19.1.10. `email.mime`: Creating email and MIME objects from scratch

Source code: Lib/email/mime/

This module is part of the legacy (`Compat32`) email API. Its functionality is partially replaced by the `contentmanager` in the new API, but in certain applications these classes may still be useful, even in non-legacy code.

Ordinarily, you get a message object structure by passing a file or some text to a parser, which parses the text and returns the root message object. However you can also build a complete message structure from scratch, or even individual `Message` objects by hand. In fact, you can also take an existing structure and add new `Message` objects, move them around, etc. This makes a very convenient interface for slicing-and-dicing MIME messages.

You can create a new object structure by creating `Message` instances, adding attachments and all the appropriate headers manually. For MIME messages though, the `email` package provides some convenient subclasses to make things easier.

Here are the classes:

class `email.mime.base.`**`MIMEBase`**(*_maintype*, *_subtype*, *, *policy=compat32*, ***_params*)

Module: `email.mime.base`

This is the base class for all the MIME-specific subclasses of `Message`. Ordinarily you won't create instances specifically of `MIMEBase`, although you could. `MIMEBase` is provided primarily as a convenient base class for more specific MIME-aware subclasses.

_maintype is the *Content-Type* major type (e.g. *text* or *image*), and *_subtype* is the *Content-Type* minor type (e.g. *plain* or *gif*). *_params* is a parameter key/value dictionary and is passed directly to `Message.add_header`.

If *policy* is specified, (defaults to the `compat32` policy) it will be passed to `Message`.

The `MIMEBase` class always adds a *Content-Type* header (based on *_maintype*, *_subtype*, and *_params*), and a *MIME-Version* header (always set to `1.0`).

Changed in version 3.6: Added *policy* keyword-only parameter.

class email.mime.nonmultipart.**MIMENonMultipart**

Module: email.mime.nonmultipart

A subclass of MIMEBase, this is an intermediate base class for MIME messages that are not *multipart*. The primary purpose of this class is to prevent the use of the attach() method, which only makes sense for *multipart* messages. If attach() is called, a MultipartConversionError exception is raised.

class email.mime.multipart.**MIMEMultipart**(*_subtype='mixed'*, *boundary=None*, *_subparts=None*, ***, *policy=compat32*, ***_params*)

Module: email.mime.multipart

A subclass of MIMEBase, this is an intermediate base class for MIME messages that are *multipart*. Optional *_subtype* defaults to *mixed*, but can be used to specify the subtype of the message. A *Content-Type* header of *multipart/_subtype* will be added to the message object. A *MIME-Version* header will also be added.

Optional *boundary* is the multipart boundary string. When None (the default), the boundary is calculated when needed (for example, when the message is serialized).

_subparts is a sequence of initial subparts for the payload. It must be possible to convert this sequence to a list. You can always attach new subparts to the message by using the Message.attach method.

Optional *policy* argument defaults to compat32.

Additional parameters for the *Content-Type* header are taken from the keyword arguments, or passed into the *_params* argument, which is a keyword dictionary.

Changed in version 3.6: Added *policy* keyword-only parameter.

class email.mime.application.**MIMEApplication**(*_data*, *_subtype='octet-stream'*, *_encoder=email.encoders.encode_base64*, ***, *policy=compat32*, ***_params*)

Module: email.mime.application

A subclass of MIMENonMultipart, the MIMEApplication class is used to represent MIME message objects of major type *application*. *_data* is a string containing the raw byte data. Optional *_subtype* specifies the MIME subtype and defaults to *octet-stream*.

Optional *_encoder* is a callable (i.e. function) which will perform the actual encoding of the data for transport. This callable takes one argument, which is the

`MIMEApplication` instance. It should use `get_payload()` and `set_payload()` to change the payload to encoded form. It should also add any *Content-Transfer-Encoding* or other headers to the message object as necessary. The default encoding is base64. See the `email.encoders` module for a list of the built-in encoders.

Optional *policy* argument defaults to `compat32`.

_params are passed straight through to the base class constructor.

