This function sets a socket option.

int setsockopt(

SOCKET s,

int level,

int optname,

const char FAR\* optval,

int optlen

);

*s*

[in] Descriptor identifying a socket.

*level*

[in] Level at which the option is defined; the supported levels include:

[IPPROTO\_IP](http://msdn.microsoft.com/en-us/library/aa916098.aspx)

[IPPROTO\_IPV6](http://msdn.microsoft.com/en-us/library/aa916864.aspx)

[IPPROTO\_TCP](http://msdn.microsoft.com/en-us/library/aa916716.aspx)

[IPPROTO\_UDP](http://msdn.microsoft.com/en-us/library/aa917461.aspx)

[SOL\_SOCKET](http://msdn.microsoft.com/en-us/library/aa916772.aspx)

[SOL\_IRLMP](http://msdn.microsoft.com/en-us/library/aa921099.aspx)

[IPPROTO\_RAW](http://msdn.microsoft.com/en-us/library/aa915661.aspx)

*optname*

[in] Socket option for which the value is to be set.

*optval*

[in] Pointer to the buffer in which the value for the requested option is supplied.

*optlen*

[in] Size of the *optval* buffer.

[Return Value](javascript:void(0))

If no error occurs, this function returns zero. If an error occurs, a value of SOCKET\_ERROR is returned, and a specific error code can be retrieved by calling [WSAGetLastError](http://msdn.microsoft.com/en-us/library/aa915624.aspx).

All SO\_\* socket options apply equally to IPv4 and IPv6 (except SO\_BROADCAST, since broadcast is not implemented in IPv6).

