Windows Sockets 2
TCP/IP Extensions

Jan 31, 1995

Draft

Introduction

This document describes the requirement of the TCP/IP Extension Group and lists proposed updates to the WinSock2 API and SPI.

Summary of proposed features

Interfaces list support ability to get list of supported interfaces and some characteristics for each interface.

Support for IP_TTL option

RFC 793/1122 OOB - allow to choose the type of OOB support.

multicast support - ability to send and receive multicast packets.

IP6 addresses support - ability to choose the type of IP address to use.

Disable UDP checksum - ability to turn of UDP checksum.

IP options - abilty to specify IP options.

RAW_ICMP/RAW_IP - support for two types of raw data: with and without IP header.

New features description and justification

Interfaces list support ability to get list of supported interfaces and some characteristics for each interface.

The list of supported interfaces should include at least the following parameters for each interface:

Interface ID (opaque value);
Flags (Up/Down, Multicast, Broadcast, etc.);
IP address;
subnet mask;
broadcast address;

Inspite of the fact that this requirement contradicts the 'General Referential Parameter' ("Windows Sockets is not SNMP'), it was decided to include interface list support into this draft. The R easons for this are:

there were multiple requests for support of 'gethostid','get my IP address', etc.

multicast support for a host with multiple interfaces requires to know the list of these interfaces.

RFC 793/1122 OOB allow to choose the type of OOB support.

Support a socket option which allows to set the desired type of OOB data handling.

Multicast support ability to send and receive multicast packets.

Multicast support should allow:

send/receive multicast packets;
join/leave multicast groups;
set TTL for multicast packets;
choose interface which will be used by a socket when sending multicast packets.

'IP Multicast extensions for 4.3 BSD' by Steve Deering should be used as a guideline for implementation.

IP6 addresses support ability to choose the type of IP address to use.

It should be possible to specify both the IPv4 and IPv6 type of IP addresses.

IPv6 addresses is treated as addresses of different family, i.e. IPv4 addresses belong to AF_INET adress family, while IPv6 addresses belong to AF_INET6 family. The new structure should be defined to describe IPv6 addresses:

```
struct sockaddr_in6{
        u_short         sin6_family;    // AF_INET6
        u_short         sin6_port;      // port number
        u_long          sin6_flowlabel; // IPv6 flowlabel
        u_long          sin6_addr[4];   // IPv6 address
};
```

Disable UDP checksum ability to turn of UDP checksum.

Setting "UDP checksum off" causes UDP datagrams to be sent with a checksum of zero, and received UDP datagrams with a checksum of zero to be passed to the application. The default is "UDP checksum on". In this case the real checksum is calculated for the UDP datagrams to be sent, and UDP datagrams with checksum of zero are silently discarded.(See RFC 1122,section 4.1.3.4)

IP options abilty to specify IP options

IP security requires the access to the IP option part of the IP header. The format of the passed options should follow the BSD imlementation.

Support IP_TTL option Ability to overwrite the default value of TTL in IP header.

RAW_ICMP/RAW_IP support for two types of raw data: with and without IP header.

A user should be able to specify two types of 'raw' sockets. One type assumes that an IP header is created by the WinSock2 (or by the stack) for packets sent over the socket. Another type assumes that user must provide an IP header for each packet sent.

Proposed updates to WinSock2 API/SPI

These updates introduce several new options. Some of them are optional. If Service Provider doesn't support them, it should return WSAEINVAL on an attempt to get/set one of these options. If an application wants to know if any of these options is supported, it may open a socket and call 'getsockopt()' for the choosen option.

Interfaces list support

New command SIOGIFCONF should be added to the 'ioctl' function. This command returns the list of configured intefaces and their parameters. The support of this command is required for WinSock2 compliant service providers. Both API and SPI description of the 'ioctl' function should be updated.

The parameter *argp* points to the buffer which contains the information about interfaces. The description of the structure of this buffer follows:

Definition INTERFACE_LIST Structure:

DWORD ilLengthOfList - on input, the count of bytes in the buffer pointed by *argp*; on output, the count of bytes written into this buffer.

INTERFACE_INFO ilInterface[1] - array of structures each of which describes a single interface.

