

Error Codes

The following is a list of possible error codes returned by the [WSAGetLastError](#) call, along with their extended explanations. Errors are listed in alphabetical order by error macro. Some error codes defined in WINSOCK2.H are not returned from any function - these have not been listed here.

WSAEACCES

(10013)

Permission denied.

An attempt was made to access a socket in a way forbidden by its access permissions. An example is using a broadcast address for **sendto** without broadcast permission being set using **setsockopt(SO_BROADCAST)**.

WSAEADDRINUSE

(10048)

Address already in use.

Only one usage of each socket address (protocol/IP address/port) is normally permitted. This error occurs if an application attempts to **bind** a socket to an IP address/port that has already been used for an existing socket, or a socket that wasn't closed properly, or one that is still in the process of closing. For server applications that need to **bind** multiple sockets to the same port number, consider using **setsockopt(SO_REUSEADDR)**. Client applications usually need not call **bind** at all - **connect** will choose an unused port automatically.

WSAEADDRNOTAVAIL

(10049)

Cannot assign requested address.

The requested address is not valid in its context. Normally results from an attempt to **bind** to an address that is not valid for the local machine, or **connect/sendto** an address or port that is not valid for a remote machine (e.g. port 0).

WSAEAFNOSUPPORT

(10047)

Address family not supported by protocol family.

An address incompatible with the requested protocol was used. All sockets are created with an associated "address family" (i.e. AF_INET for Internet Protocols) and a generic protocol type (i.e. SOCK_STREAM). This error will be returned if an incorrect protocol is explicitly requested in the **socket** call, or if an address of the wrong family is used for a socket, e.g. in **sendto**.

WSAEALREADY

(10037)

Operation already in progress.

An operation was attempted on a non-blocking socket that already had an operation in progress - i.e. calling **connect** a second time on a non-blocking socket that is already connecting, or canceling an asynchronous request (WSAAsyncGetXbyY) that has already been canceled or completed.

WSAECONNABORTED

(10053)

Software caused connection abort.

An established connection was aborted by the software in your host machine, possibly due to a data transmission timeout or protocol error.

WSAECONNREFUSED

(10061)

Connection refused.

No connection could be made because the target machine actively refused it. This usually results from trying to connect to a service that is inactive on the foreign host - i.e. one with no server application running.

WSAECONNRESET

(10054)

Connection reset by peer.

A existing connection was forcibly closed by the remote host. This normally results if the peer application on the remote host is suddenly stopped, the host is rebooted, or the remote host used a "hard close" (see [setsockopt](#) for more information on the **SO_LINGER** option on the remote socket.)

WSAEDESTADDRREQ

(10039)

Destination address required.

A required address was omitted from an operation on a socket. For example, this error will be returned if [sendto](#) is called with the remote address of ADDR_ANY.

WSAEFAULT

(10014)

Bad address.

The system detected an invalid pointer address in attempting to use a pointer argument of a call. This error occurs if an application passes an invalid pointer value, or if the length of the buffer is too small. For instance, if the length of an argument which is a struct sockaddr is smaller than sizeof(struct sockaddr).

WSAEHOSTDOWN

(10064)

Host is down.

A socket operation failed because the destination host was down. A socket operation encountered a dead host. Networking activity on the local host has not been initiated. These conditions are more likely to be indicated by the error WSAETIMEDOUT.

WSAEHOSTUNREACH

(10065)

No route to host.

A socket operation was attempted to an unreachable host. See WSAENETUNREACH

WSAEINPROGRESS

(10036)

Operation now in progress.

A blocking operation is currently executing. Windows Sockets only allows a single blocking operation to be outstanding per task (or thread), and if any other function call is made (whether or not it references that or any other socket) the function fails with the WSAEINPROGRESS error.

WSAEINTR

(10004)

Interrupted function call.

A blocking operation was interrupted by a call to [WSACancelBlockingCall](#).

WSAEINVAL

(10022)

Invalid argument.

Some invalid argument was supplied (for example, specifying an invalid level to the [setsockopt](#) function). In some instances, it also refers to the current state of the socket - for instance, calling [accept](#) on a socket that is not **listening**.

WSAEISCONN

(10056)

Socket is already connected.

A connect request was made on an already connected socket. Some implementations also return this error if [sendto](#) is called on a connected SOCK_DGRAM socket (For SOCK_STREAM sockets, the *to* parameter in [sendto](#) is ignored), although other implementations treat this as a legal occurrence.