Changed in version 3.6: Added *policy* keyword-only parameter.

class `email.mime.audio.`**`MIMEAudio`**(*_audiodata*, *_subtype=None*, *_encoder=email.encoders.encode_base64*, *, *policy=compat32*, ***_params*)

Module: `email.mime.audio`

A subclass of `MIMENonMultipart`, the `MIMEAudio` class is used to create MIME message objects of major type *audio*. *_audiodata* is a string containing the raw audio data. If this data can be decoded by the standard Python module `sndhdr`, then the subtype will be automatically included in the *Content-Type* header. Otherwise you can explicitly specify the audio subtype via the *_subtype* argument. If the minor type could not be guessed and *_subtype* was not given, then `TypeError` is raised.

Optional *_encoder* is a callable (i.e. function) which will perform the actual encoding of the audio data for transport. This callable takes one argument, which is the `MIMEAudio` instance. It should use `get_payload()` and `set_payload()` to change the payload to encoded form. It should also add any *Content-Transfer-Encoding* or other headers to the message object as necessary. The default encoding is base64. See the `email.encoders` module for a list of the built-in encoders.

Optional *policy* argument defaults to `compat32`.

_params are passed straight through to the base class constructor.

Changed in version 3.6: Added *policy* keyword-only parameter.

class `email.mime.image.`**`MIMEImage`**(*_imagedata*, *_subtype=None*, *_encoder=email.encoders.encode_base64*, *, *policy=compat32*, ***_params*)

Module: `email.mime.image`

A subclass of `MIMENonMultipart`, the `MIMEImage` class is used to create MIME message objects of major type *image*. *_imagedata* is a string containing the raw image data. If this data can be decoded by the standard Python module `imghdr`, then the subtype will be automatically included in the *Content-Type*

header. Otherwise you can explicitly specify the image subtype via the *_subtype* argument. If the minor type could not be guessed and *_subtype* was not given, then `TypeError` is raised.

Optional *_encoder* is a callable (i.e. function) which will perform the actual encoding of the image data for transport. This callable takes one argument, which is the `MIMEImage` instance. It should use `get_payload()` and `set_payload()` to change the payload to encoded form. It should also add any *Content-Transfer-Encoding* or other headers to the message object as necessary. The default encoding is base64. See the `email.encoders` module for a list of the built-in encoders.

Optional *policy* argument defaults to `compat32`.

_params are passed straight through to the `MIMEBase` constructor.

Changed in version 3.6: Added *policy* keyword-only parameter.

class `email.mime.message.`**MIMEMessage**(*_msg*, *_subtype='rfc822'*, ***, *policy=compat32*)

Module: `email.mime.message`

A subclass of `MIMENonMultipart`, the `MIMEMessage` class is used to create MIME objects of main type *message*. *_msg* is used as the payload, and must be an instance of class `Message` (or a subclass thereof), otherwise a `TypeError` is raised.

Optional *_subtype* sets the subtype of the message; it defaults to *rfc822*.

Optional *policy* argument defaults to `compat32`.

Changed in version 3.6: Added *policy* keyword-only parameter.

class `email.mime.text.`**MIMEText**(*_text*, *_subtype='plain'*, *_charset=None*, ***, *policy=compat32*)

Module: `email.mime.text`

A subclass of `MIMENonMultipart`, the `MIMEText` class is used to create MIME objects of major type *text*. *_text* is the string for the payload. *_subtype* is the minor type and defaults to *plain*. *_charset* is the character set of the text and is passed as an argument to the `MIMENonMultipart` constructor; it defaults to `us-ascii` if the string contains only `ascii` code points, and `utf-8` otherwise. The *_charset* parameter accepts either a string or a `Charset` instance.

Unless the *_charset* argument is explicitly set to `None`, the MIMEText object created will have both a *Content-Type* header with a `charset` parameter, and a

Content-Transfer-Encoding header. This means that a subsequent `set_payload` call will not result in an encoded payload, even if a charset is passed in the `set_payload` command. You can "reset" this behavior by deleting the `Content-Transfer-Encoding` header, after which a `set_payload` call will automatically encode the new payload (and add a new *Content-Transfer-Encoding* header).

Optional *policy* argument defaults to `compat32`.

Changed in version 3.5: *_charset* also accepts `Charset` instances.

Changed in version 3.6: Added *policy* keyword-only parameter.