**SOL\_SOCKET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Option** | **get/set** | **Optval Type** | **Description** |
| PVD\_CONFIG | both | char [] | The default is implementation dependent. This option retrieves an opaque data structure object from the service provider associated with socket *s*. This object stores the configured settings of the service provider. The exact format of this data structure is service provider specific. |
| SO\_ACCEPTCONN | get | BOOL | The default is FALSE unless a **WSPListen** has been performed. This option indicates whether a socket is in listening mode. The socket listens through through [WSPListen](http://msdn.microsoft.com/en-us/library/aa916712.aspx). Valid for connection oriented protocols only. |
| SO\_BROADCAST | both | BOOL | The default is FALSE. This option allows transmission of broadcast messages on the socket. Valid only for protocols that support broadcasting (IPX, UDP/IPv4, and others). |
| SO\_CONDITIONAL\_ACCEPT | both | BOOL | By default, ATM sets this option to TRUE. This option indicates incoming connections will be accepted or rejected by the application and not the stack  Setting the SO\_CONDITIONAL\_ACCEPT socket option to TRUE delays the acknowledgment of a connection until after the [WSAAccept](http://msdn.microsoft.com/en-us/library/aa917467.aspx) condition function is called. If FALSE, the connection may be accepted before the condition function is called, but the connection will be disconnected if the condition function rejects the call. This option must be set before calling the [listen](http://msdn.microsoft.com/en-us/library/aa924101.aspx) function, otherwise WSAEINVAL is returned. SO\_CONDITIONAL\_ACCEPT is only supported for TCP and ATM.  TCP sets SO\_CONDITIONAL\_ACCEPT to FALSE by default, and therefore by default the connection will be accepted before [WSAAccept](http://msdn.microsoft.com/en-us/library/aa917467.aspx) is called. When set to TRUE, the conditional decision must be made within the TCP connection time-out. CF\_DEFER connections are still subject to the time-out. |
| SO\_DEBUG | both | BOOL | The default is FALSE. Windows Sockets service providers are encouraged, but not required, to supply output debug information if the SO\_DEBUG option is set by an application or a Windows Sockets SPI client. The mechanism for generating the debug information and the form it takes are beyond the scope of this document. |
| SO\_DONTLINGER | both | BOOL | The default is TRUE. This option indicates whether a linger value was set on a socket. If TRUE, the SO\_LINGER option is disabled. Valid for reliable, connection oriented protocols only. |
| SO\_DONTROUTE | both | BOOL | The default is FALSE. This option indicates that routing is disabled, and outgoing data should be sent on whatever interface the socket and bound to. Valid for message oriented protocols only. Microsoft providers silently ignore this option and always consult the routing table to find appropriate outgoing interface. |
| SO\_ERROR | get | int | The default is zero (0). This option returns and resets the per socket–based error code. This is different from the per thread based–error code that is handled by using the **WSAGetLastError** and [WSASetLastError](http://msdn.microsoft.com/en-us/library/aa920050.aspx) function calls. A successful call that uses the socket does not reset the socket-based error code returned by the SO\_ERROR option. |
| SO\_GROUP\_ID | get | NULL | Reserved. Do not use. This option is exclusive to **getsockopt** and the value should be NULL. |
| SO\_GROUP\_PRIORITY | get | int | Reserved. Do not use. The default is zero (0) |
| SO\_KEEPALIVE | both | BOOL | The default is FALSE. An application or the Windows Sockets SPI client can request that a TCP/IP service provider enable the use of keep-alive packets on TCP connections by turning on the SO\_KEEPALIVE socket option. A Windows Sockets provider need not support the use of keep-alive packets. If it does, the precise semantics are implementation-specific but should conform to section 4.2.3.6 of RFC 1122: *Requirements for Internet Hosts — Communication Layers*. If a connection is dropped as the result of keep-alive packets, the error code WSAENETRESET is returned to any calls in progress on the socket and any subsequent calls will fail with WSAENOTCONN. SO\_KEEPALIVE is not supported on ATM sockets, and requests to enable the use of keep-alive packets on an ATM socket results in an error being returned by the socket. |
| SO\_LINGER | both | [linger](http://msdn.microsoft.com/en-us/library/aa921077.aspx) structure | The default is 1 (ON). This option controls the action taken when unsent data is queued on a socket and a [closesocket](http://msdn.microsoft.com/en-us/library/aa917514.aspx) or [WSPCloseSocket](http://msdn.microsoft.com/en-us/library/aa916350.aspx) is performed. See [closesocket](http://msdn.microsoft.com/en-us/library/aa917514.aspx) or [WSPCloseSocket](http://msdn.microsoft.com/en-us/library/aa916350.aspx)for a description of the way in which the SO\_LINGER settings affect the semantics of **closesocket**.  The application or Windows Sockets SPI client gets the current behavior by retrieving a [linger](http://msdn.microsoft.com/en-us/library/aa921077.aspx) structure (pointed to by the *optval* parameter) with the **l\_onoff** and **l\_linger** members set appropriately. |