WSAEMFILE**(10024)**

Too many open files.

Too many open sockets. Each implementation may have a maximum number of socket handles available, either globally, per process or per thread.

WSAEMSGSIZE**(10040)**

Message too long.

A message sent on a datagram socket was larger than the internal message buffer or some other network limit, or the buffer used to receive a datagram into was smaller than the datagram itself.

WSAENETDOWN**(10050)**

Network is down.

A socket operation encountered a dead network. This could indicate a serious failure of the network system (i.e. the protocol stack that the WinSock DLL runs over), the network interface, or the local network itself.

WSAENETRESET**(10052)**

Network dropped connection on reset.

The host you were connected to crashed and rebooted. May also be returned by [setsockopt](#) if an attempt is made to set SO_KEEPALIVE on a connection that has already failed.

WSAENETUNREACH**(10051)**

Network is unreachable.

A socket operation was attempted to an unreachable network. This usually means the local software knows no route to reach the remote host.

WSAENOBUFS**(10055)**

No buffer space available.

An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full.

WSAENOPROTOPT**(10042)**

Bad protocol option.

An unknown, invalid or unsupported option or level was specified in a [getsockopt](#) or [setsockopt](#) call.

WSAENOTCONN**(10057)**

Socket is not connected.

A request to send or receive data was disallowed because the socket is not connected and (when sending on a datagram socket using [sendto](#)) no address was supplied. Any other type of operation might also return this error - for example, [setsockopt](#) setting SO_KEEPALIVE if the connection has been reset.

WSAENOTSOCK
(10038)

Socket operation on non-socket.

An operation was attempted on something that is not a socket. Either the socket handle parameter did not reference a valid socket, or for [select](#), a member of an fd_set was not valid.

WSAEOPNOTSUPP
(10045)

Operation not supported.

The attempted operation is not supported for the type of object referenced. Usually this occurs when a socket descriptor to a socket that cannot support this operation, for example, trying to accept a connection on a datagram socket.

WSAEPFNOSUPPORT
(10046)

Protocol family not supported.

The protocol family has not been configured into the system or no implementation for it exists. Has a slightly different meaning to WSAEAFNOSUPPORT, but is interchangeable in most cases, and all Windows Sockets functions that return one of these specify WSAEAFNOSUPPORT.

WSAEPROCLIM
(10067)

Too many processes.

A Windows Sockets implementation may have a limit on the number of applications that may use it simultaneously. [WSAStartup](#) may fail with this error if the limit has been reached.

WSAEPROTONOSUPPORT
(10043)

Protocol not supported.

The requested protocol has not been configured into the system, or no implementation for it exists. For example, a [socket](#) call requests a SOCK_DGRAM socket, but specifies a stream protocol.

WSAEPROTOTYPE
(10041)

Protocol wrong type for socket.

A protocol was specified in the [socket](#) function call that does not support the semantics of the socket type requested. For example, the ARPA Internet UDP protocol cannot be specified with a socket type of SOCK_STREAM.

WSAESHUTDOWN
(10058)

Cannot send after socket shutdown.

A request to send or receive data was disallowed because the socket had already been shut down in that direction with a previous [shutdown](#) call. By calling [shutdown](#) a partial close of a socket is requested, which is a signal that sending or receiving or both has been discontinued.

WSAESOCKTNOSUPPORT
(10044)

Socket type not supported.

The support for the specified socket type does not exist in this address family. For example, the optional type SOCK_RAW might be selected in a [socket](#) call, and the implementation does not support SOCK_RAW sockets at all.

WSAETIMEDOUT
(10060)

Connection timed out.

A connection attempt failed because the connected party did not properly respond after a period of

time, or established connection failed because connected host has failed to respond.

WSAEWOULDBLOCK
(10035)

Resource temporarily unavailable.

This error is returned from operations on non-blocking sockets that cannot be completed immediately, for example [recv](#) when no data is queued to be read from the socket. It is a non-fatal error, and the operation should be retried later. It is normal for WSAEWOULDBLOCK to be reported as the result from calling [connect](#) on a non-blocking SOCK_STREAM socket, since some time must elapse for the connection to be established.

WSAHOST_NOT_FOUND
(11001)

Host not found.

No such host is known. The name is not an official hostname or alias, or it cannot be found in the database(s) being queried. This error may also be returned for protocol and service queries, and means the specified name could not be found in the relevant database.

