:mod:`email.generator`: Generating MIME documents

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]email.generator.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.generator.rst`, line 4)
>
> Unknown directive type "module".
>
> ```
>     .. module:: email.generator
>        :synopsis: Generate flat text email messages from a message structure.
> ```

Source code: :source:`Lib/email/generator.py`

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]email.generator.rst`, line 7); *backlink*
>
> Unknown interpreted text role "source".

One of the most common tasks is to generate the flat (serialized) version of the email message represented by a message object structure. You will need to do this if you want to send your message via :meth:`smtplib.SMTP.sendmail` or the :mod:`nntplib` module, or print the message on the console. Taking a message object structure and producing a serialized representation is the job of the generator classes.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]email.generator.rst`, line 11); *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.generator.rst`, line 11); *backlink*
>
> Unknown interpreted text role "mod".

As with the :mod:`email.parser` module, you aren't limited to the functionality of the bundled generator; you could write one from scratch yourself. However the bundled generator knows how to generate most email in a standards-compliant way, should handle MIME and non-MIME email messages just fine, and is designed so that the bytes-oriented parsing and generation operations are inverses, assuming the same non-transforming :mod:`~email.policy` is used for both. That is, parsing the serialized byte stream via the :class:`~email.parser.BytesParser` class and then regenerating the serialized byte stream using :class:`BytesGenerator` should produce output identical to the input [1]. (On the other hand, using the generator on an :class:`~email.message.EmailMessage` constructed by program may result in changes to the :class:`~email.message.EmailMessage` object as defaults are filled in.)

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]email.generator.rst`, line 18); *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.generator.rst`, line 18); *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.generator.rst`, line 18); *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.generator.rst`, line 18); *backlink*

Unknown interpreted text role "class".

The :class:`Generator` class can be used to flatten a message into a text (as opposed to binary) serialized representation, but since Unicode cannot represent binary data directly, the message is of necessity transformed into something that contains only ASCII characters, using the standard email RFC Content Transfer Encoding techniques for encoding email messages for transport over channels that are not "8 bit clean".

