *after* any AddType directives, so it is possible they may undo the effects of the latter if both occur within the same directory configuration.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

TypesConfig Directive

Description:	The location of the mime.types file
Syntax:	`TypesConfig` *`file-path`*
Default:	`TypesConfig conf/mime.types`
Context:	server config
Status:	Base
Module:	mod_mime

The TypesConfig directive sets the location of the MIME types configuration file. *File-path* is relative to the ServerRoot. This file sets the default list of mappings from filename extensions to content types. Most administrators use the provided `mime.types` file, which associates common filename extensions with IANA registered content types. The current list is maintained at

http://www.isi.edu/in-notes/iana/assignments/media-types/media-types. This simplifies the `httpd.conf` file by providing the majority of media-type definitions, and may be overridden by `AddType` directives as needed. You should not edit the `mime.types` file, because it may be replaced when you upgrade your server.

The file contains lines in the format of the arguments to an `AddType` directive:

```
MIME-type [extension] ...
```

The case of the extension does not matter. Blank lines, and lines beginning with a hash character (#) are ignored.

Please do **not** send requests to the Apache HTTP Server Project to add any new entries in the distributed `mime.types` file unless (1) they are already registered with IANA, and (2) they use widely accepted, non-conflicting filename extensions across platforms. `category/x-subtype` requests will be automatically rejected, as will any new two-letter extensions as they will likely conflict later with the already crowded language and character set namespace.

See also

- `mod_mime_magic`

URI References

[1] http://httpd.apache.org/docs-2.1/content-negotiation.html
[2] http://httpd.apache.org/docs-2.1/handler.html
[3] http://httpd.apache.org/docs-2.1/filter.html
[4] http://www.ietf.org/rfc/rfc2616.txt
[5] http://httpd.apache.org/docs-2.1/mod/mod_negotiation.html