:mod:`socket` --- Low-level networking interface

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.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]socket.rst, line 4)

Unknown directive type "module".

```
.. module:: socket
   :synopsis: Low-level networking interface.
```

Source code: :source:`Lib/socket.py`

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

Unknown interpreted text role "source".

This module provides access to the BSD *socket* interface. It is available on all modern Unix systems, Windows, MacOS, and probably additional platforms.

Note

Some behavior may be platform dependent, since calls are made to the operating system socket APIs.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 19)

Unknown directive type "index".

```
.. index:: object: socket
```

The Python interface is a straightforward transliteration of the Unix system call and library interface for sockets to Python's object-oriented style: the :func:`.socket` function returns a :dfn:`socket object` whose methods implement the various socket system calls. Parameter types are somewhat higher-level than in the C interface: as with :meth:`read` and :meth:`write` operations on Python files, buffer allocation on receive operations is automatic, and buffer length is implicit on send operations.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 21); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 21); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 21); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 21); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 30)

Unknown directive type "seealso".

```
.. seealso::

   Module :mod:`socketserver`
      Classes that simplify writing network servers.
```

```
Module :mod:`ssl`
   A TLS/SSL wrapper for socket objects.
```

Socket families

Depending on the system and the build options, various socket families are supported by this module.

The address format required by a particular socket object is automatically selected based on the address family specified when the socket object was created. Socket addresses are represented as follows:

- The address of an :const:`AF_UNIX` socket bound to a file system node is represented as a string, using the file system encoding and the `'surrogateescape'` error handler (see PEP 383). An address in Linux's abstract namespace is returned as a :term:`bytes-like object` with an initial null byte; note that sockets in this namespace can communicate with normal file system sockets, so programs intended to run on Linux may need to deal with both types of address. A string or bytes-like object can be used for either type of address when passing it as an argument.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 49); *backlink*

  Unknown interpreted text role "const".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 49); *backlink*

  Unknown interpreted text role "term".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 59)

  Unknown directive type "versionchanged".

  ```
  .. versionchanged:: 3.3
     Previously, :const:`AF_UNIX` socket paths were assumed to use UTF-8
     encoding.
  ```

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 63)

  Unknown directive type "versionchanged".

  ```
  .. versionchanged:: 3.5
     Writable :term:`bytes-like object` is now accepted.
  ```

- A pair `(host, port)` is used for the :const:`AF_INET` address family, where *host* is a string representing either a hostname in internet domain notation like `'daring.cwi.nl'` or an IPv4 address like `'100.50.200.5'`, and *port* is an integer.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 68); *backlink*

  Unknown interpreted text role "const".

  - For IPv4 addresses, two special forms are accepted instead of a host address: `''` represents :const:`INADDR_ANY`, which is used to bind to all interfaces, and the string `'<broadcast>'` represents :const:`INADDR_BROADCAST`. This behavior is not compatible with IPv6, therefore, you may want to avoid these if you intend to support IPv6 with your Python programs.

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 73); *backlink*

    Unknown interpreted text role "const".

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 73); *backlink*

    Unknown interpreted text role "const".

- For :const:`AF_INET6` address family, a four-tuple `(host, port, flowinfo, scope_id)` is used, where *flowinfo* and *scope_id* represent the `sin6_flowinfo` and `sin6_scope_id` members in :const:`struct sockaddr_in6` in C. For :mod:`socket` module methods, *flowinfo* and *scope_id* can be omitted just for backward compatibility. Note, however, omission of *scope_id* can cause problems in manipulating scoped IPv6 addresses.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 80); *backlink*

  Unknown interpreted text role "const".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 80); *backlink*

  Unknown interpreted text role "const".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 80); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 87)

  Unknown directive type "versionchanged".

  ```
  .. versionchanged:: 3.7
     For multicast addresses (with *scope_id* meaningful) *address* may not contain
     ``%scope_id`` (or ``zone id``) part. This information is superfluous and may
     be safely omitted (recommended).
  ```

- :const:`AF_NETLINK` sockets are represented as pairs `(pid, groups)`.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 92); *backlink*

  Unknown interpreted text role "const".

- Linux-only support for TIPC is available using the :const:`AF_TIPC` address family. TIPC is an open, non-IP based networked protocol designed for use in clustered computer environments. Addresses are represented by a tuple, and the fields depend on the address type. The general tuple form is `(addr_type, v1, v2, v3 [, scope])`, where:

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 94); *backlink*

  Unknown interpreted text role "const".

  - *addr_type* is one of :const:`TIPC_ADDR_NAMESEQ`, :const:`TIPC_ADDR_NAME`, or :const:`TIPC_ADDR_ID`.

    System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 100); *backlink*

    Unknown interpreted text role "const".

    System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 100); *backlink*

    Unknown interpreted text role "const".

    System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 100); *backlink*

    Unknown interpreted text role "const".

  - *scope* is one of :const:`TIPC_ZONE_SCOPE`, :const:`TIPC_CLUSTER_SCOPE`, and

:const:`TIPC_NODE_SCOPE`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 102); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 102); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 102); *backlink*

Unknown interpreted text role "const".

- If *addr_type* is :const:`TIPC_ADDR_NAME`, then *v1* is the server type, *v2* is the port identifier, and *v3* should be 0.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 104); *backlink*

Unknown interpreted text role "const".

If *addr_type* is :const:`TIPC_ADDR_NAMESEQ`, then *v1* is the server type, *v2* is the lower port number, and *v3* is the upper port number.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 107); *backlink*

Unknown interpreted text role "const".

If *addr_type* is :const:`TIPC_ADDR_ID`, then *v1* is the node, *v2* is the reference, and *v3* should be set to 0.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 110); *backlink*

Unknown interpreted text role "const".

- A tuple `(interface, )` is used for the :const:`AF_CAN` address family, where *interface* is a string representing a network interface name like `'can0'`. The network interface name `''` can be used to receive packets from all network interfaces of this family.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 113); *backlink*

Unknown interpreted text role "const".

- :const:`CAN_ISOTP` protocol require a tuple `(interface, rx_addr, tx_addr)` where both additional parameters are unsigned long integer that represent a CAN identifier (standard or extended).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 118); *backlink*

Unknown interpreted text role "const".

- :const:`CAN_J1939` protocol require a tuple `(interface, name, pgn, addr)` where additional parameters are 64-bit unsigned integer representing the ECU name, a 32-bit unsigned integer representing the Parameter Group Number (PGN), and an 8-bit integer representing the address.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 121); *backlink*

Unknown interpreted text role "const".

- A string or a tuple `(id, unit)` is used for the :const:`SYSPROTO_CONTROL` protocol of the :const:`PF_SYSTEM` family. The string is the name of a kernel control using a dynamically-assigned ID. The tuple can be used if ID and unit number of the kernel control are known or if a registered ID is used.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 126); *backlink*

  Unknown interpreted text role "const".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 126); *backlink*

  Unknown interpreted text role "const".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 132)

  Unknown directive type "versionadded".

  ```
  .. versionadded:: 3.3
  ```

- :const:`AF_BLUETOOTH` supports the following protocols and address formats:

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 134); *backlink*

  Unknown interpreted text role "const".

  - :const:`BTPROTO_L2CAP` accepts `(bdaddr, psm)` where `bdaddr` is the Bluetooth address as a string and `psm` is an integer.

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 137); *backlink*

    Unknown interpreted text role "const".

  - :const:`BTPROTO_RFCOMM` accepts `(bdaddr, channel)` where `bdaddr` is the Bluetooth address as a string and `channel` is an integer.

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 140); *backlink*

    Unknown interpreted text role "const".

  - :const:`BTPROTO_HCI` accepts `(device_id,)` where `device_id` is either an integer or a string with the Bluetooth address of the interface. (This depends on your OS; NetBSD and DragonFlyBSD expect a Bluetooth address while everything else expects an integer.)

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 143); *backlink*

    Unknown interpreted text role "const".

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 148)

    Unknown directive type "versionchanged".

    ```
    .. versionchanged:: 3.2
       NetBSD and DragonFlyBSD support added.
    ```

  - :const:`BTPROTO_SCO` accepts `bdaddr` where `bdaddr` is a :class:`bytes` object containing the Bluetooth address in a string format. (ex. `b'12:23:34:45:56:67'`) This protocol is not supported under FreeBSD.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 151); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 151); *backlink*

Unknown interpreted text role "class".

- :const:`AF_ALG` is a Linux-only socket based interface to Kernel cryptography. An algorithm socket is configured with a tuple of two to four elements `(type, name [, feat [, mask]])`, where:

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 156); *backlink*

  Unknown interpreted text role "const".

  - *type* is the algorithm type as string, e.g. `aead`, `hash`, `skcipher` or `rng`.
  - *name* is the algorithm name and operation mode as string, e.g. `sha256`, `hmac(sha256)`, `cbc(aes)` or `drbg_nopr_ctr_aes256`.
  - *feat* and *mask* are unsigned 32bit integers.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 168)

  Unknown directive type "availability".

  ```
  .. availability:: Linux 2.6.38, some algorithm types require more recent Kernels.
  ```

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 170)

  Unknown directive type "versionadded".

  ```
  .. versionadded:: 3.6
  ```

- :const:`AF_VSOCK` allows communication between virtual machines and their hosts. The sockets are represented as a `(CID, port)` tuple where the context ID or CID and port are integers.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 172); *backlink*

  Unknown interpreted text role "const".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 176)

  Unknown directive type "availability".

  ```
  .. availability:: Linux >= 4.8 QEMU >= 2.8 ESX >= 4.0 ESX Workstation >= 6.5.
  ```

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 178)

  Unknown directive type "versionadded".

  ```
  .. versionadded:: 3.7
  ```

- :const:`AF_PACKET` is a low-level interface directly to network devices. The packets are represented by the tuple `(ifname, proto[, pkttype[, hatype[, addr]]])` where:

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 180); *backlink*

  Unknown interpreted text role "const".