To accommodate reproducible processing of SMIME-signed messages :class:`Generator` disables header folding for message parts of type `multipart/signed` and all subparts.

```
.. class:: BytesGenerator(outfp, mangle_from_=None, maxheaderlen=None, *, \
                          policy=None)

   Return a :class:`BytesGenerator` object that will write any message provided
   to the :meth:`flatten` method, or any surrogateescape encoded text provided
   to the :meth:`write` method, to the :term:`file-like object` *outfp*.
   *outfp* must support a ``write`` method that accepts binary data.

   If optional *mangle_from_* is ``True``, put a ``>`` character in front of
   any line in the body that starts with the exact string ``"From "``, that is
   ``From`` followed by a space at the beginning of a line.  *mangle_from_*
   defaults to the value of the :attr:`~email.policy.Policy.mangle_from_`
   setting of the *policy* (which is ``True`` for the
   :data:`~email.policy.compat32` policy and ``False`` for all others).
   *mangle_from_* is intended for use when messages are stored in unix mbox
   format (see :mod:`mailbox` and `WHY THE CONTENT-LENGTH FORMAT IS BAD
   <https://www.jwz.org/doc/content-length.html>`_).

   If *maxheaderlen* is not ``None``, refold any header lines that are longer
   than *maxheaderlen*, or if ``0``, do not rewrap any headers.  If
   *manheaderlen* is ``None`` (the default), wrap headers and other message
   lines according to the *policy* settings.

   If *policy* is specified, use that policy to control message generation.  If
   *policy* is ``None`` (the default), use the policy associated with the
   :class:`~email.message.Message` or :class:`~email.message.EmailMessage`
   object passed to ``flatten`` to control the message generation.  See
   :mod:`email.policy` for details on what *policy* controls.

   .. versionadded:: 3.2

   .. versionchanged:: 3.3 Added the *policy* keyword.

   .. versionchanged:: 3.6 The default behavior of the *mangle_from_*
      and *maxheaderlen* parameters is to follow the policy.
```

```
.. method:: flatten(msg, unixfrom=False, linesep=None)

   Print the textual representation of the message object structure rooted
   at *msg* to the output file specified when the :class:`BytesGenerator`
   instance was created.

   If the :mod:`~email.policy` option :attr:`~email.policy.Policy.cte_type`
   is ``8bit`` (the default), copy any headers in the original parsed
   message that have not been modified to the output with any bytes with the
   high bit set reproduced as in the original, and preserve the non-ASCII
   :mailheader:`Content-Transfer-Encoding` of any body parts that have them.
   If ``cte_type`` is ``7bit``, convert the bytes with the high bit set as
   needed using an ASCII-compatible :mailheader:`Content-Transfer-Encoding`.
   That is, transform parts with non-ASCII
   :mailheader:`Content-Transfer-Encoding`
   (:mailheader:`Content-Transfer-Encoding: 8bit`) to an ASCII compatible
   :mailheader:`Content-Transfer-Encoding`, and encode RFC-invalid non-ASCII
   bytes in headers using the MIME ``unknown-8bit`` character set, thus
   rendering them RFC-compliant.

   .. XXX: There should be an option that just does the RFC
      compliance transformation on headers but leaves CTE 8bit parts alone.

   If *unixfrom* is ``True``, print the envelope header delimiter used by
   the Unix mailbox format (see :mod:`mailbox`) before the first of the
   :rfc:`5322` headers of the root message object.  If the root object has
   no envelope header, craft a standard one.  The default is ``False``.
   Note that for subparts, no envelope header is ever printed.

   If *linesep* is not ``None``, use it as the separator character between
   all the lines of the flattened message.  If *linesep* is ``None`` (the
   default), use the value specified in the *policy*.

   .. XXX: flatten should take a *policy* keyword.


.. method:: clone(fp)

   Return an independent clone of this :class:`BytesGenerator` instance with
   the exact same option settings, and *fp* as the new *outfp*.


.. method:: write(s)

   Encode *s* using the ``ASCII`` codec and the ``surrogateescape`` error
   handler, and pass it to the *write* method of the *outfp* passed to the
   :class:`BytesGenerator`'s constructor.
```

As a convenience, :class:`~email.message.EmailMessage` provides the methods :meth:`~email.message.EmailMessage.as_bytes` and `bytes(aMessage)` (a.k.a. :meth:`~email.message.EmailMessage.__bytes__`), which simplify the generation of a serialized binary representation of a message object. For more detail, see :mod:`email.message`.