| SO\_MAX\_MSG\_SIZE | get | DWORD | The default is implementation dependent. This is a get-only socket option that indicates the maximum outbound (send) size of a message for message-oriented socket types (for example, SOCK\_DGRAM) as implemented by a particular service provider. It has no description for byte stream oriented sockets. There is no provision to find out the maximum inbound–message size. |
| SO\_OOBINLINE | both | BOOL | The default is FALSE. This option indicates OOB data should be returned in-line with regular data. Valid for connection oriented protocols which support out-of-band data. |
| SO\_PROTOCOL\_INFO | get | [WSAPROTOCOL\_INFO](http://msdn.microsoft.com/en-us/library/aa916786.aspx) structure | The default is Protocol dependent. This option provides protocol information for the protocol that is bound to this socket. Socket owners can use this option to determine the provider that created the socket. This option supplies the [WSAPROTOCOL\_INFO](http://msdn.microsoft.com/en-us/library/aa916786.aspx) structure associated with this socket. See [WSAEnumProtocols](http://msdn.microsoft.com/en-us/library/aa916326.aspx) for more information about this structure. |
| SO\_PROTOCOL\_INFOW | get | WSAPROTOCOL\_INFOW | Supplies the [WSAPROTOCOL\_INFO](http://msdn.microsoft.com/en-us/library/aa916786.aspx) structure associated with this socket. See [WSCEnumProtocols](http://msdn.microsoft.com/en-us/library/aa922629.aspx) for more information about this structure. |
| SO\_RCVBUF | both | int | The default is implementation dependent. The Windows Embedded CE TCP/UDP provider uses 32768 bytes as default. The receive buffer can be set to a maximum of 1 MB.  This option specifies the total per-socket buffer space reserved for receives.  When a Windows Sockets implementation supports the SO\_RCVBUF and SO\_SNDBUF options, an application can request different buffer sizes (larger or smaller) by calling **setsockopt**. The call to **setsockopt** can succeed even when the implementation did not provide the whole amount requested. An application must call [getsockopt (Windows Sockets)](http://msdn.microsoft.com/en-us/library/aa916172.aspx) with the same option to check the buffer size actually provided. |
| SO\_REUSEADDR | both | BOOL | The default is FALSE. Allows the socket to be bound to an address that is already in use. (See the [bind](http://msdn.microsoft.com/en-us/library/aa926768.aspx) or [WSPBind](http://msdn.microsoft.com/en-us/library/aa916180.aspx) functions.)  By default, a socket cannot be bound to a local address that is already in use. On occasion, however, it can be necessary to reuse an address. Because every connection is uniquely identified by the combination of local and remote addresses, two sockets can be bound to the same local address as long as the remote addresses are different.  SO\_REUSEADDR is interpreted only at the time of the **bind** or **WSPBind**. To inform the Windows Sockets provider that a **bind** or **WSPBind** on a socket should not be disallowed because an address is already in use, the application or Windows Sockets SPI client should set SO\_REUSEADDR before issuing the **bind** or **WSPBind**.  REUSEADDR is not applicable for ATM sockets, and although requests to reuse and address do not result in an error, they have no affect on when an ATM socket is in use. |
| SO\_SNDBUF | both | int | The default is implementation dependent. This option sets the per-socket buffer size for sending data. When a Windows Sockets implementation supports the SO\_RCVBUF and SO\_SNDBUF options, an application or Windows Sockets SPI client can request different buffer sizes (larger or smaller). The call to [setsockopt](http://msdn.microsoft.com/en-us/library/aa926453.aspx) or [WSPSetSockOpt](http://msdn.microsoft.com/en-us/library/aa917502.aspx) can succeed even if the implementation did not provide the whole amount requested.  This is unrelated to SO\_MAX\_MSG\_SIZE or the size of a TCP window. Care should be taken when setting the SO\_SNDBUF value as setting this value either too high or too low can have a negative effect on performance. An application must call this function with the same option to check the buffer size actually provided. A value of 0 is not supported. |
| SO\_TYPE | get | int | The default is the socket type that was created with [socket](http://msdn.microsoft.com/en-us/library/aa916640.aspx). This option returns the socket type for the given socket (e.g. SOCK\_STREAM, SOCK\_DGRAM, etc.) |