- *ifname* - String specifying the device name.
- *proto* - An in network-byte-order integer specifying the Ethernet protocol number.
- *pkttype* - Optional integer specifying the packet type:
  - `PACKET_HOST` (the default) - Packet addressed to the local host.
  - `PACKET_BROADCAST` - Physical-layer broadcast packet.
  - `PACKET_MULTICAST` - Packet sent to a physical-layer multicast address.
  - `PACKET_OTHERHOST` - Packet to some other host that has been caught by a device driver in promiscuous mode.
  - `PACKET_OUTGOING` - Packet originating from the local host that is looped back to a packet socket.
- *hatype* - Optional integer specifying the ARP hardware address type.
- *addr* - Optional bytes-like object specifying the hardware physical address, whose interpretation depends on the device.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 200)**
>
> Unknown directive type "availability".
>
> ```
> .. availability:: Linux >= 2.2.
> ```

- :const:`AF_QIPCRTR` is a Linux-only socket based interface for communicating with services running on co-processors in Qualcomm platforms. The address family is represented as a `(node, port)` tuple where the *node* and *port* are non-negative integers.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 202); *backlink***
>
> Unknown interpreted text role "const".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 207)**
>
> Unknown directive type "availability".
>
> ```
> .. availability:: Linux >= 4.7.
> ```

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 209)**
>
> Unknown directive type "versionadded".
>
> ```
> .. versionadded:: 3.8
> ```

- :const:`IPPROTO_UDPLITE` is a variant of UDP which allows you to specify what portion of a packet is covered with the checksum. It adds two socket options that you can change. `self.setsockopt(IPPROTO_UDPLITE, UDPLITE_SEND_CSCOV, length)` will change what portion of outgoing packets are covered by the checksum and `self.setsockopt(IPPROTO_UDPLITE, UDPLITE_RECV_CSCOV, length)` will filter out packets which cover too little of their data. In both cases `length` should be in `range(8, 2**16, 8)`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 211); *backlink***
>
> Unknown interpreted text role "const".

  Such a socket should be constructed with `socket(AF_INET, SOCK_DGRAM, IPPROTO_UDPLITE)` for IPv4 or `socket(AF_INET6, SOCK_DGRAM, IPPROTO_UDPLITE)` for IPv6.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 224)**
>
> Unknown directive type "availability".
>
> ```
> .. availability:: Linux >= 2.6.20, FreeBSD >= 10.1-RELEASE
> ```

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 226)**
>
> Unknown directive type "versionadded".
>
> ```
> .. versionadded:: 3.9
> ```

If you use a hostname in the *host* portion of IPv4/v6 socket address, the program may show a nondeterministic behavior, as Python uses the first address returned from the DNS resolution. The socket address will be resolved differently into an actual IPv4/v6 address, depending on the results from DNS resolution and/or the host configuration. For deterministic behavior use a numeric address in *host* portion.

All errors raise exceptions. The normal exceptions for invalid argument types and out-of-memory conditions can be raised; starting from Python 3.3, errors related to socket or address semantics raise :exc:`OSError` or one of its subclasses (they used to raise :exc:`socket.error`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 235); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 235); *backlink*

Unknown interpreted text role "exc".

Non-blocking mode is supported through :meth:`~socket.setblocking`. A generalization of this based on timeouts is supported through :meth:`~socket.settimeout`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 240); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 240); *backlink*

Unknown interpreted text role "meth".

Module contents

The module :mod:`socket` exports the following elements.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 248); *backlink*

Unknown interpreted text role "mod".

Exceptions

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 254)

Unknown directive type "exception".

```
.. exception:: error

   A deprecated alias of :exc:`OSError`.

   .. versionchanged:: 3.3
      Following :pep:`3151`, this class was made an alias of :exc:`OSError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 262)

Unknown directive type "exception".

```
.. exception:: herror

   A subclass of :exc:`OSError`, this exception is raised for
   address-related errors, i.e. for functions that use *h_errno* in the POSIX
   C API, including :func:`gethostbyname_ex` and :func:`gethostbyaddr`.
   The accompanying value is a pair ``(h_errno, string)`` representing an
   error returned by a library call.  *h_errno* is a numeric value, while
   *string* represents the description of *h_errno*, as returned by the
   :c:func:`hstrerror` C function.

   .. versionchanged:: 3.3
      This class was made a subclass of :exc:`OSError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 275)

Unknown directive type "exception".

```
.. exception:: gaierror

   A subclass of :exc:`OSError`, this exception is raised for
   address-related errors by :func:`getaddrinfo` and :func:`getnameinfo`.
   The accompanying value is a pair ``(error, string)`` representing an error
   returned by a library call.  *string* represents the description of
   *error*, as returned by the :c:func:`gai_strerror` C function.  The
   numeric *error* value will match one of the :const:`EAI_\*` constants
   defined in this module.

   .. versionchanged:: 3.3
      This class was made a subclass of :exc:`OSError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 288)

Unknown directive type "exception".

```
.. exception:: timeout

   A deprecated alias of :exc:`TimeoutError`.

   A subclass of :exc:`OSError`, this exception is raised when a timeout
   occurs on a socket which has had timeouts enabled via a prior call to
   :meth:`~socket.settimeout` (or implicitly through
   :func:`~socket.setdefaulttimeout`).  The accompanying value is a string
   whose value is currently always "timed out".

   .. versionchanged:: 3.3
      This class was made a subclass of :exc:`OSError`.

   .. versionchanged:: 3.10
      This class was made an alias of :exc:`TimeoutError`.
```

Constants

The AF_* and SOCK_* constants are now :class:`AddressFamily` and :class:`SocketKind` :class:`.IntEnum` collections.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 308); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 308); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 308); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 311)

Unknown directive type "versionadded".

```
.. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 313)

Unknown directive type "data".

```
.. data:: AF_UNIX
          AF_INET
          AF_INET6
```

```
   These constants represent the address (and protocol) families, used for the
   first argument to :func:`.socket`.  If the :const:`AF_UNIX` constant is not
   defined then this protocol is unsupported.  More constants may be available
   depending on the system.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 323)

Unknown directive type "data".

```
.. data:: SOCK_STREAM
          SOCK_DGRAM
          SOCK_RAW
          SOCK_RDM
          SOCK_SEQPACKET

   These constants represent the socket types, used for the second argument to
   :func:`.socket`.  More constants may be available depending on the system.
   (Only :const:`SOCK_STREAM` and :const:`SOCK_DGRAM` appear to be generally
   useful.)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 334)

Unknown directive type "data".

```
.. data:: SOCK_CLOEXEC
          SOCK_NONBLOCK

   These two constants, if defined, can be combined with the socket types and
   allow you to set some flags atomically (thus avoiding possible race
   conditions and the need for separate calls).

   .. seealso::

      `Secure File Descriptor Handling <http://udrepper.livejournal.com/20407.html>`_
      for a more thorough explanation.

   .. availability:: Linux >= 2.6.27.

   .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 350)

Unknown directive type "data".

```
.. data:: SO_*
          SOMAXCONN
          MSG_*
          SOL_*
          SCM_*
          IPPROTO_*
          IPPORT_*
          INADDR_*
          IP_*
          IPV6_*
          EAI_*
          AI_*
          NI_*
          TCP_*

   Many constants of these forms, documented in the Unix documentation on sockets
   and/or the IP protocol, are also defined in the socket module. They are
   generally used in arguments to the :meth:`setsockopt` and :meth:`getsockopt`
   methods of socket objects.  In most cases, only those symbols that are defined
   in the Unix header files are defined; for a few symbols, default values are
   provided.

   .. versionchanged:: 3.6
      ``SO_DOMAIN``, ``SO_PROTOCOL``, ``SO_PEERSEC``, ``SO_PASSSEC``,
      ``TCP_USER_TIMEOUT``, ``TCP_CONGESTION`` were added.

   .. versionchanged:: 3.6.5
      On Windows, ``TCP_FASTOPEN``, ``TCP_KEEPCNT`` appear if run-time Windows
      supports.

   .. versionchanged:: 3.7
      ``TCP_NOTSENT_LOWAT`` was added.

      On Windows, ``TCP_KEEPIDLE``, ``TCP_KEEPINTVL`` appear if run-time Windows
      supports.
```

```
   .. versionchanged:: 3.10
      ``IP_RECVTOS`` was added.
       Added ``TCP_KEEPALIVE``. On MacOS this constant can be used in the same
       way that ``TCP_KEEPIDLE`` is used on Linux.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 391)

Unknown directive type "data".

```
   .. data:: AF_CAN
             PF_CAN
             SOL_CAN_*
             CAN_*

      Many constants of these forms, documented in the Linux documentation, are
      also defined in the socket module.

      .. availability:: Linux >= 2.6.25, NetBSD >= 8.

      .. versionadded:: 3.3

      .. versionchanged:: 3.11
         NetBSD support was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 406)

Unknown directive type "data".

```
   .. data:: CAN_BCM
             CAN_BCM_*

      CAN_BCM, in the CAN protocol family, is the broadcast manager (BCM) protocol.
      Broadcast manager constants, documented in the Linux documentation, are also
      defined in the socket module.

      .. availability:: Linux >= 2.6.25.

      .. note::
         The :data:`CAN_BCM_CAN_FD_FRAME` flag is only available on Linux >= 4.8.

      .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 420)

Unknown directive type "data".

```
   .. data:: CAN_RAW_FD_FRAMES

      Enables CAN FD support in a CAN_RAW socket. This is disabled by default.
      This allows your application to send both CAN and CAN FD frames; however,
      you must accept both CAN and CAN FD frames when reading from the socket.

      This constant is documented in the Linux documentation.

      .. availability:: Linux >= 3.6.

      .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 432)

Unknown directive type "data".

```
   .. data:: CAN_RAW_JOIN_FILTERS

      Joins the applied CAN filters such that only CAN frames that match all
      given CAN filters are passed to user space.

      This constant is documented in the Linux documentation.

      .. availability:: Linux >= 4.1.

      .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 443)

Unknown directive type "data".

```
.. data:: CAN_ISOTP

   CAN_ISOTP, in the CAN protocol family, is the ISO-TP (ISO 15765-2) protocol.
   ISO-TP constants, documented in the Linux documentation.

   .. availability:: Linux >= 2.6.25.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 452)

Unknown directive type "data".

```
.. data:: CAN_J1939

   CAN_J1939, in the CAN protocol family, is the SAE J1939 protocol.
   J1939 constants, documented in the Linux documentation.

   .. availability:: Linux >= 5.4.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 462)

Unknown directive type "data".

```
.. data:: AF_PACKET
          PF_PACKET
          PACKET_*

   Many constants of these forms, documented in the Linux documentation, are
   also defined in the socket module.

   .. availability:: Linux >= 2.2.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 472)

Unknown directive type "data".

