Technologies

References & Guides

Feedback

Sign in

Search

MIME types

The **Multipurpose Internet Mail Extensions (MIME) type** is a standardized way to indicate the nature and format of a document. It is defined and standardized in IETF RFC 6838. The Internet Assigned Numbers Authority (IANA) is the official body responsible for keeping track of all official MIME types, and you can find the most up-to-date and complete list at the Media Types page.

Browsers often use the MIME type (and not the file extension) to determine how it will process a document; it is therefore important that servers are set up correctly to attach the correct MIME type to the header of the response object.

Syntax

General structure

```
type/subtype
```

The structure of a MIME type is very simple; it consists of a type and a subtype, two strings, separated by a `'/'`. No space is allowed. The *type* represents the category and can be a *discrete* or a *multipart* type. The *subtype* is specific to each type.

A MIME type is case-insensitive but traditionally is written all in lower case.

Discrete types

```
text/plain
text/html
image/jpeg
image/png
audio/mpeg
audio/ogg
audio/*
video/mp4
application/*
application/json
application/javascript
application/ecmascript
application/octet-stream
…
```

Discrete types indicate the category of the document, it can be one of the following:

Type	Description	Example of typical subtypes
`text`	Represents any document that contains text and is theoretically human readable	`text/plain`, `text/html`, `text/css`, `text/javascript`
`image`	Represents any kind of images. Videos are not included, though animated images (like animated gif) are described with an image type.	`image/gif`, `image/png`, `image/jpeg`, `image/bmp`, `image/webp`
`audio`	Represents any kind of audio files	`audio/midi`, `audio/mpeg, audio/webm, audio/ogg, audio/wav`
`video`	Represents any kind of video files	`video/webm`, `video/ogg`

Type	Description	Example of typical subtypes
`application`	Represents any kind of binary data.	`application/octet-stream`, `application/pkcs12`, `application/vnd.mspowerpoint`, `application/xhtml+xml`, `application/xml`, `application/pdf`

For text documents without specific subtype, `text/plain` should be used. Similarly, for binary documents without specific or known subtype, `application/octet-stream` should be used.

Multipart types

```
multipart/form-data
multipart/byteranges
```

Multipart types indicate a category of document that are broken in distinct parts, often with different MIME types. It is a way to represent a *composite* document. With the exception of `multipart/form-data`, that are used in relation of HTML Forms and `POST` method, and `multipart/byteranges` that are used in conjunction with `206 Partial Content` status message to send only a subset of a whole document, HTTP doesn't handle multipart documents in a specific way: the message is simply transmitted to the browser (which will likely propose a Save As window, not knowing how to display the document inline.)

Important MIME types for Web developers

`application/octet-stream`

This is the default value for a binary file. As it really means *unknown binary* file, browsers usually don't automatically execute it, or even ask if it should be executed. They treat it as if the `Content-Disposition` header was set with the value `attachment` and propose a 'Save As' file.

`text/plain`

This is the default value for textual files. Even if it really means *unknown textual* file, browsers assume they can display it.

> Note that `text/plain` does not mean *any kind of textual data*. If they expect a specific kind of textual data, they will likely not consider it a match. Specifically if they download a `text/plain` file from a `<link>` element declaring a CSS files, they will not recognize it as a valid CSS files if presented with `text/plain`. The CSS mime type `text/css` must be used.

`text/css`

Any CSS files that have to be interpreted as such in a Web page **must** be of the `text/css` files. Often servers do not recognize files with the `.css` suffix as CSS files, instead they send them with `text/plain` or `application/octet-stream` MIME type: in these cases, they won't be recognized as CSS files by most browsers and will be silently ignored. Special attention has to be paid to serve CSS files with the correct type.

`text/html`

All HTML content should be served with this type. Alternative MIME types for XHTML (like `application/xml+html)` are mostly useless nowadays (HTML5 unified these formats).

Images types

Only a handful of image types are widely recognized and are considered Web safe, ready for use in a Web page:

MIME type	Image type
`image/gif`	GIF images (lossless compression, superseded by PNG)
`image/jpeg`	JPEG images
`image/png`	PNG images
`image/svg+xml`	SVG images (vector images)

There is a discussion to add WebP (`image/webp`) to this list, but as each new image type will increase the size of a codebase, this may introduce new security problems, so browser vendors are cautious in accepting it.

Other kinds of images can be found in Web documents. For example, many browsers support icon image types for favicons or similar. In particular, ICO images are supported in this context with the `image/x-icon` MIME type.

Audio and video types

Like images, HTML doesn't define a set of supported types to use with the `<audio>` and `<video>` elements, so only a relatively small group of them can be used on the Web. The Media formats supported by the HTML audio and video elements explains both the codecs and container formats which can be used.

The MIME type of such files mostly represent the container formats and the most common ones in a Web context are:

MIME type	Audio or video type

`audio/wave`
`audio/wav`
`audio/x-wav`
`audio/x-pn-wav`
	An audio file in the WAVE container format. The PCM audio codec (WAVE codec "1") is often supported, but other codecs have more limited support (if any).