浅析：setsockopt（）

1. 如果在已经处于 ESTABLISHED状态下的socket(一般由端口号和标志符区分）调用closesocket（一般不会立即关闭而经历TIME\_WAIT的过程）后想继续重用该socket：  
BOOL bReuseaddr=TRUE;  
setsockopt(s，SOL\_SOCKET ，SO\_REUSEADDR，(const char\*)&bReuseaddr，sizeof(BOOL));

2. 如果要已经处于连接状态的soket在调用closesocket后强制关闭，不经历TIME\_WAIT的过程：  
BOOL bDontLinger = FALSE;   
setsockopt(s，SOL\_SOCKET，SO\_DONTLINGER，(const char\*)&bDontLinger，sizeof(BOOL));

3.在send()，recv()过程中有时由于网络状况等原因，发收不能预期进行，而设置收发时限：  
int nNetTimeout=1000;//1秒  
//发送时限  
setsockopt(socket，SOL\_S0CKET，SO\_SNDTIMEO，(char \*)&nNetTimeout，sizeof(int));  
//接收时限  
setsockopt(socket，SOL\_S0CKET，SO\_RCVTIMEO，(char \*)&nNetTimeout，sizeof(int));

4.在send()的时候，返回的是实际发送出去的字节(同步)或发送到socket缓冲区的字节(异步);系统默认的状态发送和接收一次 为8688字节(约为8.5K)；在实际的过程中发送数据和接收数据量比较大，可以设置socket缓冲区，而避免了send()，recv()不断的循 环收发：  
// 接收缓冲区  
int nRecvBuf=32\*1024;//设置为32K  
setsockopt(s，SOL\_SOCKET，SO\_RCVBUF，(const char\*)&nRecvBuf，sizeof(int));  
//发送缓冲区  
int nSendBuf=32\*1024;//设置为32K  
setsockopt(s，SOL\_SOCKET，SO\_SNDBUF，(const char\*)&nSendBuf，sizeof(int));

5. 如果在发送数据的时，希望不经历由系统缓冲区到socket缓冲区的拷贝而影响程序的性能：  
int nZero=0;  
setsockopt(socket，SOL\_S0CKET，SO\_SNDBUF，(char \*)&nZero，sizeof(nZero));

6.同上在recv()完成上述功能(默认情况是将socket缓冲区的内容拷贝到系统缓冲区)：  
int nZero=0;  
setsockopt(socket，SOL\_S0CKET，SO\_RCVBUF，(char \*)&nZero，sizeof(int));

7.一般在发送UDP数据报的时候，希望该socket发送的数据具有广播特性：  
BOOL bBroadcast=TRUE;   
setsockopt(s，SOL\_SOCKET，SO\_BROADCAST，(const char\*)&bBroadcast，sizeof(BOOL));

8.在client连接服务器过程中，如果处于非阻塞模式下的socket在connect()的过程中可以设置connect()延时，直到accpet()被呼叫(本函数设

置只有在非阻塞的过程中有显著的作用，在阻塞的函数调用中作用不大)  
BOOL bConditionalAccept=TRUE;  
setsockopt(s，SOL\_SOCKET，SO\_CONDITIONAL\_ACCEPT，(const char\*)&bConditionalAccept，sizeof(BOOL));

9.如果在发送数据的过程中(send()没有完成，还有数据没发送)而调用了closesocket()，以前我们一般采取的措施是"从容关闭"shutdown(s，

SD\_BOTH)，但是数据是肯定丢失了，如何设置让程序满足具体应用的要求(即让没发完的数据发送出去后在关闭socket)？  
struct linger {  
u\_short l\_onoff;  
u\_short l\_linger;  
};  
linger m\_sLinger;  
m\_sLinger.l\_onoff=1;//(在closesocket()调用，但是还有数据没发送完毕的时候容许逗留)  
// 如果m\_sLinger.l\_onoff=0;则功能和2.)作用相同;  
m\_sLinger.l\_linger=5;//(容许逗留的时间为5秒)  
setsockopt(s，SOL\_SOCKET，SO\_LINGER，(const char\*)&m\_sLinger，sizeof(linger));  
Note:1.在设置了逗留延时，用于一个非阻塞的socket是作用不大的，最好不用;2.如果想要程序不经历SO\_LINGER需要设置SO\_DONTLINGER，或

者设置l\_onoff=0；

10.还一个用的比较少的是在SDI或者是Dialog的程序中，可以记录socket的调试信息：  
(前不久做过这个函数的测试，调式信息可以保存，包括socket建立时候的参数，采用的  
具体协议，以及出错的代码都可以记录下来）  
BOOL bDebug=TRUE;  
setsockopt(s，SOL\_SOCKET，SO\_DEBUG，(const char\*)&bDebug，sizeof(BOOL));

11.附加：往往通过setsockopt()设置了缓冲区大小，但还不能满足数据的传输需求，我的习惯是自己写个处理网络缓冲的类，动态分配内存;

nRet = ioctlsocket(sockClient, FIONREAD, &ul);//查看RCVBUF有多少數據可以讀取

int lRes = GetRecBuf(sockClient); 設置RCVBUF的最大大小

nRet = ioctlsocket(sockClient, FIONREAD, &ul);//查看A緩E衝O區?有L多?少-字?節s數E據?可A以E被irecv

int lRes = GetRecBuf(sockClient);