```
.. data:: AF_RDS
          PF_RDS
          SOL_RDS
          RDS_*

   Many constants of these forms, documented in the Linux documentation, are
   also defined in the socket module.

   .. availability:: Linux >= 2.6.30.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 485)

Unknown directive type "data".

```
.. data:: SIO_RCVALL
          SIO_KEEPALIVE_VALS
          SIO_LOOPBACK_FAST_PATH
          RCVALL_*

   Constants for Windows' WSAIoctl(). The constants are used as arguments to the
   :meth:`~socket.socket.ioctl` method of socket objects.

   .. versionchanged:: 3.6
      ``SIO_LOOPBACK_FAST_PATH`` was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 497)

Unknown directive type "data".

```
.. data:: TIPC_*

   TIPC related constants, matching the ones exported by the C socket API. See
   the TIPC documentation for more information.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 502)

Unknown directive type "data".

```
.. data:: AF_ALG
          SOL_ALG
          ALG_*

   Constants for Linux Kernel cryptography.

   .. availability:: Linux >= 2.6.38.

   .. versionadded:: 3.6
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 513)

Unknown directive type "data".

```
.. data:: AF_VSOCK
          IOCTL_VM_SOCKETS_GET_LOCAL_CID
          VMADDR*
          SO_VM*

   Constants for Linux host/guest communication.

   .. availability:: Linux >= 4.8.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 524)

Unknown directive type "data".

```
.. data:: AF_LINK

  .. availability:: BSD, macOS.

  .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 530)

Unknown directive type "data".

```
.. data:: has_ipv6

   This constant contains a boolean value which indicates if IPv6 is supported on
   this platform.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 535)

Unknown directive type "data".

```
.. data:: BDADDR_ANY
          BDADDR_LOCAL

   These are string constants containing Bluetooth addresses with special
   meanings. For example, :const:`BDADDR_ANY` can be used to indicate
   any address when specifying the binding socket with
   :const:`BTPROTO_RFCOMM`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 543)

Unknown directive type "data".

```
.. data:: HCI_FILTER
          HCI_TIME_STAMP
```

```
        HCI_DATA_DIR

    For use with :const:`BTPROTO_HCI`. :const:`HCI_FILTER` is not
    available for NetBSD or DragonFlyBSD. :const:`HCI_TIME_STAMP` and
    :const:`HCI_DATA_DIR` are not available for FreeBSD, NetBSD, or
    DragonFlyBSD.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 552)

Unknown directive type "data".

```
.. data:: AF_QIPCRTR

   Constant for Qualcomm's IPC router protocol, used to communicate with
   service providing remote processors.

   .. availability:: Linux >= 4.7.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 559)

Unknown directive type "data".

```
.. data:: SCM_CREDS2
          LOCAL_CREDS
          LOCAL_CREDS_PERSISTENT

   LOCAL_CREDS and LOCAL_CREDS_PERSISTENT can be used
   with SOCK_DGRAM, SOCK_STREAM sockets, equivalent to
   Linux/DragonFlyBSD SO_PASSCRED, while LOCAL_CREDS
   sends the credentials at first read, LOCAL_CREDS_PERSISTENT
   sends for each read, SCM_CREDS2 must be then used for
   the latter for the message type.

   .. versionadded:: 3.11

   .. availability:: FreeBSD.
```

Functions

Creating sockets

The following functions all create :ref:`socket objects <socket-objects>`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 580); *backlink*

Unknown interpreted text role "ref".

Create a new socket using the given address family, socket type and protocol number. The address family should be :const:`AF_INET` (the default), :const:`AF_INET6`, :const:`AF_UNIX`, :const:`AF_CAN`, :const:`AF_PACKET`, or :const:`AF_RDS`. The socket type should be :const:`SOCK_STREAM` (the default), :const:`SOCK_DGRAM`, :const:`SOCK_RAW` or perhaps one of the other `SOCK_` constants. The protocol number is usually zero and may be omitted or in the case where the address family is :const:`AF_CAN` the protocol should be one of :const:`CAN_RAW`, :const:`CAN_BCM`, :const:`CAN_ISOTP` or :const:`CAN_J1939`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 585); *backlink*

Unknown interpreted text role "const".

If *fileno* is specified, the values for *family*, *type*, and *proto* are auto-detected from the specified file descriptor. Auto-detection can be overruled by calling the function with explicit *family*, *type*, or *proto* arguments. This only affects how Python represents e.g. the return value of :meth:`socket.getpeername` but not the actual OS resource. Unlike :func:`socket.fromfd`, *fileno* will return the same socket and not a duplicate. This may help close a detached socket using :meth:`socket.close()`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 595); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 595); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]socket.rst, line 595); *backlink*

Unknown interpreted text role "meth".

The newly created socket is :ref:`non-inheritable <fd_inheritance>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 604); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 606)

Unknown directive type "audit-event".

```
.. audit-event:: socket.__new__ self,family,type,protocol socket.socket
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 608)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   The AF_CAN family was added.
   The AF_RDS family was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 612)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.4
    The CAN_BCM protocol was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 615)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.4
   The returned socket is now non-inheritable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 618)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
    The CAN_ISOTP protocol was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 621)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   When :const:`SOCK_NONBLOCK` or :const:`SOCK_CLOEXEC`
   bit flags are applied to *type* they are cleared, and
   :attr:`socket.type` will not reflect them.  They are still passed
   to the underlying system `socket()` call.  Therefore,

   ::

       sock = socket.socket(
           socket.AF_INET,
           socket.SOCK_STREAM | socket.SOCK_NONBLOCK)

   will still create a non-blocking socket on OSes that support
   ``SOCK_NONBLOCK``, but ``sock.type`` will be set to
   ``socket.SOCK_STREAM``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 637)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.9
    The CAN_J1939 protocol was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 640)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
    The IPPROTO_MPTCP protocol was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 643)

Unknown directive type "function".

```
.. function:: socketpair([family[, type[, proto]]])

   Build a pair of connected socket objects using the given address family, socket
   type, and protocol number.  Address family, socket type, and protocol number are
   as for the :func:`.socket` function above. The default family is :const:`AF_UNIX`
   if defined on the platform; otherwise, the default is :const:`AF_INET`.

   The newly created sockets are :ref:`non-inheritable <fd_inheritance>`.

   .. versionchanged:: 3.2
      The returned socket objects now support the whole socket API, rather
      than a subset.

   .. versionchanged:: 3.4
      The returned sockets are now non-inheritable.

   .. versionchanged:: 3.5
      Windows support added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 663)

Unknown directive type "function".

```
.. function:: create_connection(address[, timeout[, source_address]])

   Connect to a TCP service listening on the internet *address* (a 2-tuple
   ``(host, port)``), and return the socket object.  This is a higher-level
   function than :meth:`socket.connect`: if *host* is a non-numeric hostname,
   it will try to resolve it for both :data:`AF_INET` and :data:`AF_INET6`,
   and then try to connect to all possible addresses in turn until a
   connection succeeds.  This makes it easy to write clients that are
   compatible to both IPv4 and IPv6.

   Passing the optional *timeout* parameter will set the timeout on the
   socket instance before attempting to connect.  If no *timeout* is
   supplied, the global default timeout setting returned by
   :func:`getdefaulttimeout` is used.

   If supplied, *source_address* must be a 2-tuple ``(host, port)`` for the
   socket to bind to as its source address before connecting.  If host or port
   are '' or 0 respectively the OS default behavior will be used.

   .. versionchanged:: 3.2
      *source_address* was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 685)

Unknown directive type "function".

```
.. function:: create_server(address, *, family=AF_INET, backlog=None, reuse_port=False, dualstack_ipv6=

   Convenience function which creates a TCP socket bound to *address* (a 2-tuple
   ``(host, port)``) and return the socket object.

   *family* should be either :data:`AF_INET` or :data:`AF_INET6`.
   *backlog* is the queue size passed to :meth:`socket.listen`; when ``0``
   a default reasonable value is chosen.
   *reuse_port* dictates whether to set the :data:`SO_REUSEPORT` socket option.

   If *dualstack_ipv6* is true and the platform supports it the socket will
   be able to accept both IPv4 and IPv6 connections, else it will raise
   :exc:`ValueError`. Most POSIX platforms and Windows are supposed to support
   this functionality.
   When this functionality is enabled the address returned by
   :meth:`socket.getpeername` when an IPv4 connection occurs will be an IPv6
   address represented as an IPv4-mapped IPv6 address.
   If *dualstack_ipv6* is false it will explicitly disable this functionality
   on platforms that enable it by default (e.g. Linux).
```

```
This parameter can be used in conjunction with :func:`has_dualstack_ipv6`:

::

  import socket

  addr = ("", 8080)  # all interfaces, port 8080
  if socket.has_dualstack_ipv6():
      s = socket.create_server(addr, family=socket.AF_INET6, dualstack_ipv6=True)
  else:
      s = socket.create_server(addr)

.. note::
 On POSIX platforms the :data:`SO_REUSEADDR` socket option is set in order to
 immediately reuse previous sockets which were bound on the same *address*
 and remained in TIME_WAIT state.

.. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 723)

Unknown directive type "function".

```
.. function:: has_dualstack_ipv6()

   Return ``True`` if the platform supports creating a TCP socket which can
   handle both IPv4 and IPv6 connections.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 730)

Unknown directive type "function".

```
.. function:: fromfd(fd, family, type, proto=0)

   Duplicate the file descriptor *fd* (an integer as returned by a file object's
   :meth:`fileno` method) and build a socket object from the result.  Address
   family, socket type and protocol number are as for the :func:`.socket` function
   above. The file descriptor should refer to a socket, but this is not checked ---
   subsequent operations on the object may fail if the file descriptor is invalid.
   This function is rarely needed, but can be used to get or set socket options on
   a socket passed to a program as standard input or output (such as a server
   started by the Unix inet daemon).  The socket is assumed to be in blocking mode.

   The newly created socket is :ref:`non-inheritable <fd_inheritance>`.

   .. versionchanged:: 3.4
      The returned socket is now non-inheritable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 747)

Unknown directive type "function".

```
.. function:: fromshare(data)

   Instantiate a socket from data obtained from the :meth:`socket.share`
   method.  The socket is assumed to be in blocking mode.

