TCP/IP raw sockets

Article • 01/18/2022

A raw socket is a type of socket that allows access to the underlying transport provider. This topic focuses only on raw sockets and the IPv4 and IPv6 protocols. This is because most other protocols with the exception of ATM do not support raw sockets. To use raw sockets, an application needs to have detailed information on the underlying protocol being used.

Winsock service providers for the IP protocol may support a socket *type* of **SOCK_RAW**. The Windows Sockets 2 provider for TCP/IP included on Windows supports this **SOCK_RAW** socket type.

There are two basic types of such raw sockets:

- The first type uses a known protocol type written in the IP header that is recognized by a Winsock service provider. An example of the first type of socket is a socket for the ICMP protocol (IP protocol type = 1) or the ICMPv6 protocol (IP procotol type = 58).
- The second type allows any protocol type to be specified. An example of the second type would be an experimental protocol that is not directly supported by the Winsock service provider such as the Stream Control Transmission Protocol (SCTP).

Determining if Raw Sockets are Supported

If a Winsock service provider supports **SOCK_RAW** sockets for the AF_INET or AF_INET6 address families, the socket type of **SOCK_RAW** should be included in the **WSAPROTOCOL_INFO** structure returned by **WSAEnumProtocols** function for one or more of the available transport providers.

The **iAddressFamily** member in the **WSAPROTOCOL_INFO** structure should specify AF_INET or AF_INET6 and the **iSocketType** member of the **WSAPROTOCOL_INFO** structure should specify **SOCK_RAW** for one of the transport providers.

The **iProtocol** member of the **WSAPROTOCOL_INFO** structure may be set to **IPROTO_IP**. The **iProtocol** member of the **WSAPROTOCOL_INFO** structure may also be set to zero if the service provider allows an application to use a **SOCK_RAW** socket type for other network protocols other than the Internet Protocol for the address family.

The other members in the **WSAPROTOCOL_INFO** structure indicate other properties of the protocol support for **SOCK_RAW** and indicate how a socket of **SOCK_RAW** should be treated. These other members of the **WSAPROTOCOL_INFO** for **SOCK_RAW** normally specify that the protocol is connectionless, message-oriented, supports broadcast/multicast (the XP1_CONNECTIONLESS, XP1_MESSAGE_ORIENTED, XP1_SUPPORT_BROADCAST, and XP1_SUPPORT_MULTIPOINT bits are set in the dwServiceFlags1 member), and can have a maximum message size of 65,467 bytes.

IPR2025-00265
SportsCastr Inc. EX-2020

On Windows XP and later, the *NetSh.exe* command can be used to determine if raw sockets are supported. The following command run from a CMD window will display data from the Winsock catalog on the console:

netsh winsock show catalog

The output will include a list that contains some of the data from the **WSAPROTOCOL_INFO** structures supported on the local computer. Search for the term RAW/IP or RAW/IPv6 in the Description field to find those protocols that support raw sockets.

Creating a Raw Socket

To create a socket of type **SOCK_RAW**, call the **socket** or **WSASocket** function with the *af* parameter (address family) set to AF_INET or AF_INET6, the *type* parameter set to **SOCK_RAW**, and the *protocol* parameter set to the protocol number required. The *protocol* parameter becomes the protocol value in the IP header (SCTP is 132, for example).

> ⓘ **Note**
>
> An application may not specify zero (0) as the *protocol* parameter for the socket, WSASocket, and WSPSocket functions if the *type* parameter is set to **SOCK_RAW**.

Raw sockets offer the capability to manipulate the underlying transport, so they can be used for malicious purposes that pose a security threat. Therefore, only members of the Administrators group can create sockets of type SOCK_RAW on Windows 2000 and later.

Send and Receive Operations

Once an application creates a socket of type **SOCK_RAW**, this socket may be used to send and receive data. All packets sent or received on a socket of type **SOCK_RAW** are treated as datagrams on an unconnected socket.

The following rules apply to the operations over **SOCK_RAW** sockets:

- The **sendto** or **WSASendTo** function is normally used to send data on a socket of type **SOCK_RAW**. The destination address can be any valid address in the socket's address family, including a broadcast or multicast address. To send to a broadcast address, an application must have used **setsockopt** with SO_BROADCAST enabled. Otherwise, **sendto** or **WSASendTo** will fail with the error code WSAEACCES. For IP, an application can send to any multicast address (without becoming a group member).

- When sending IPv4 data, an application has a choice on whether to specify the IPv4 header at the front of the outgoing datagram for the packet. If the **IP_HDRINCL** socket option is set to true for an IPv4 socket (address family of AF_INET), the application must supply the IPv4 header in the outgoing data for send operations. If this socket option is false (the default setting), then the IPv4 header should not be in included the outgoing data for send operations.

- When sending IPv6 data, an application has a choice on whether to specify the IPv6 header at the front of the outgoing datagram for the packet. If the **IPV6_HDRINCL** socket option is set to true for an IPv6 socket (address family of AF_INET6), the application must supply the IPv6 header in the outgoing data for send operations. The default setting for this option is false. If this socket option is false (the default setting), then the IPv6 header should not be included in the outgoing data for send operations. For IPv6, there should be no need to include the IPv6 header. If information is available using socket functions, then the IPv6 header should not be included to avoid compatibility problems in the future. These issues are discussed in RFC 3542 published by the IETF. Using the **IPV6_HDRINCL** socket option is not recommended and may be deprecated in future.