Definition INTERFACE_INFO Structure:

DWORD iiFlags - a bitmask describing the status of the interface. The following flags are possible:

IFF_UP	- interface is up
IFF_BROADCAST	- broadcast is supported
IFF_LOOPBACK	- this is loopback interface
IFF_POINTTOPOINT	- this is point-to-point link
IFF_MULTICAST	- multicat is supported

**sockaddr FAR ***iiAddress - address of the interface

**sockaddr FAR ***iiBroadcastAddress - broadcast address of the interface or the address of the other side for point-to-point links

**sockaddr FAR ***iiNetmask - netmask used by the interface

New IP options

The set of additional IP option requires the updates in the description of 'get/setsockopt' both in WinSock2 API and SPI documents. These update includes the new supported *level* IPPROTO_IP and the following new options:

IP_OPTIONS	- optional
IP_TOS	- optional
IP_TTL	- optional
IP_HDRINCL	- required for SOCK_RAW socket types.

The following options are required if protocol supports multicast , i.e. flag XP1_SUPPORTS_MULTICAST is set on output in **WSAEnumProtocols**():

IP_MULTICAST_IF
IP_MULTICAST_TTL
IP_MULTICAST_LOOP
IP_ADD_MEMBERSHIP
IP_DROP_MEMBERSHIP

Value	Type	Meaning
IP_OPTIONS	char FAR *	List of IP options to be inserted into outcoming packets. Setting the new options overwrites all the previously specified options. Setting *optval* to zero means removing of all the previously specified options.
IP_TOS	int	Specifies type of service to be used
IP_TTL	int	Specify TTL to be used
IP_HDRINCL	BOOL	If true, user should include IP header in the packets sent over SOCK_RAW interface, otherwise the header is provided by the protocol stack (service provider).
IP_MULTICAST_IF	struct in_addr FAR *	Select interface for outgoing multicast packets. The *optval* should point to the address of the interface to be used. If NULL, the default interface is used.
IP_MULTICAST_TTL	int	TTL used for the multicast packets
IP_MULTICAST_LOOP	BOOL	If true, multicast loopback is enabled, otherwise - disabled.
IP_ADD_MEMBERSHIP	struct ip_mreq FAR *	Specify the multicast group to join
IP_DROP_MEMBERSHIP	struct ip_mreq FAR *	Specify the multicast group to leave

```
struct ip_mreq  {
        struct in_addr imr_multiaddr; /* multicust group to join/drop */
        struct in_addr imr_interface; /* interface to join/drop on */
```

RFC 793/1122 OOB

New socket option requires the updates in the description of 'get/setsockopt' both in WinSock2 API and SPI documents. This option allows to choose between BSD and RFC-1122 style of expedited data. This option is not required.

Level	Value	Type	Meaning
SOL_SOCKET	SO_EXPEDITED_1122	BOOL	If set, the Service Provider implements the expedited data as specified in RFC-1222, otherwise the BSD stily is used. This option can be set on the connection only once, i.e. once on, this option can not be turned off.

Disable UDP checksum

The new UDP option requires the updates in the description of 'get/setsockopt' both in WinSock2 API and SPI documents.

Level	Value	Type	Meaning
IPPROTO_UDP	UDP_NOCHECKSUM	BOOL	If the option is set, UDP datagrams are sent with the checksum of zero and received UDP datagrams with checksum of zero are passed to application. This option is required. If a service provider does not have a mechanism to disable UDP checksum calculation, it may just store this option without doing any actions.

RAW_ICMP/RAW_IP

Service providers may support SOCK_RAW type of the socket. There are two types of such sockets: the first one assumes known protocol type as written in IP header, the second one allows to use any protocol number. The example of the first type of socket is ICMP, the example of the second type is an experimental protocol. The second type of protocols also allows an application to implement a protocol which is not supported by service provider.

The following updates are required to support SOCK_RAW type of sockets:

- WSAEnumProtocol() - *ipProtocol* field may be set to 0. This indicates that a caller may specify any value for the protocol parameter to the socket() API. If *ipProtocol* is set to 0, the *bMultiple* should be set to true.

- It should be stated that when the SOCK_RAW type of sockets for AF_INET family is used, it is assumed that

- when send() is called, the caller may or may not include IP header into the buffer passed to the send() depending on the IP_HDRINCL option.

- when receive() is called, the caller receives datagram which includes IP header regardless of the IP_HDRINCL option.

- received packets are copied into all raw socket that which satisfy the following conditions:

    - if the protocol number is specified for the socket, it should match the protocol number in the IP header of the received packet;

    - if a local address is defined for the socket, it should correspond to the destination address as specified in IP header of the received packet;

    - if a foreign address is defined for the socket, it should correspond to the source address as specified in IP header of the received packet;

Questions to discuss

1. Should we specify the required set of supported IP options ?