Because strings cannot represent binary data, the :class:`Generator` class must convert any binary data in any message it flattens to an ASCII compatible format, by converting them to an ASCII compatible :mailheader:`Content-Transfer_Encoding`. Using the terminology of the email RFCs, you can think of this as :class:`Generator` serializing to an I/O stream that is not "8 bit clean". In other words, most applications will want to be using :class:`BytesGenerator`, and not :class:`Generator`.

```
.. class:: Generator(outfp, mangle_from_=None, maxheaderlen=None, *, \
                      policy=None)

   Return a :class:`Generator` object that will write any message provided
   to the :meth:`flatten` method, or any text provided to the :meth:`write`
   method, to the :term:`file-like object` *outfp*.  *outfp* must support a
   ``write`` method that accepts string data.

   If optional *mangle_from_* is ``True``, put a ``>`` character in front of
   any line in the body that starts with the exact string ``"From "``, that is
   ``From`` followed by a space at the beginning of a line.  *mangle_from_*
   defaults to the value of the :attr:`~email.policy.Policy.mangle_from_`
   setting of the *policy* (which is ``True`` for the
   :data:`~email.policy.compat32` policy and ``False`` for all others).
   *mangle_from_* is intended for use when messages are stored in unix mbox
   format (see :mod:`mailbox` and `WHY THE CONTENT-LENGTH FORMAT IS BAD
   <https://www.jwz.org/doc/content-length.html>`_).

   If *maxheaderlen* is not ``None``, refold any header lines that are longer
   than *maxheaderlen*, or if ``0``, do not rewrap any headers.  If
   *manheaderlen* is ``None`` (the default), wrap headers and other message
   lines according to the *policy* settings.

   If *policy* is specified, use that policy to control message generation.  If
   *policy* is ``None`` (the default), use the policy associated with the
   :class:`~email.message.Message` or :class:`~email.message.EmailMessage`
   object passed to ``flatten`` to control the message generation.  See
   :mod:`email.policy` for details on what *policy* controls.

   .. versionchanged:: 3.3 Added the *policy* keyword.

   .. versionchanged:: 3.6 The default behavior of the *mangle_from_*
      and *maxheaderlen* parameters is to follow the policy.


   .. method:: flatten(msg, unixfrom=False, linesep=None)
```

```
        Print the textual representation of the message object structure rooted
        at *msg* to the output file specified when the :class:`Generator`
        instance was created.

        If the :mod:`~email.policy` option :attr:`~email.policy.Policy.cte_type`
        is ``8bit``, generate the message as if the option were set to ``7bit``.
        (This is required because strings cannot represent non-ASCII bytes.)
        Convert any bytes with the high bit set as needed using an
        ASCII-compatible :mailheader:`Content-Transfer-Encoding`.  That is,
        transform parts with non-ASCII :mailheader:`Content-Transfer-Encoding`
        (:mailheader:`Content-Transfer-Encoding: 8bit`) to an ASCII compatible
        :mailheader:`Content-Transfer-Encoding`, and encode RFC-invalid non-ASCII
        bytes in headers using the MIME ``unknown-8bit`` character set, thus
        rendering them RFC-compliant.

        If *unixfrom* is ``True``, print the envelope header delimiter used by
        the Unix mailbox format (see :mod:`mailbox`) before the first of the
        :rfc:`5322` headers of the root message object.  If the root object has
        no envelope header, craft a standard one.  The default is ``False``.
        Note that for subparts, no envelope header is ever printed.

        If *linesep* is not ``None``, use it as the separator character between
        all the lines of the flattened message.  If *linesep* is ``None`` (the
        default), use the value specified in the *policy*.

        .. XXX: flatten should take a *policy* keyword.

        .. versionchanged:: 3.2
           Added support for re-encoding ``8bit`` message bodies, and the
           *linesep* argument.


    .. method:: clone(fp)

        Return an independent clone of this :class:`Generator` instance with the
        exact same options, and *fp* as the new *outfp*.


    .. method:: write(s)

        Write *s* to the *write* method of the *outfp* passed to the
        :class:`Generator`'s constructor.  This provides just enough file-like
        API for :class:`Generator` instances to be used in the :func:`print`
        function.
```

As a convenience, :class:`~email.message.EmailMessage` provides the methods :meth:`~email.message.EmailMessage.as_string` and str(aMessage) (a.k.a. :meth:`~email.message.EmailMessage.__str__`), which simplify the generation of a formatted string representation of a message object. For more detail, see :mod:`email.message`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-`**
`main\Doc\library\[cpython-main][Doc][library]email.generator.rst,` line 228);** *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.generator.rst,` line 228);** *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.generator.rst,` line 228);** *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.generator.rst,` line 228);** *backlink*

Unknown interpreted text role "mod".

The :mod:`email.generator` module also provides a derived class, :class:`DecodedGenerator`, which is like the :class:`Generator` base class, except that non-:mimetype:`text` parts are not serialized, but are instead represented in the output stream by a string derived from a template filled in with information about the part.

Footnotes

[1]   This statement assumes that you use the appropriate setting for `unixfrom`, and that there are no :mod:`policy` settings calling for automatic adjustments (for example, :attr:`~email.policy.Policy.refold_source` must be `none`, which is *not* the default). It is also not 100% true, since if the message does not conform to the RFC standards occasionally information about the exact original text is lost during parsing error recovery. It is a goal to fix these latter edge cases when possible.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]email.generator.rst`, **line 276);** *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]email.generator.rst`, **line 276);** *backlink*

Unknown interpreted text role "attr".