- The **recvfrom** or **WSARecvFrom** function is normally used to receive data on a socket of type **SOCK_RAW**. Both of these functions have an option to return the source IP address where the packet was sent from. The received data is a datagram from an unconnected socket.

- For IPv4 (address family of AF_INET), an application receives the IP header at the front of each received datagram regardless of the **IP_HDRINCL** socket option.

- For IPv6 (address family of AF_INET6), an application receives everything after the last IPv6 header in each received datagram regardless of the **IPV6_HDRINCL** socket option. The application does not receive any IPv6 headers using a raw socket.

- Received datagrams are copied into all **SOCK_RAW** sockets that satisfy the following conditions:
  - The protocol number specified in the *protocol* parameter when the socket was created should match the protocol number in the IP header of the received datagram.
  - If a local IP address is defined for the socket, it should correspond to the destination address as specified in the IP header of the received datagram. An application may specify the local IP address by calling the **bind** function. If no local IP address is specified for the socket, the datagrams are copied into the socket regardless of the destination IP address in the IP header of the received datagram.
  - If a foreign address is defined for the socket, it should correspond to the source address as specified in the IP header of the received datagram. An application may specify the foreign IP address by calling the **connect** or **WSAConnect** function. If no foreign IP address is specified for the socket, the datagrams are copied into the socket regardless of the source IP address in the IP header of the received datagram.

It is important to understand that some sockets of type **SOCK_RAW** may receive many unexpected datagrams. For example, a PING program may create a socket of type **SOCK_RAW** to send ICMP echo

requests and receive responses. While the application is expecting ICMP echo responses, all other ICMP messages (such as ICMP HOST_UNREACHABLE) may also be delivered to this application. Moreover, if several **SOCK_RAW** sockets are open on a computer at the same time, the same datagrams may be delivered to all the open sockets. An application must have a mechanism to recognize the datagrams of interest and to ignore all others. For a PING program, such a mechanism might include inspecting the received IP header for unique identifiers in the ICMP header (the application's process ID, for example).

> **Note**
>
> To use a socket of type **SOCK_RAW** requires administrative privileges. Users running Winsock applications that use raw sockets must be a member of the Administrators group on the local computer, otherwise raw socket calls will fail with an error code of WSAEACCES. On Windows Vista and later, access for raw sockets is enforced at socket creation. In earlier versions of Windows, access for raw sockets is enforced during other socket operations.

Common Uses of Raw Sockets

One common use of raw sockets are troubleshooting applications that need to examine IP packets and headers in detail. For example, a raw socket can be used with the SIO_RCVALL IOCTL to enable a socket to receive all IPv4 or IPv6 packets passing through a network interface. For more information, see the SIO_RCVALL reference.

Limitations on Raw Sockets

On Windows 7, Windows Vista, Windows XP with Service Pack 2 (SP2), and Windows XP with Service Pack 3 (SP3), the ability to send traffic over raw sockets has been restricted in several ways:

- TCP data cannot be sent over raw sockets.
- UDP datagrams with an invalid source address cannot be sent over raw sockets. The IP source address for any outgoing UDP datagram must exist on a network interface or the datagram is dropped. This change was made to limit the ability of malicious code to create distributed denial-of-service attacks and limits the ability to send spoofed packets (TCP/IP packets with a forged source IP address).
- A call to the bind function with a raw socket for the IPPROTO_TCP protocol is not allowed.

> **Note**

The bind function with a raw socket is allowed for other protocols (IPPROTO_IP, IPPROTO_UDP, or IPPROTO_SCTP, for example).

These above restrictions do not apply to Windows Server 2008 R2, Windows Server 2008 , Windows Server 2003, or to versions of the operating system earlier than Windows XP with SP2.

> **Note**
>
> The Microsoft implementation of TCP/IP on Windows is capable of opening a raw UDP or TCP socket based on the above restrictions. Other Winsock providers may not support the use of raw sockets.

There are further limitations for applications that use a socket of type **SOCK_RAW**. For example, all applications listening for a specific protocol will receive all packets received for this protocol. This may not be what is desired for multiple applications using a protocol. This is also not suitable for high-performance applications. To get around these issues, it may be required to write a Windows network protocol driver (device driver) for the specific network protocol. On Windows Vista and later, Winsock Kernel (WSK), a new transport-independent kernel mode Network Programming Interface can be used to write a network protocol driver. On Windows Server 2003 and earlier, a Transport Driver Interface (TDI) provider and a Winsock helper DLL can be written to support the network protocol. The network protocol would then be added to the Winsock catalog as a supported protocol. This allows multiple applications to open sockets for this specific protocol and the device driver can keep track of which socket receives specific packets and errors. For information on writing a network protocol provider, see the sections on WSK and TDI in the Windows Driver Kit (WDK).

Applications also need to be aware of the impact that firewall settings may have on sending and receiving packets using raw sockets.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A