   .. availability:: Windows.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 757)

Unknown directive type "data".

```
.. data:: SocketType

   This is a Python type object that represents the socket object type. It is the
   same as ``type(socket(...))``.
```

Other functions

The :mod:`socket` module also offers various network-related services:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 766); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 769)

Unknown directive type "function".

```
.. function:: close(fd)

   Close a socket file descriptor. This is like :func:`os.close`, but for
   sockets. On some platforms (most noticeable Windows) :func:`os.close`
   does not work for socket file descriptors.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 777)

Unknown directive type "function".

```
.. function:: getaddrinfo(host, port, family=0, type=0, proto=0, flags=0)

   Translate the *host*/*port* argument into a sequence of 5-tuples that contain
   all the necessary arguments for creating a socket connected to that service.
   *host* is a domain name, a string representation of an IPv4/v6 address
   or ``None``. *port* is a string service name such as ``'http'``, a numeric
   port number or ``None``.  By passing ``None`` as the value of *host*
   and *port*, you can pass ``NULL`` to the underlying C API.

   The *family*, *type* and *proto* arguments can be optionally specified
   in order to narrow the list of addresses returned.  Passing zero as a
   value for each of these arguments selects the full range of results.
   The *flags* argument can be one or several of the ``AI_*`` constants,
   and will influence how results are computed and returned.
   For example, :const:`AI_NUMERICHOST` will disable domain name resolution
   and will raise an error if *host* is a domain name.

   The function returns a list of 5-tuples with the following structure:

   ``(family, type, proto, canonname, sockaddr)``

   In these tuples, *family*, *type*, *proto* are all integers and are
   meant to be passed to the :func:`.socket` function.  *canonname* will be
   a string representing the canonical name of the *host* if
   :const:`AI_CANONNAME` is part of the *flags* argument; else *canonname*
   will be empty.  *sockaddr* is a tuple describing a socket address, whose
   format depends on the returned *family* (a ``(address, port)`` 2-tuple for
   :const:`AF_INET`, a ``(address, port, flowinfo, scope_id)`` 4-tuple for
   :const:`AF_INET6`), and is meant to be passed to the :meth:`socket.connect`
   method.

   .. audit-event:: socket.getaddrinfo host,port,family,type,protocol socket.getaddrinfo

   The following example fetches address information for a hypothetical TCP
   connection to ``example.org`` on port 80 (results may differ on your
   system if IPv6 isn't enabled)::

      >>> socket.getaddrinfo("example.org", 80, proto=socket.IPPROTO_TCP)
      [(socket.AF_INET6, socket.SOCK_STREAM,
       6, '', ('2606:2800:220:1:248:1893:25c8:1946', 80, 0, 0)),
       (socket.AF_INET, socket.SOCK_STREAM,
       6, '', ('93.184.216.34', 80))]

   .. versionchanged:: 3.2
      parameters can now be passed using keyword arguments.

   .. versionchanged:: 3.7
      for IPv6 multicast addresses, string representing an address will not
      contain ``%scope_id`` part.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 827)

Unknown directive type "function".

```
.. function:: getfqdn([name])

   Return a fully qualified domain name for *name*. If *name* is omitted or empty,
   it is interpreted as the local host.  To find the fully qualified name, the
```

```
hostname returned by :func:`gethostbyaddr` is checked, followed by aliases for the
host, if available.  The first name which includes a period is selected.  In
case no fully qualified domain name is available and *name* was provided,
it is returned unchanged.  If *name* was empty or equal to ``'0.0.0.0'``,
the hostname from :func:`gethostname` is returned.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 838)

Unknown directive type "function".

```
.. function:: gethostbyname(hostname)

   Translate a host name to IPv4 address format.  The IPv4 address is returned as a
   string, such as  ``'100.50.200.5'``.  If the host name is an IPv4 address itself
   it is returned unchanged.  See :func:`gethostbyname_ex` for a more complete
   interface. :func:`gethostbyname` does not support IPv6 name resolution, and
   :func:`getaddrinfo` should be used instead for IPv4/v6 dual stack support.

   .. audit-event:: socket.gethostbyname hostname socket.gethostbyname
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 849)

Unknown directive type "function".

```
.. function:: gethostbyname_ex(hostname)

   Translate a host name to IPv4 address format, extended interface. Return a
   triple ``(hostname, aliaslist, ipaddrlist)`` where *hostname* is the host's
   primary host name, *aliaslist* is a (possibly
   empty) list of alternative host names for the same address, and *ipaddrlist* is
   a list of IPv4 addresses for the same interface on the same host (often but not
   always a single address). :func:`gethostbyname_ex` does not support IPv6 name
   resolution, and :func:`getaddrinfo` should be used instead for IPv4/v6 dual
   stack support.

   .. audit-event:: socket.gethostbyname hostname socket.gethostbyname_ex
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 863)

Unknown directive type "function".

```
.. function:: gethostname()

   Return a string containing the hostname of the machine where  the Python
   interpreter is currently executing.

   .. audit-event:: socket.gethostname "" socket.gethostname

   Note: :func:`gethostname` doesn't always return the fully qualified domain
   name; use :func:`getfqdn` for that.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 874)

Unknown directive type "function".

```
.. function:: gethostbyaddr(ip_address)

   Return a triple ``(hostname, aliaslist, ipaddrlist)`` where *hostname* is the
   primary host name responding to the given *ip_address*, *aliaslist* is a
   (possibly empty) list of alternative host names for the same address, and
   *ipaddrlist* is a list of IPv4/v6 addresses for the same interface on the same
   host (most likely containing only a single address). To find the fully qualified
   domain name, use the function :func:`getfqdn`. :func:`gethostbyaddr` supports
   both IPv4 and IPv6.

   .. audit-event:: socket.gethostbyaddr ip_address socket.gethostbyaddr
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 887)

Unknown directive type "function".

```
.. function:: getnameinfo(sockaddr, flags)

   Translate a socket address *sockaddr* into a 2-tuple ``(host, port)``. Depending
   on the settings of *flags*, the result can contain a fully-qualified domain name
   or numeric address representation in *host*.  Similarly, *port* can contain a
   string port name or a numeric port number.

   For IPv6 addresses, ``%scope_id`` is appended to the host part if *sockaddr*
   contains meaningful *scope_id*. Usually this happens for multicast addresses.

   For more information about *flags* you can consult :manpage:`getnameinfo(3)`.

   .. audit-event:: socket.getnameinfo sockaddr socket.getnameinfo
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 901)

Unknown directive type "function".

```
.. function:: getprotobyname(protocolname)

   Translate an internet protocol name (for example, ``'icmp'``) to a constant
   suitable for passing as the (optional) third argument to the :func:`.socket`
   function.  This is usually only needed for sockets opened in "raw" mode
   (:const:`SOCK_RAW`); for the normal socket modes, the correct protocol is chosen
   automatically if the protocol is omitted or zero.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 910)

Unknown directive type "function".

```
.. function:: getservbyname(servicename[, protocolname])

   Translate an internet service name and protocol name to a port number for that
   service.  The optional protocol name, if given, should be ``'tcp'`` or
   ``'udp'``, otherwise any protocol will match.

   .. audit-event:: socket.getservbyname servicename,protocolname socket.getservbyname
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 919)

Unknown directive type "function".

```
.. function:: getservbyport(port[, protocolname])

   Translate an internet port number and protocol name to a service name for that
   service.  The optional protocol name, if given, should be ``'tcp'`` or
   ``'udp'``, otherwise any protocol will match.

   .. audit-event:: socket.getservbyport port,protocolname socket.getservbyport
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 928)

Unknown directive type "function".

```
.. function:: ntohl(x)

   Convert 32-bit positive integers from network to host byte order.  On machines
   where the host byte order is the same as network byte order, this is a no-op;
   otherwise, it performs a 4-byte swap operation.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 935)

Unknown directive type "function".

```
.. function:: ntohs(x)

   Convert 16-bit positive integers from network to host byte order.  On machines
   where the host byte order is the same as network byte order, this is a no-op;
   otherwise, it performs a 2-byte swap operation.

   .. versionchanged:: 3.10
```

```
      Raises :exc:`OverflowError` if *x* does not fit in a 16-bit unsigned
      integer.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 946)

Unknown directive type "function".

```
.. function:: htonl(x)

   Convert 32-bit positive integers from host to network byte order.  On machines
   where the host byte order is the same as network byte order, this is a no-op;
   otherwise, it performs a 4-byte swap operation.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 953)

Unknown directive type "function".

```
.. function:: htons(x)

   Convert 16-bit positive integers from host to network byte order.  On machines
   where the host byte order is the same as network byte order, this is a no-op;
   otherwise, it performs a 2-byte swap operation.

   .. versionchanged:: 3.10
      Raises :exc:`OverflowError` if *x* does not fit in a 16-bit unsigned
      integer.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 964)

Unknown directive type "function".

```
.. function:: inet_aton(ip_string)

   Convert an IPv4 address from dotted-quad string format (for example,
   '123.45.67.89') to 32-bit packed binary format, as a bytes object four characters in
   length.  This is useful when conversing with a program that uses the standard C
   library and needs objects of type :c:type:`struct in_addr`, which is the C type
   for the 32-bit packed binary this function returns.

   :func:`inet_aton` also accepts strings with less than three dots; see the
   Unix manual page :manpage:`inet(3)` for details.

   If the IPv4 address string passed to this function is invalid,
   :exc:`OSError` will be raised. Note that exactly what is valid depends on
   the underlying C implementation of :c:func:`inet_aton`.

   :func:`inet_aton` does not support IPv6, and :func:`inet_pton` should be used
   instead for IPv4/v6 dual stack support.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 983)

Unknown directive type "function".

```
.. function:: inet_ntoa(packed_ip)

   Convert a 32-bit packed IPv4 address (a :term:`bytes-like object` four
   bytes in length) to its standard dotted-quad string representation (for example,
   '123.45.67.89').  This is useful when conversing with a program that uses the
   standard C library and needs objects of type :c:type:`struct in_addr`, which
   is the C type for the 32-bit packed binary data this function takes as an
   argument.

   If the byte sequence passed to this function is not exactly 4 bytes in
   length, :exc:`OSError` will be raised. :func:`inet_ntoa` does not
   support IPv6, and :func:`inet_ntop` should be used instead for IPv4/v6 dual
   stack support.

   .. versionchanged:: 3.5
      Writable :term:`bytes-like object` is now accepted.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1001)

Unknown directive type "function".

