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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 4)

Unknown directive type "module".

```
.. module:: email.mime
   :synopsis: Build MIME messages.
```

Source code: :source:`Lib/email/mime/`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 7); *backlink*

Unknown interpreted text role "source".

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 11); *backlink*

Unknown interpreted text role "mod".

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 :class:`~email.message.Message` objects by hand. In fact, you can also take an existing structure and add new :class:`~email.message.Message` objects, move them around, etc. This makes a very convenient interface for slicing-and-dicing MIME messages.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 16); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 16); *backlink*

Unknown interpreted text role "class".

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 24); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 24); *backlink*

Unknown interpreted text role "mod".

Here are the classes:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 31)

Unknown directive type "currentmodule".

```
.. currentmodule:: email.mime.base
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 33)

Invalid class attribute value for "class" directive: "MIMEBase(_maintype, _subtype, *, policy=compat32, **_params)".

```
.. class:: MIMEBase(_maintype, _subtype, *, policy=compat32, **_params)

   Module: :mod:`email.mime.base`

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

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

   If *policy* is specified, (defaults to the
   :class:`compat32 <email.policy.Compat32>` policy) it will be passed to
   :class:`~email.message.Message`.

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

   .. versionchanged:: 3.6
      Added *policy* keyword-only parameter.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 61)

Unknown directive type "currentmodule".

```
.. currentmodule:: email.mime.nonmultipart
```

Module: :mod:`email.mime.nonmultipart`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 65); *backlink*

Unknown interpreted text role "mod".

A subclass of :class:`~email.mime.base.MIMEBase`, this is an intermediate base class for MIME messages that are not :mimetype:`multipart`. The primary purpose of this class is to prevent the use of the :meth:`~email.message.Message.attach` method, which only makes sense for :mimetype:`multipart` messages. If :meth:`~email.message.Message.attach` is called, a :exc:`~email.errors.MultipartConversionError` exception is raised.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 67); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 67); *backlink*

Unknown interpreted text role "mimetype".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 67); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 67); ***backlink***

Unknown interpreted text role "mimetype".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 67); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 67); ***backlink***

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 75)

Unknown directive type "currentmodule".

```
.. currentmodule:: email.mime.multipart
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 77)

Invalid class attribute value for "class" directive: "MIMEMultipart(_subtype='mixed', boundary=None, _subparts=None, \ *, policy=compat32, **_params)".

```
.. class:: MIMEMultipart(_subtype='mixed', boundary=None, _subparts=None, \
                         *, policy=compat32, **_params)

   Module: :mod:`email.mime.multipart`

   A subclass of :class:`~email.mime.base.MIMEBase`, this is an intermediate base
   class for MIME messages that are :mimetype:`multipart`.  Optional *_subtype*
   defaults to :mimetype:`mixed`, but can be used to specify the subtype of the
   message.  A :mailheader:`Content-Type` header of :mimetype:`multipart/_subtype`
   will be added to the message object.  A :mailheader:`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 :meth:`Message.attach
   <email.message.Message.attach>` method.

   Optional *policy* argument defaults to :class:`compat32 <email.policy.Compat32>`.

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

   .. versionchanged:: 3.6
      Added *policy* keyword-only parameter.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 107)

Unknown directive type "currentmodule".

```
.. currentmodule:: email.mime.application
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst, line 109)

Invalid class attribute value for "class" directive: "MIMEApplication(_data, _subtype='octet-stream', \ _encoder=email.encoders.encode_base64, \ *, policy=compat32, **_params)".

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

   Module: :mod:`email.mime.application`

   A subclass of :class:`~email.mime.nonmultipart.MIMENonMultipart`, the
   :class:`MIMEApplication` class is used to represent MIME message objects of
   major type :mimetype:`application`.  *_data* is a string containing the raw
   byte data.  Optional *_subtype* specifies the MIME subtype and defaults to
   :mimetype:`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 :class:`MIMEApplication` instance. It should use
   :meth:`~email.message.Message.get_payload` and
   :meth:`~email.message.Message.set_payload` to change the payload to encoded
   form.  It should also add
   any :mailheader:`Content-Transfer-Encoding` or other headers to the message
   object as necessary.  The default encoding is base64.  See the
   :mod:`email.encoders` module for a list of the built-in encoders.

   Optional *policy* argument defaults to :class:`compat32 <email.policy.Compat32>`.

   *_params* are passed straight through to the base class constructor.

   .. versionchanged:: 3.6
      Added *policy* keyword-only parameter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst, line 138)

Unknown directive type "currentmodule".