`audio/webm`	An audio file in the WebM container format. Vorbis and Opus are the most common audio codecs.
`video/webm`	A video file, possibly with audio, in the WebM container format. VP8 and VP9 are the most common video codecs used within it; Vorbis and Opus the most common audio codecs.
`audio/ogg`	An audio file in the OGG container format. Vorbis is the most common audio codec used in such a container.
`video/ogg`	A video file, possibly with audio, in the OGG container format. Theora is the usual video codec used within it; Vorbis is the usual audio codec.
`application/ogg`	An audio or video file using the OGG container format. Theora is the usual video codec used within it; Vorbis is the usual audio codec.

multipart/form-data

The `multipart/form-data` type can be used when sending the content of a completed HTML Form from the browser to the server. As a multipart document format, it consists of different parts, delimited by a boundary (a string starting with a double dash `'--'`). Each part is an entity by itself, with its own HTTP headers, `Content-Disposition`, and `Content-Type` for file uploading fields, and the most common (`Content-Length` is ignored as the boundary line is used as the delimiter).

```
Content-Type: multipart/form-data; boundary=aBoundaryString
(other headers associated with the multipart document as a whole)

--aBoundaryString
Content-Disposition: form-data; name="myFile"; filename="img.jpg"
Content-Type: image/jpeg

(data)
--aBoundaryString
Content-Disposition: form-data; name="myField"

(data)
--aBoundaryString
(more subparts)
--aBoundaryString--
```

The following form:

```
<form action="http://localhost:8000/" method="post" enctype="multipart/form-c
  <input type="text" name="myTextField">
  <input type="checkbox" name="myCheckBox">Check</input>
  <input type="file" name="myFile">
  <button>Send the file</button>
</form>
```

will send this message:

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=---------------------------872165
Content-Length: 465

-----------------------------8721656041911415653955004498
Content-Disposition: form-data; name="myTextField"

Test
-----------------------------8721656041911415653955004498
Content-Disposition: form-data; name="myCheckBox"

on
-----------------------------8721656041911415653955004498
Content-Disposition: form-data; name="myFile"; filename="test.txt"
Content-Type: text/plain

Simple file.
-----------------------------8721656041911415653955004498--
```

multipart/byteranges

The `multipart/byteranges` MIME type is used in the context of sending partial responses back to the browser. When the `206 Partial Content` status code is sent, this MIME type is used to indicate that the document is composed of several parts, one for each of the requested range. Like other multipart types, the `Content-Type` uses the `boundary` directive to define the boundary string. Each of the different parts have a `Content-Type` header with the actual type of the document and a `Content-Range` with the range they represent.

```
HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
```

```
Content-Type: multipart/byteranges; boundary=3d6b6a416f9b5
Content-Length: 385

--3d6b6a416f9b5
Content-Type: text/html
Content-Range: bytes 100-200/1270

eta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="vieport" content
--3d6b6a416f9b5
Content-Type: text/html
Content-Range: bytes 300-400/1270

-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: "Open Sans", "Helvetica
--3d6b6a416f9b5--
```

Importance of setting the correct MIME type

Most web servers send unknown-type resources using the default `application/octet-stream` MIME type. For security reasons, most browsers do not allow setting a custom default action for such resources, forcing the user to store it to disk to use it. Some commonly seen incorrect server configurations happen with the following file types:

- RAR-encoded files. In this case, the ideal would be to set the true type of the encoded files; this is often not possible (as it may not be known to the server and these files may contain several resources of different types). In this case, configuring the server to send the `application/x-rar-compressed` MIME type, users will not have defined a useful default action for them.

- Audio and video files. Only resources with the correct MIME Type will be recognized and played in `<video>` or `<audio>` elements. Be sure to use the correct type for audio and video.

- Proprietary file types. Pay particular attention when serving a proprietary file type. Avoid using `application/octet-stream` as special handling will not be possible: most browsers do not allow defining a default behavior (like "Opening in Word") for this generic MIME type.

MIME sniffing

In the absence of a MIME type, or in some other cases where a client believes they are incorrectly set, browsers may conduct MIME sniffing, which is guessing the correct MIME type by looking at the resource. Each browser performs this differently and under different circumstances. There are some security concerns with this practice, as some MIME types represent executable content and others not. Servers can block MIME sniffing by sending the `X-Content-Type-Options` along the `Content-Type`.

Other methods of conveying document type

MIME types are not the only way to convey the document type information:

- Name suffixes are sometimes used, especially on Microsoft Windows systems. Not all operating systems consider these suffixes meaningful (especially Linux and Mac OS), and like an external MIME type, there is no guarantee they are correct.
- Magic numbers. The syntax of the different kind of files allows file-type inference by looking at the structure. E.g. each GIF files starts with the 47 49 46 38 hexadecimal value [GIF89] or PNG files with 89 50 4E 47 [.PNG]. Not all types of files have magic numbers, so this is not a 100% reliable system either.

See also

- Properly configuring server MIME types

- Media formats supported by the HTML audio and video elements
- https://www.iana.org/assignments/media-types/application/json