```
.. function:: inet_pton(address_family, ip_string)

   Convert an IP address from its family-specific string format to a packed,
   binary format. :func:`inet_pton` is useful when a library or network protocol
   calls for an object of type :c:type:`struct in_addr` (similar to
   :func:`inet_aton`) or :c:type:`struct in6_addr`.

   Supported values for *address_family* are currently :const:`AF_INET` and
   :const:`AF_INET6`. If the IP address string *ip_string* is invalid,
   :exc:`OSError` will be raised. Note that exactly what is valid depends on
   both the value of *address_family* and the underlying implementation of
   :c:func:`inet_pton`.

   .. availability:: Unix (maybe not all platforms), Windows.

   .. versionchanged:: 3.4
      Windows support added
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1020)

Unknown directive type "function".

```
.. function:: inet_ntop(address_family, packed_ip)

   Convert a packed IP address (a :term:`bytes-like object` of some number of
   bytes) to its standard, family-specific string representation (for
   example, ``'7.10.0.5'`` or ``'5aef:2b::8'``).
   :func:`inet_ntop` is useful when a library or network protocol returns an
   object of type :c:type:`struct in_addr` (similar to :func:`inet_ntoa`) or
   :c:type:`struct in6_addr`.

   Supported values for *address_family* are currently :const:`AF_INET` and
   :const:`AF_INET6`. If the bytes object *packed_ip* is not the correct
   length for the specified address family, :exc:`ValueError` will be raised.
   :exc:`OSError` is raised for errors from the call to :func:`inet_ntop`.

   .. availability:: Unix (maybe not all platforms), Windows.

   .. versionchanged:: 3.4
      Windows support added

   .. versionchanged:: 3.5
      Writable :term:`bytes-like object` is now accepted.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1049)

Unknown directive type "function".

```
.. function:: CMSG_LEN(length)

   Return the total length, without trailing padding, of an ancillary
   data item with associated data of the given *length*.  This value
   can often be used as the buffer size for :meth:`~socket.recvmsg` to
   receive a single item of ancillary data, but :rfc:`3542` requires
   portable applications to use :func:`CMSG_SPACE` and thus include
   space for padding, even when the item will be the last in the
   buffer.  Raises :exc:`OverflowError` if *length* is outside the
   permissible range of values.

   .. availability:: most Unix platforms, possibly others.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1065)

Unknown directive type "function".

```
.. function:: CMSG_SPACE(length)

   Return the buffer size needed for :meth:`~socket.recvmsg` to
   receive an ancillary data item with associated data of the given
   *length*, along with any trailing padding.  The buffer space needed
   to receive multiple items is the sum of the :func:`CMSG_SPACE`
   values for their associated data lengths.  Raises
```

```
   :exc:`OverflowError` if *length* is outside the permissible range
   of values.

   Note that some systems might support ancillary data without
   providing this function.  Also note that setting the buffer size
   using the results of this function may not precisely limit the
   amount of ancillary data that can be received, since additional
   data may be able to fit into the padding area.

   .. availability:: most Unix platforms, possibly others.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1086)

Unknown directive type "function".

```
.. function:: getdefaulttimeout()

   Return the default timeout in seconds (float) for new socket objects. A value
   of ``None`` indicates that new socket objects have no timeout. When the socket
   module is first imported, the default is ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1093)

Unknown directive type "function".

```
.. function:: setdefaulttimeout(timeout)

   Set the default timeout in seconds (float) for new socket objects.  When
   the socket module is first imported, the default is ``None``.  See
   :meth:`~socket.settimeout` for possible values and their respective
   meanings.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1101)

Unknown directive type "function".

```
.. function:: sethostname(name)

   Set the machine's hostname to *name*.  This will raise an
   :exc:`OSError` if you don't have enough rights.

   .. audit-event:: socket.sethostname name socket.sethostname

   .. availability:: Unix.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1113)

Unknown directive type "function".

```
.. function:: if_nameindex()

   Return a list of network interface information
   (index int, name string) tuples.
   :exc:`OSError` if the system call fails.

   .. availability:: Unix, Windows.

   .. versionadded:: 3.3

   .. versionchanged:: 3.8
      Windows support was added.

   .. note::

      On Windows network interfaces have different names in different contexts
      (all names are examples):

      * UUID: ``{FB605B73-AAC2-49A6-9A2F-25416AEA0573}``
      * name: ``ethernet_32770``
      * friendly name: ``vEthernet (nat)``
```

```
   * description: ``Hyper-V Virtual Ethernet Adapter``

   This function returns names of the second form from the list, ``ethernet_32770``
   in this example case.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1140)

Unknown directive type "function".

```
.. function:: if_nametoindex(if_name)

   Return a network interface index number corresponding to an
   interface name.
   :exc:`OSError` if no interface with the given name exists.

   .. availability:: Unix, Windows.

   .. versionadded:: 3.3

   .. versionchanged:: 3.8
      Windows support was added.

   .. seealso::
      "Interface name" is a name as documented in :func:`if_nameindex`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1157)

Unknown directive type "function".

```
.. function:: if_indextoname(if_index)

   Return a network interface name corresponding to an
   interface index number.
   :exc:`OSError` if no interface with the given index exists.

   .. availability:: Unix, Windows.

   .. versionadded:: 3.3

   .. versionchanged:: 3.8
      Windows support was added.

   .. seealso::
      "Interface name" is a name as documented in :func:`if_nameindex`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1174)

Unknown directive type "function".

```
.. function:: send_fds(sock, buffers, fds[, flags[, address]])

   Send the list of file descriptors *fds* over an :const:`AF_UNIX` socket *sock*.
   The *fds* parameter is a sequence of file descriptors.
   Consult :meth:`sendmsg` for the documentation of these parameters.

   .. availability:: Unix supporting :meth:`~socket.sendmsg` and :const:`SCM_RIGHTS` mechanism.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1185)

Unknown directive type "function".

```
.. function:: recv_fds(sock, bufsize, maxfds[, flags])

   Receive up to *maxfds* file descriptors from an :const:`AF_UNIX` socket *sock*.
   Return ``(msg, list(fds), flags, addr)``.
   Consult :meth:`recvmsg` for the documentation of these parameters.

   .. availability:: Unix supporting :meth:`~socket.recvmsg` and :const:`SCM_RIGHTS` mechanism.

   .. versionadded:: 3.9

   .. note::
```

```
   Any truncated integers at the end of the list of file descriptors.
```

Socket Objects

Socket objects have the following methods. Except for :meth:`~socket.makefile`, these correspond to Unix system calls applicable to sockets.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1205); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1209)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.2
   Support for the :term:`context manager` protocol was added.  Exiting the
   context manager is equivalent to calling :meth:`~socket.close`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1214)

Unknown directive type "method".

```
.. method:: socket.accept()

   Accept a connection. The socket must be bound to an address and listening for
   connections. The return value is a pair ``(conn, address)`` where *conn* is a
   *new* socket object usable to send and receive data on the connection, and
   *address* is the address bound to the socket on the other end of the connection.

   The newly created socket is :ref:`non-inheritable <fd_inheritance>`.

   .. versionchanged:: 3.4
      The socket is now non-inheritable.

   .. versionchanged:: 3.5
      If the system call is interrupted and the signal handler does not raise
      an exception, the method now retries the system call instead of raising
      an :exc:`InterruptedError` exception (see :pep:`475` for the rationale).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1232)

Unknown directive type "method".

```
.. method:: socket.bind(address)

   Bind the socket to *address*.  The socket must not already be bound. (The format
   of *address* depends on the address family --- see above.)

   .. audit-event:: socket.bind self,address socket.socket.bind
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1239)

Unknown directive type "method".

```
.. method:: socket.close()

   Mark the socket closed.  The underlying system resource (e.g. a file
   descriptor) is also closed when all file objects from :meth:`makefile()`
   are closed.  Once that happens, all future operations on the socket
   object will fail. The remote end will receive no more data (after
   queued data is flushed).

   Sockets are automatically closed when they are garbage-collected, but
   it is recommended to :meth:`close` them explicitly, or to use a
   :keyword:`with` statement around them.

   .. versionchanged:: 3.6
      :exc:`OSError` is now raised if an error occurs when the underlying
      :c:func:`close` call is made.
```

```
   .. note::

      :meth:`close()` releases the resource associated with a connection but
      does not necessarily close the connection immediately.  If you want
      to close the connection in a timely fashion, call :meth:`shutdown()`
      before :meth:`close()`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1263)

Unknown directive type "method".

```
   .. method:: socket.connect(address)

      Connect to a remote socket at *address*. (The format of *address* depends on the
      address family --- see above.)

      If the connection is interrupted by a signal, the method waits until the
      connection completes, or raise a :exc:`TimeoutError` on timeout, if the
      signal handler doesn't raise an exception and the socket is blocking or has
      a timeout. For non-blocking sockets, the method raises an
      :exc:`InterruptedError` exception if the connection is interrupted by a
      signal (or the exception raised by the signal handler).

      .. audit-event:: socket.connect self,address socket.socket.connect

      .. versionchanged:: 3.5
         The method now waits until the connection completes instead of raising an
         :exc:`InterruptedError` exception if the connection is interrupted by a
         signal, the signal handler doesn't raise an exception and the socket is
         blocking or has a timeout (see the :pep:`475` for the rationale).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1284)

Unknown directive type "method".

```
   .. method:: socket.connect_ex(address)

      Like ``connect(address)``, but return an error indicator instead of raising an
      exception for errors returned by the C-level :c:func:`connect` call (other
      problems, such as "host not found," can still raise exceptions).  The error
      indicator is ``0`` if the operation succeeded, otherwise the value of the
      :c:data:`errno` variable.  This is useful to support, for example, asynchronous
      connects.

      .. audit-event:: socket.connect self,address socket.socket.connect_ex
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1295)

Unknown directive type "method".

```
   .. method:: socket.detach()

      Put the socket object into closed state without actually closing the
      underlying file descriptor.  The file descriptor is returned, and can
      be reused for other purposes.

      .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1304)

Unknown directive type "method".

```
   .. method:: socket.dup()

      Duplicate the socket.

      The newly created socket is :ref:`non-inheritable <fd_inheritance>`.

      .. versionchanged:: 3.4
         The socket is now non-inheritable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1314)

Unknown directive type "method".