```
.. currentmodule:: email.mime.audio
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst, line 140)

Invalid class attribute value for "class" directive: "MIMEAudio(_audiodata, _subtype=None, \ _encoder=email.encoders.encode_base64, \ *, policy=compat32, **_params)".

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

   Module: :mod:`email.mime.audio`

   A subclass of :class:`~email.mime.nonmultipart.MIMENonMultipart`, the
   :class:`MIMEAudio` class is used to create MIME message objects of major type
   :mimetype:`audio`. *_audiodata* is a string containing the raw audio data.  If
   this data can be decoded by the standard Python module :mod:`sndhdr`, then the
   subtype will be automatically included in the :mailheader:`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 :exc:`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 :class:`MIMEAudio` instance. It should use
   :meth:`~email.message.Message.get_payload` and
   :meth:`~email.message.Message.set_payload` to change the payload to encoded
   form.  It should also add
   any :mailheader:`Content-Transfer-Encoding` or other headers to the message
   object as necessary.  The default encoding is base64.  See the
   :mod:`email.encoders` module for a list of the built-in encoders.
   Optional *policy* argument defaults to :class:`compat32 <email.policy.Compat32>`.

   *_params* are passed straight through to the base class constructor.

   .. versionchanged:: 3.6
      Added *policy* keyword-only parameter.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 172)

Unknown directive type "currentmodule".

```
.. currentmodule:: email.mime.image
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 174)

Invalid class attribute value for "class" directive: "MIMEImage(_imagedata, _subtype=None, \ _encoder=email.encoders.encode_base64, \ *, policy=compat32, **_params)".

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

   Module: :mod:`email.mime.image`

   A subclass of :class:`~email.mime.nonmultipart.MIMENonMultipart`, the
   :class:`MIMEImage` class is used to create MIME message objects of major type
   :mimetype:`image`. *_imagedata* is a string containing the raw image data.  If
   this data can be decoded by the standard Python module :mod:`imghdr`, then the
   subtype will be automatically included in the :mailheader:`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 :exc:`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 :class:`MIMEImage` instance. It should use
   :meth:`~email.message.Message.get_payload` and
   :meth:`~email.message.Message.set_payload` to change the payload to encoded
   form.  It should also add
   any :mailheader:`Content-Transfer-Encoding` or other headers to the message
   object as necessary.  The default encoding is base64.  See the
   :mod:`email.encoders` module for a list of the built-in encoders.

   Optional *policy* argument defaults to :class:`compat32 <email.policy.Compat32>`.

   *_params* are passed straight through to the :class:`~email.mime.base.MIMEBase`
   constructor.

   .. versionchanged:: 3.6
      Added *policy* keyword-only parameter.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 207)

Unknown directive type "currentmodule".

```
.. currentmodule:: email.mime.message
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 209)

Invalid class attribute value for "class" directive: "MIMEMessage(_msg, _subtype='rfc822', *, policy=compat32)".

```
.. class:: MIMEMessage(_msg, _subtype='rfc822', *, policy=compat32)

   Module: :mod:`email.mime.message`

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

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

   Optional *policy* argument defaults to :class:`compat32 <email.policy.Compat32>`.

   .. versionchanged:: 3.6
      Added *policy* keyword-only parameter.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 227)

Unknown directive type "currentmodule".

```
.. currentmodule:: email.mime.text
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)email.mime.rst`, line 229)

Invalid class attribute value for "class" directive: "MIMEText(_text, _subtype='plain', _charset=None, *, policy=compat32)".

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

   Module: :mod:`email.mime.text`

   A subclass of :class:`~email.mime.nonmultipart.MIMENonMultipart`, the
   :class:`MIMEText` class is used to create MIME objects of major type
   :mimetype:`text`. *_text* is the string for the payload.  *_subtype* is the
   minor type and defaults to :mimetype:`plain`.  *_charset* is the character
   set of the text and is passed as an argument to the
   :class:`~email.mime.nonmultipart.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
   :class:`~email.charset.Charset` instance.

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

   Optional *policy* argument defaults to :class:`compat32 <email.policy.Compat32>`.

   .. versionchanged:: 3.5
      *_charset* also accepts :class:`~email.charset.Charset` instances.

   .. versionchanged:: 3.6
      Added *policy* keyword-only parameter.
```