WSA_INVALID_HANDLE
(OS dependent)

Specified event object handle is invalid.

An application attempts to use an event object, but the specified handle is not valid.

WSA_INVALID_PARAMETER
(OS dependent)

One or more parameters are invalid.

An application used a Windows Sockets function which directly maps to a Win32 function. The Win32 function is indicating a problem with one or more parameters.

WSA_INVALIDPROC
(OS dependent)

Invalid procedure table from service provider.

A service provider returned a bogus proc table to WS2_32.DLL. (Usually caused by one or more of the function pointers being NULL.)

WSA_INVALIDPROVIDER
(OS dependent)

Invalid service provider version number.

A service provider returned a version number other than 2.0.

WSA_IO_PENDING
(OS dependent)

Overlapped operations will complete later.

The application has initiated an overlapped operation which cannot be completed immediately. A completion indication will be given at a later time when the operation has been completed.

WSA_IO_INCOMPLETE
(OS dependent)

Overlapped I/O event object not in signaled state.

The application has tried to determine the status of an overlapped operation which is not yet completed. Applications that use [WSAWaitForMultipleEvents](#) in a polling mode to determine when an overlapped operation has completed will get this error code until the operation is complete.

WSA_NOT_ENOUGH_MEMORY
(OS dependent)

Insufficient memory available.

An application used a Windows Sockets function which directly maps to a Win32 function. The Win32 function is indicating a lack of required memory resources.

WSANOTINITIALISED
(10093)

Successful WSAStartup not yet performed.

Either the application hasn't called [WSAStartup](#) or [WSAStartup](#) failed. The application may be accessing a socket which the current active task does not own (i.e. trying to share a socket between tasks), or **WSACleanup** has been called too many times.

WSANO_DATA
(11004)

Valid name, no data record of requested type.

The requested name is valid and was found in the database, but it does not have the correct associated data being resolved for. The usual example for this is a hostname -> address translation attempt (using [gethostbyname](#) or [WSAAsyncGetHostByName](#)) which uses the DNS (Domain Name Server), and an MX record is returned but no A record - indicating the host itself exists, but is not directly reachable.

WSANO_RECOVERY
(11003)

This is a non-recoverable error.

This indicates some sort of non-recoverable error occurred during a database lookup. This may be because the database files (e.g. BSD-compatible HOSTS, SERVICES or PROTOCOLS files) could not be found, or a DNS request was returned by the server with a severe error.

WSAPROVIDERFAILEDINIT
(OS dependent)

Unable to initialize a service provider.

Either a service provider's DLL could not be loaded (**LoadLibrary** failed) or the provider's **WSPStartup/NSPStartup** function failed.

WSASYS CALLFAILURE
(OS dependent)

System call failure.

Returned when a system call that should never fail does. For example, if a call to **WaitForMultipleObjects** fails or one of the registry functions fails trying to manipulate the protocol/namespace catalogs.

WSASYSNOTREADY
(10091)

Network subsystem is unavailable.

This error is returned by [WSAStartup](#) if the Windows Sockets implementation cannot function at this time because the underlying system it uses to provide network services is currently unavailable. Users should check:

- ? that the appropriate Windows Sockets DLL file is in the current path,
- ? that they are not trying to use more than one Windows Sockets implementation simultaneously. If there is more than one WINSOCK DLL on your system, be sure the first one in the path is appropriate for the network subsystem currently loaded.
- ? the Windows Sockets implementation documentation to be sure all necessary components are currently installed and configured correctly.

WSATRY_AGAIN
(11002)

Non-authoritative host not found.

This is usually a temporary error during hostname resolution and means that the local server did not receive a response from an authoritative server. A retry at some time later may be successful.

WSAVERNOTSUPPORTED

(10092)

WINSOCK.DLL version out of range.

The current Windows Sockets implementation does not support the Windows Sockets specification version requested by the application. Check that no old Windows Sockets DLL files are being accessed.

WSAEDISCON

(10094)

Graceful shutdown in progress.

Returned by [recv](#), [WSARecv](#) to indicate the remote party has initiated a graceful shutdown sequence.

WSA_OPERATION_ABORTED

(OS dependent)

Overlapped operation aborted.

An overlapped operation was canceled due to the closure of the socket, or the execution of the SIO_FLUSH command in [WSAIoctl](#).