```
.. method:: socket.fileno()

   Return the socket's file descriptor (a small integer), or -1 on failure. This
   is useful with :func:`select.select`.

   Under Windows the small integer returned by this method cannot be used where a
   file descriptor can be used (such as :func:`os.fdopen`).  Unix does not have
   this limitation.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1323)

Unknown directive type "method".

```
.. method:: socket.get_inheritable()

   Get the :ref:`inheritable flag <fd_inheritance>` of the socket's file
   descriptor or socket's handle: ``True`` if the socket can be inherited in
   child processes, ``False`` if it cannot.

   .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1332)

Unknown directive type "method".

```
.. method:: socket.getpeername()

   Return the remote address to which the socket is connected.  This is useful to
   find out the port number of a remote IPv4/v6 socket, for instance. (The format
   of the address returned depends on the address family --- see above.)  On some
   systems this function is not supported.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1340)

Unknown directive type "method".

```
.. method:: socket.getsockname()

   Return the socket's own address.  This is useful to find out the port number of
   an IPv4/v6 socket, for instance. (The format of the address returned depends on
   the address family --- see above.)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1347)

Unknown directive type "method".

```
.. method:: socket.getsockopt(level, optname[, buflen])

   Return the value of the given socket option (see the Unix man page
   :manpage:`getsockopt(2)`).  The needed symbolic constants (:const:`SO_\*` etc.)
   are defined in this module.  If *buflen* is absent, an integer option is assumed
   and its integer value is returned by the function.  If *buflen* is present, it
   specifies the maximum length of the buffer used to receive the option in, and
   this buffer is returned as a bytes object.  It is up to the caller to decode the
   contents of the buffer (see the optional built-in module :mod:`struct` for a way
   to decode C structures encoded as byte strings).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1359)

Unknown directive type "method".

```
.. method:: socket.getblocking()

   Return ``True`` if socket is in blocking mode, ``False`` if in
   non-blocking.

   This is equivalent to checking ``socket.gettimeout() == 0``.
```

```
   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1369)

Unknown directive type "method".

```
.. method:: socket.gettimeout()

   Return the timeout in seconds (float) associated with socket operations,
   or ``None`` if no timeout is set.  This reflects the last call to
   :meth:`setblocking` or :meth:`settimeout`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1376)

Unknown directive type "method".

```
.. method:: socket.ioctl(control, option)

   :platform: Windows

   The :meth:`ioctl` method is a limited interface to the WSAIoctl system
   interface.  Please refer to the `Win32 documentation
   <https://msdn.microsoft.com/en-us/library/ms741621%28VS.85%29.aspx>`_ for more
   information.

   On other platforms, the generic :func:`fcntl.fcntl` and :func:`fcntl.ioctl`
   functions may be used; they accept a socket object as their first argument.

   Currently only the following control codes are supported:
   ``SIO_RCVALL``, ``SIO_KEEPALIVE_VALS``, and ``SIO_LOOPBACK_FAST_PATH``.

   .. versionchanged:: 3.6
      ``SIO_LOOPBACK_FAST_PATH`` was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1394)

Unknown directive type "method".

```
.. method:: socket.listen([backlog])

   Enable a server to accept connections.  If *backlog* is specified, it must
   be at least 0 (if it is lower, it is set to 0); it specifies the number of
   unaccepted connections that the system will allow before refusing new
   connections. If not specified, a default reasonable value is chosen.

   .. versionchanged:: 3.5
      The *backlog* parameter is now optional.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1404)

Unknown directive type "method".

```
.. method:: socket.makefile(mode='r', buffering=None, *, encoding=None, \
                            errors=None, newline=None)

   .. index:: single: I/O control; buffering

   Return a :term:`file object` associated with the socket.  The exact returned
   type depends on the arguments given to :meth:`makefile`.  These arguments are
   interpreted the same way as by the built-in :func:`open` function, except
   the only supported *mode* values are ``'r'`` (default), ``'w'`` and ``'b'``.

   The socket must be in blocking mode; it can have a timeout, but the file
   object's internal buffer may end up in an inconsistent state if a timeout
   occurs.

   Closing the file object returned by :meth:`makefile` won't close the
   original socket unless all other file objects have been closed and
   :meth:`socket.close` has been called on the socket object.

   .. note::

      On Windows, the file-like object created by :meth:`makefile` cannot be
      used where a file object with a file descriptor is expected, such as the
      stream arguments of :meth:`subprocess.Popen`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1429)

Unknown directive type "method".

```
.. method:: socket.recv(bufsize[, flags])

   Receive data from the socket.  The return value is a bytes object representing the
   data received.  The maximum amount of data to be received at once is specified
   by *bufsize*.  See the Unix manual page :manpage:`recv(2)` for the meaning of
   the optional argument *flags*; it defaults to zero.

   .. note::

      For best match with hardware and network realities, the value of  *bufsize*
      should be a relatively small power of 2, for example, 4096.

   .. versionchanged:: 3.5
      If the system call is interrupted and the signal handler does not raise
      an exception, the method now retries the system call instead of raising
      an :exc:`InterruptedError` exception (see :pep:`475` for the rationale).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1447)

Unknown directive type "method".

```
.. method:: socket.recvfrom(bufsize[, flags])

   Receive data from the socket.  The return value is a pair ``(bytes, address)``
   where *bytes* is a bytes object representing the data received and *address* is the
   address of the socket sending the data.  See the Unix manual page
   :manpage:`recv(2)` for the meaning of the optional argument *flags*; it defaults
   to zero. (The format of *address* depends on the address family --- see above.)

   .. versionchanged:: 3.5
      If the system call is interrupted and the signal handler does not raise
      an exception, the method now retries the system call instead of raising
      an :exc:`InterruptedError` exception (see :pep:`475` for the rationale).

   .. versionchanged:: 3.7
      For multicast IPv6 address, first item of *address* does not contain
      ``%scope_id`` part anymore. In order to get full IPv6 address use
      :func:`getnameinfo`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1465)

Unknown directive type "method".

```
.. method:: socket.recvmsg(bufsize[, ancbufsize[, flags]])

   Receive normal data (up to *bufsize* bytes) and ancillary data from
   the socket.  The *ancbufsize* argument sets the size in bytes of
   the internal buffer used to receive the ancillary data; it defaults
   to 0, meaning that no ancillary data will be received.  Appropriate
   buffer sizes for ancillary data can be calculated using
   :func:`CMSG_SPACE` or :func:`CMSG_LEN`, and items which do not fit
   into the buffer might be truncated or discarded.  The *flags*
   argument defaults to 0 and has the same meaning as for
   :meth:`recv`.

   The return value is a 4-tuple: ``(data, ancdata, msg_flags,
   address)``.  The *data* item is a :class:`bytes` object holding the
   non-ancillary data received.  The *ancdata* item is a list of zero
   or more tuples ``(cmsg_level, cmsg_type, cmsg_data)`` representing
   the ancillary data (control messages) received: *cmsg_level* and
   *cmsg_type* are integers specifying the protocol level and
   protocol-specific type respectively, and *cmsg_data* is a
   :class:`bytes` object holding the associated data.  The *msg_flags*
   item is the bitwise OR of various flags indicating conditions on
   the received message; see your system documentation for details.
   If the receiving socket is unconnected, *address* is the address of
   the sending socket, if available; otherwise, its value is
   unspecified.

   On some systems, :meth:`sendmsg` and :meth:`recvmsg` can be used to
   pass file descriptors between processes over an :const:`AF_UNIX`
   socket.  When this facility is used (it is often restricted to
   :const:`SOCK_STREAM` sockets), :meth:`recvmsg` will return, in its
   ancillary data, items of the form ``(socket.SOL_SOCKET,
```

```
socket.SCM_RIGHTS, fds)``, where *fds* is a :class:`bytes` object
representing the new file descriptors as a binary array of the
native C :c:type:`int` type.  If :meth:`recvmsg` raises an
exception after the system call returns, it will first attempt to
close any file descriptors received via this mechanism.

Some systems do not indicate the truncated length of ancillary data
items which have been only partially received.  If an item appears
to extend beyond the end of the buffer, :meth:`recvmsg` will issue
a :exc:`RuntimeWarning`, and will return the part of it which is
inside the buffer provided it has not been truncated before the
start of its associated data.

On systems which support the :const:`SCM_RIGHTS` mechanism, the
following function will receive up to *maxfds* file descriptors,
returning the message data and a list containing the descriptors
(while ignoring unexpected conditions such as unrelated control
messages being received).  See also :meth:`sendmsg`. ::

   import socket, array

   def recv_fds(sock, msglen, maxfds):
       fds = array.array("i")   # Array of ints
       msg, ancdata, flags, addr = sock.recvmsg(msglen, socket.CMSG_LEN(maxfds * fds.itemsize))
       for cmsg_level, cmsg_type, cmsg_data in ancdata:
           if cmsg_level == socket.SOL_SOCKET and cmsg_type == socket.SCM_RIGHTS:
               # Append data, ignoring any truncated integers at the end.
               fds.frombytes(cmsg_data[:len(cmsg_data) - (len(cmsg_data) % fds.itemsize)])
       return msg, list(fds)

.. availability:: most Unix platforms, possibly others.

.. versionadded:: 3.3

.. versionchanged:: 3.5
   If the system call is interrupted and the signal handler does not raise
   an exception, the method now retries the system call instead of raising
   an :exc:`InterruptedError` exception (see :pep:`475` for the rationale).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1536)

Unknown directive type "method".

```
.. method:: socket.recvmsg_into(buffers[, ancbufsize[, flags]])

   Receive normal data and ancillary data from the socket, behaving as
   :meth:`recvmsg` would, but scatter the non-ancillary data into a
   series of buffers instead of returning a new bytes object.  The
   *buffers* argument must be an iterable of objects that export
   writable buffers (e.g. :class:`bytearray` objects); these will be
   filled with successive chunks of the non-ancillary data until it
   has all been written or there are no more buffers.  The operating
   system may set a limit (:func:`~os.sysconf` value ``SC_IOV_MAX``)
   on the number of buffers that can be used.  The *ancbufsize* and
   *flags* arguments have the same meaning as for :meth:`recvmsg`.

   The return value is a 4-tuple: ``(nbytes, ancdata, msg_flags,
   address)``, where *nbytes* is the total number of bytes of
   non-ancillary data written into the buffers, and *ancdata*,
   *msg_flags* and *address* are the same as for :meth:`recvmsg`.

   Example::

      >>> import socket
      >>> s1, s2 = socket.socketpair()
      >>> b1 = bytearray(b'----')
      >>> b2 = bytearray(b'0123456789')
      >>> b3 = bytearray(b'--------------')
      >>> s1.send(b'Mary had a little lamb')
      22
      >>> s2.recvmsg_into([b1, memoryview(b2)[2:9], b3])
      (22, [], 0, None)
      >>> [b1, b2, b3]
      [bytearray(b'Mary'), bytearray(b'01 had a 9'), bytearray(b'little lamb---')]

   .. availability:: most Unix platforms, possibly others.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1573)

Unknown directive type "method".

```
.. method:: socket.recvfrom_into(buffer[, nbytes[, flags]])

   Receive data from the socket, writing it into *buffer* instead of creating a
   new bytestring.  The return value is a pair ``(nbytes, address)`` where *nbytes* is
   the number of bytes received and *address* is the address of the socket sending
   the data.  See the Unix manual page :manpage:`recv(2)` for the meaning of the
   optional argument *flags*; it defaults to zero.  (The format of *address*
   depends on the address family --- see above.)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1583)

Unknown directive type "method".

```
.. method:: socket.recv_into(buffer[, nbytes[, flags]])

   Receive up to *nbytes* bytes from the socket, storing the data into a buffer
   rather than creating a new bytestring.  If *nbytes* is not specified (or 0),
   receive up to the size available in the given buffer.  Returns the number of
   bytes received.  See the Unix manual page :manpage:`recv(2)` for the meaning
   of the optional argument *flags*; it defaults to zero.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1592)

Unknown directive type "method".

```
.. method:: socket.send(bytes[, flags])

   Send data to the socket.  The socket must be connected to a remote socket.  The
   optional *flags* argument has the same meaning as for :meth:`recv` above.
   Returns the number of bytes sent. Applications are responsible for checking that
   all data has been sent; if only some of the data was transmitted, the
   application needs to attempt delivery of the remaining data. For further
   information on this topic, consult the :ref:`socket-howto`.

   .. versionchanged:: 3.5
      If the system call is interrupted and the signal handler does not raise
      an exception, the method now retries the system call instead of raising
      an :exc:`InterruptedError` exception (see :pep:`475` for the rationale).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1607)

Unknown directive type "method".

```
.. method:: socket.sendall(bytes[, flags])

   Send data to the socket.  The socket must be connected to a remote socket.  The
   optional *flags* argument has the same meaning as for :meth:`recv` above.
   Unlike :meth:`send`, this method continues to send data from *bytes* until
   either all data has been sent or an error occurs.  ``None`` is returned on
   success.  On error, an exception is raised, and there is no way to determine how
   much data, if any, was successfully sent.

   .. versionchanged:: 3.5
      The socket timeout is no more reset each time data is sent successfully.
      The socket timeout is now the maximum total duration to send all data.

   .. versionchanged:: 3.5
      If the system call is interrupted and the signal handler does not raise
      an exception, the method now retries the system call instead of raising
      an :exc:`InterruptedError` exception (see :pep:`475` for the rationale).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1626)

Unknown directive type "method".

```
.. method:: socket.sendto(bytes, address)
            socket.sendto(bytes, flags, address)

   Send data to the socket.  The socket should not be connected to a remote socket,
   since the destination socket is specified by *address*.  The optional *flags*
   argument has the same meaning as for :meth:`recv` above.  Return the number of
   bytes sent. (The format of *address* depends on the address family --- see
   above.)
```

```
   .. audit-event:: socket.sendto self,address socket.socket.sendto

   .. versionchanged:: 3.5
      If the system call is interrupted and the signal handler does not raise
      an exception, the method now retries the system call instead of raising
      an :exc:`InterruptedError` exception (see :pep:`475` for the rationale).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1643)

Unknown directive type "method".

```
.. method:: socket.sendmsg(buffers[, ancdata[, flags[, address]]])

   Send normal and ancillary data to the socket, gathering the
   non-ancillary data from a series of buffers and concatenating it
   into a single message.  The *buffers* argument specifies the
   non-ancillary data as an iterable of
   :term:`bytes-like objects <bytes-like object>`
   (e.g. :class:`bytes` objects); the operating system may set a limit
   (:func:`~os.sysconf` value ``SC_IOV_MAX``) on the number of buffers
   that can be used.  The *ancdata* argument specifies the ancillary
   data (control messages) as an iterable of zero or more tuples
   ``(cmsg_level, cmsg_type, cmsg_data)``, where *cmsg_level* and
   *cmsg_type* are integers specifying the protocol level and
   protocol-specific type respectively, and *cmsg_data* is a
   bytes-like object holding the associated data.  Note that
   some systems (in particular, systems without :func:`CMSG_SPACE`)
   might support sending only one control message per call.  The
   *flags* argument defaults to 0 and has the same meaning as for
   :meth:`send`.  If *address* is supplied and not ``None``, it sets a
   destination address for the message.  The return value is the
   number of bytes of non-ancillary data sent.

   The following function sends the list of file descriptors *fds*
   over an :const:`AF_UNIX` socket, on systems which support the
   :const:`SCM_RIGHTS` mechanism.  See also :meth:`recvmsg`. ::

      import socket, array

      def send_fds(sock, msg, fds):
          return sock.sendmsg([msg], [(socket.SOL_SOCKET, socket.SCM_RIGHTS, array.array("i", fds))])

   .. availability:: most Unix platforms, possibly others.

   .. audit-event:: socket.sendmsg self,address socket.socket.sendmsg

   .. versionadded:: 3.3

   .. versionchanged:: 3.5
      If the system call is interrupted and the signal handler does not raise
      an exception, the method now retries the system call instead of raising
      an :exc:`InterruptedError` exception (see :pep:`475` for the rationale).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1685)

Unknown directive type "method".

```
.. method:: socket.sendmsg_afalg([msg], *, op[, iv[, assoclen[, flags]]])

   Specialized version of :meth:`~socket.sendmsg` for :const:`AF_ALG` socket.
   Set mode, IV, AEAD associated data length and flags for :const:`AF_ALG` socket.

   .. availability:: Linux >= 2.6.38.

   .. versionadded:: 3.6
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1694)

Unknown directive type "method".

```
.. method:: socket.sendfile(file, offset=0, count=None)

   Send a file until EOF is reached by using high-performance
   :mod:`os.sendfile` and return the total number of bytes which were sent.
   *file* must be a regular file object opened in binary mode. If
   :mod:`os.sendfile` is not available (e.g. Windows) or *file* is not a
   regular file :meth:`send` will be used instead. *offset* tells from where to
   start reading the file. If specified, *count* is the total number of bytes
   to transmit as opposed to sending the file until EOF is reached. File
   position is updated on return or also in case of error in which case
```

```
      :meth:`file.tell() <io.IOBase.tell>` can be used to figure out the number of
      bytes which were sent. The socket must be of :const:`SOCK_STREAM` type.
      Non-blocking sockets are not supported.

      .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1710)

Unknown directive type "method".

```
   .. method:: socket.set_inheritable(inheritable)

      Set the :ref:`inheritable flag <fd_inheritance>` of the socket's file
      descriptor or socket's handle.

      .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1718)

Unknown directive type "method".

```
   .. method:: socket.setblocking(flag)

      Set blocking or non-blocking mode of the socket: if *flag* is false, the
      socket is set to non-blocking, else to blocking mode.

      This method is a shorthand for certain :meth:`~socket.settimeout` calls:

      * ``sock.setblocking(True)`` is equivalent to ``sock.settimeout(None)``

      * ``sock.setblocking(False)`` is equivalent to ``sock.settimeout(0.0)``

      .. versionchanged:: 3.7
         The method no longer applies :const:`SOCK_NONBLOCK` flag on
         :attr:`socket.type`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1734)

Unknown directive type "method".

```
   .. method:: socket.settimeout(value)

      Set a timeout on blocking socket operations.  The *value* argument can be a
      nonnegative floating point number expressing seconds, or ``None``.
      If a non-zero value is given, subsequent socket operations will raise a
      :exc:`timeout` exception if the timeout period *value* has elapsed before
      the operation has completed.  If zero is given, the socket is put in
      non-blocking mode. If ``None`` is given, the socket is put in blocking mode.

      For further information, please consult the :ref:`notes on socket timeouts <socket-timeouts>`.

      .. versionchanged:: 3.7
         The method no longer toggles :const:`SOCK_NONBLOCK` flag on
         :attr:`socket.type`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1750)

Unknown directive type "method".

```
   .. method:: socket.setsockopt(level, optname, value: int)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1751)

Unknown directive type "method".

```
   .. method:: socket.setsockopt(level, optname, value: buffer)
      :noindex:
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1753)

Unknown directive type "method".

```
.. method:: socket.setsockopt(level, optname, None, optlen: int)
   :noindex:

   .. index:: module: struct

   Set the value of the given socket option (see the Unix manual page
   :manpage:`setsockopt(2)`).  The needed symbolic constants are defined in the
   :mod:`socket` module (:const:`SO_\*` etc.).  The value can be an integer,
   ``None`` or a :term:`bytes-like object` representing a buffer. In the later
   case it is up to the caller to ensure that the bytestring contains the
   proper bits (see the optional built-in module :mod:`struct` for a way to
   encode C structures as bytestrings). When *value* is set to ``None``,
   *optlen* argument is required. It's equivalent to call :c:func:`setsockopt` C
   function with ``optval=NULL`` and ``optlen=optlen``.


   .. versionchanged:: 3.5
      Writable :term:`bytes-like object` is now accepted.

   .. versionchanged:: 3.6
      setsockopt(level, optname, None, optlen: int) form added.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1776)

Unknown directive type "method".

```
.. method:: socket.shutdown(how)

   Shut down one or both halves of the connection.  If *how* is :const:`SHUT_RD`,
   further receives are disallowed.  If *how* is :const:`SHUT_WR`, further sends
   are disallowed.  If *how* is :const:`SHUT_RDWR`, further sends and receives are
   disallowed.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1784)

Unknown directive type "method".

```
.. method:: socket.share(process_id)

   Duplicate a socket and prepare it for sharing with a target process.  The
   target process must be provided with *process_id*.  The resulting bytes object
   can then be passed to the target process using some form of interprocess
   communication and the socket can be recreated there using :func:`fromshare`.
   Once this method has been called, it is safe to close the socket since
   the operating system has already duplicated it for the target process.

   .. availability:: Windows.

   .. versionadded:: 3.3
```

Note that there are no methods :meth:`read` or :meth:`write`; use :meth:`~socket.recv` and :meth:`~socket.send` without *flags* argument instead.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1798); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1798); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1798); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1798); *backlink*

Unknown interpreted text role "meth".

Socket objects also have these (read-only) attributes that correspond to the values given to the :class:`~socket.socket` constructor.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1801); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1805)

Unknown directive type "attribute".

```
.. attribute:: socket.family

   The socket family.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1810)

Unknown directive type "attribute".

```
.. attribute:: socket.type

   The socket type.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1815)

Unknown directive type "attribute".

```
.. attribute:: socket.proto

   The socket protocol.
```

Notes on socket timeouts

A socket object can be in one of three modes: blocking, non-blocking, or timeout. Sockets are by default always created in blocking mode, but this can be changed by calling :func:`setdefaulttimeout`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1826); *backlink*

Unknown interpreted text role "func".

- In *blocking mode*, operations block until complete or the system returns an error (such as connection timed out).
- In *non-blocking mode*, operations fail (with an error that is unfortunately system-dependent) if they cannot be completed immediately: functions from the :mod:`select` can be used to know when and whether a socket is available for reading or writing.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1833); *backlink*

  Unknown interpreted text role "mod".

- In *timeout mode*, operations fail if they cannot be completed within the timeout specified for the socket (they raise a :exc:`timeout` exception) or if the system returns an error.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 1838); *backlink*

  Unknown interpreted text role "exc".

Note

At the operating system level, sockets in *timeout mode* are internally set in non-blocking mode. Also, the blocking and timeout modes are shared between file descriptors and socket objects that refer to the same network endpoint. This implementation detail can have visible consequences if e.g. you decide to use the :meth:`~socket.fileno()` of a

socket.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1843); *backlink*

Unknown interpreted text role "meth".

Timeouts and the `connect` method

The :meth:`~socket.connect` operation is also subject to the timeout setting, and in general it is recommended to call :meth:`~socket.settimeout` before calling :meth:`~socket.connect` or pass a timeout parameter to :meth:`create_connection`. However, the system network stack may also return a connection timeout error of its own regardless of any Python socket timeout setting.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1852); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1852); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1852); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1852); *backlink*

Unknown interpreted text role "meth".

Timeouts and the `accept` method

If :func:`getdefaulttimeout` is not :const:`None`, sockets returned by the :meth:`~socket.accept` method inherit that timeout. Otherwise, the behaviour depends on settings of the listening socket:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1862); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1862); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1862); *backlink*

Unknown interpreted text role "meth".

- if the listening socket is in *blocking mode* or in *timeout mode*, the socket returned by :meth:`~socket.accept` is in *blocking mode*;

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1866); *backlink*

  Unknown interpreted text role "meth".

- if the listening socket is in *non-blocking mode*, whether the socket returned by :meth:`~socket.accept` is in blocking or non-blocking mode is operating system-dependent. If you want to ensure cross-platform behaviour, it is recommended you manually override this setting.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1869); *backlink*

Unknown interpreted text role "meth".

Example

Here are four minimal example programs using the TCP/IP protocol: a server that echoes all data that it receives back (servicing only one client), and a client using it. Note that a server must perform the sequence :func:`.socket`, :meth:`~socket.bind`, :meth:`~socket.listen`, :meth:`~socket.accept` (possibly repeating the :meth:`~socket.accept` to service more than one client), while a client only needs the sequence :func:`.socket`, :meth:`~socket.connect`. Also note that the server does not :meth:`~socket.sendall`/:meth:`~socket.recv` on the socket it is listening on but on the new socket returned by :meth:`~socket.accept`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst`, line 1880); *backlink*

Unknown interpreted text role "meth".

The first two examples support IPv4 only.

```
# Echo server program
import socket

HOST = ''                 # Symbolic name meaning all available interfaces
PORT = 50007              # Arbitrary non-privileged port
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen(1)
    conn, addr = s.accept()
    with conn:
```

```
        print('Connected by', addr)
        while True:
            data = conn.recv(1024)
            if not data: break
            conn.sendall(data)

# Echo client program
import socket

HOST = 'daring.cwi.nl'    # The remote host
PORT = 50007              # The same port as used by the server
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hello, world')
    data = s.recv(1024)
print('Received', repr(data))
```

The next two examples are identical to the above two, but support both IPv4 and IPv6. The server side will listen to the first address family available (it should listen to both instead). On most of IPv6-ready systems, IPv6 will take precedence and the server may not accept IPv4 traffic. The client side will try to connect to the all addresses returned as a result of the name resolution, and sends traffic to the first one connected successfully.

```
# Echo server program
import socket
import sys

HOST = None               # Symbolic name meaning all available interfaces
PORT = 50007              # Arbitrary non-privileged port
s = None
for res in socket.getaddrinfo(HOST, PORT, socket.AF_UNSPEC,
                              socket.SOCK_STREAM, 0, socket.AI_PASSIVE):
    af, socktype, proto, canonname, sa = res
    try:
        s = socket.socket(af, socktype, proto)
    except OSError as msg:
        s = None
        continue
    try:
        s.bind(sa)
        s.listen(1)
    except OSError as msg:
        s.close()
        s = None
        continue
    break
if s is None:
    print('could not open socket')
    sys.exit(1)
conn, addr = s.accept()
with conn:
    print('Connected by', addr)
    while True:
        data = conn.recv(1024)
        if not data: break
        conn.send(data)

# Echo client program
import socket
import sys

HOST = 'daring.cwi.nl'    # The remote host
PORT = 50007              # The same port as used by the server
s = None
for res in socket.getaddrinfo(HOST, PORT, socket.AF_UNSPEC, socket.SOCK_STREAM):
    af, socktype, proto, canonname, sa = res
    try:
        s = socket.socket(af, socktype, proto)
    except OSError as msg:
        s = None
        continue
    try:
        s.connect(sa)
    except OSError as msg:
        s.close()
        s = None
        continue
    break
if s is None:
    print('could not open socket')
    sys.exit(1)
with s:
    s.sendall(b'Hello, world')
    data = s.recv(1024)
print('Received', repr(data))
```

The next example shows how to write a very simple network sniffer with raw sockets on Windows. The example requires administrator privileges to modify the interface:

```
import socket
```

```
# the public network interface
HOST = socket.gethostbyname(socket.gethostname())

# create a raw socket and bind it to the public interface
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_IP)
s.bind((HOST, 0))

# Include IP headers
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

# receive all packages
s.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)

# receive a package
print(s.recvfrom(65565))

# disabled promiscuous mode
s.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF)
```

The next example shows how to use the socket interface to communicate to a CAN network using the raw socket protocol. To use CAN with the broadcast manager protocol instead, open a socket with:

```
socket.socket(socket.AF_CAN, socket.SOCK_DGRAM, socket.CAN_BCM)
```

After binding (:const:`CAN_RAW`) or connecting (:const:`CAN_BCM`) the socket, you can use the :meth:`socket.send`, and the :meth:`socket.recv` operations (and their counterparts) on the socket object as usual.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 2024); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 2024); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 2024); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 2024); *backlink*

Unknown interpreted text role "meth".

This last example might require special privileges:

```
import socket
import struct


# CAN frame packing/unpacking (see 'struct can_frame' in <linux/can.h>)

can_frame_fmt = "=IB3x8s"
can_frame_size = struct.calcsize(can_frame_fmt)

def build_can_frame(can_id, data):
    can_dlc = len(data)
    data = data.ljust(8, b'\x00')
    return struct.pack(can_frame_fmt, can_id, can_dlc, data)

def dissect_can_frame(frame):
    can_id, can_dlc, data = struct.unpack(can_frame_fmt, frame)
    return (can_id, can_dlc, data[:can_dlc])


# create a raw socket and bind it to the 'vcan0' interface
s = socket.socket(socket.AF_CAN, socket.SOCK_RAW, socket.CAN_RAW)
s.bind(('vcan0',))

while True:
    cf, addr = s.recvfrom(can_frame_size)

    print('Received: can_id=%x, can_dlc=%x, data=%s' % dissect_can_frame(cf))

    try:
        s.send(cf)
    except OSError:
        print('Error sending CAN frame')

    try:
        s.send(build_can_frame(0x01, b'\x01\x02\x03'))
```

```
        except OSError:
            print('Error sending CAN frame')
```

Running an example several times with too small delay between executions, could lead to this error:

```
OSError: [Errno 98] Address already in use
```

This is because the previous execution has left the socket in a `TIME_WAIT` state, and can't be immediately reused.

There is a :mod:`socket` flag to set, in order to prevent this, :data:`socket.SO_REUSEADDR`:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 2076); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 2076); *backlink***
>
> Unknown interpreted text role "data".

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind((HOST, PORT))
```

the :data:`SO_REUSEADDR` flag tells the kernel to reuse a local socket in `TIME_WAIT` state, without waiting for its natural timeout to expire.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 2083); *backlink***
>
> Unknown interpreted text role "data".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]socket.rst, line 2087)**
>
> Unknown directive type "seealso".
>
> ```
> .. seealso::
>
>    For an introduction to socket programming (in C), see the following papers:
>
>    - *An Introductory 4.3BSD Interprocess Communication Tutorial*, by Stuart Sechrest
>
>    - *An Advanced 4.3BSD Interprocess Communication Tutorial*, by Samuel J.  Leffler et
>      al,
>
>    both in the UNIX Programmer's Manual, Supplementary Documents 1 (sections
>    PS1:7 and PS1:8).  The platform-specific reference material for the various
>    socket-related system calls are also a valuable source of information on the
>    details of socket semantics.  For Unix, refer to the manual pages; for Windows,
>    see the WinSock (or Winsock 2) specification.  For IPv6-ready APIs, readers may
>    want to refer to :rfc:`3493` titled Basic Socket Interface Extensions for IPv6.
> ```