Appendix K

IPX/SPX Transport Provider

K.1 General

This appendix is a Preliminary Specification. It specifies protocol-specific information that is relevant for mapping XTI functions to SPX and IPX transport providers.

The description given here is limited to the IPX protocol and the enhanced SPX (or SPXII) protocol. All references to the SPX protocol refer to this version unless specifically noted. In compliance with the X/Open Interface Adoption Criteria, this protocol is obtainable from multiple sources.

Notes:

1. Neither IPX nor SPX supports expedited data. All data is handled on a first-come, first-served basis.
2. The protocol-specific data structures used by IPX and SPX (most notably, the **T_SPX2_OPTIONS** structure) are likely to grow in the future.

K.2 Namespace

K.2.1 IPX

If the header **xti_ipx.h** is included, identifiers with the prefixes, suffixes or complete names shown are reserved for any use by the implementation.

Header	Prefix
<xti_ipx.h>	t_ipx

If the header **xti_ipx.h** is included, macros with the following prefixes may be defined. After the last inclusion of **xti_ipx.h** an application may use identifiers with the following prefixes for its own purpose, provided their use is preceded by an **#undef** of the macro.

Header	Prefix
<xti_ipx.h>	T_IPX_

K.2.2 SPX

If the header **xti_spx.h** is included identifiers with the prefixes, suffixes or complete names shown are reserved for any use by the implementation.

Header	Prefix
<xti_spx.h>	t_spx, T_SPX

If the header **xti_spx.h** is included, macros with the prefixes may be defined. After the last inclusion of **xti_spx.h** an application may use identifiers with the following prefixes for its own purpose, provided there use is preceded by an **#undef** of the macro.

Header	Prefix
<xti_spx.h>	T_SPX_

K.3 Options

K.3.1 IPX-level Options

IPX options are association related. IPX options may be negotiated in all XTI states except T_UNBND and T_UNINIT. The level associated with each option is T_IPX_OPT. The name member should be set to T_IPX_OPTS_V1. IPX options are stored in a structure of type **t_ipxOptions** that follows the XTI **t_opthdr** structure containing at least the following members:

```
typedef struct t_ipxOptions {
    unsigned short    ipx_checksum       /* Checksum        */
    unsigned char     ipx_packet_type    /* IPX Packet type */
    } t_ipxOpts_t;
```

This option may only be manipulated using the *t_sndudata*( ) or *t_rcvudata*( ) functions.

Other options may be defined in the future.

At least the following packet types are known to IPX:

T_IPX_NULL_PACKET_TYPE Used for all packets not classified by any other type

T_IPX_NCP_PACKET_TYPE Used for NCP packets (that is, Netware Control Protocol)

T_IPX_SPX_PACKET_TYPE Sequenced packet protocol used for SPX packets

K.3.2 SPX-level Options

Some values of the SPX options are association-related as defined below. Unless otherwise noted, they may be negotiated in all XTI states except T_UNBND and T_UNINIT. The level associated with each option is T_SPX_OPT. The name member is set to T_SPX_OPTS_V1. SPX options are stored following the XTI **t_opthdr** structure in a format defined by the following data structure:

```
typedef struct t_spx2_options {
    unsigned int    versionNumber;
    unsigned int    spxIIOptionNegotiate;
    unsigned int    spxIIRetryCount;
    unsigned int    spxIIMinimumRetryDelay;
    unsigned int    spxIIMaximumRetryDelta;
    unsigned int    spxIIWatchdogTimeout;
    unsigned int    spxIIConnectionTimeout;
    unsigned int    spxIILocalWindowSize;
    unsigned int    spxIIRemoteWindowSize;
    unsigned int    spxIIConnectionID;
    unsigned int    spxIIInboundPacketSize;
    unsigned int    spxIIOutboundPacketSize;
    unsigned int    spxIISessionFlags;
    } T_SPX2_OPTIONS;
```

An application passing a **t_opthdr** structure followed by a **T_SPX2_OPTIONS** structure to a function that can support options information will get information about all legal options on each call.

Each of this structure's members is described below.

versionNumber
: The application must set this member to T_SPX_OPTIONS_VERSION. This member is used to manage forward compatibility as this structure is extended in the future. Access is through the *t_optmgmt*( ) or other functions that use the **t_call** data structure. T_SPX_OPTIONS_VERSION is currently set to 1.

spxIIOptionNegotiate
: This value is association-related. This member can be set to T_SPX_NEGOTIATE_OPTIONS (default) to indicate that an endpoint wishes to exchange option data with a remote endpoint, or to T_SPX_NO_NEGOTIATE_OPTIONS to indicate that it does not wish to do this. This is a negotiable value, and may be manipulated through the *t_optmgmt*( ) function or through functions that use the **t_call** data structure.

spxIIRetryCount
: This value specifies the number of unsuccessful transmission attempts that SPX will make before aborting a connection. Setting this value to 0 causes the default value (10) to be used. This is a negotiable value, and may be manipulated through the *t_optmgmt*( ) function or through functions that use the **t_call** data structure.

spxIIMinimumRetryDelay
: An internal round-trip time algorithm normally calculates the delay before a transmission retry. Setting this member to a non-zero value overrides this algorithm and uses the value as the fixed number of milliseconds before a retransmit. The default value is 300 milliseconds. This is a negotiable value, and may be manipulated through the *t_optmgmt*( ) function or through functions that use the **t_call** data structure.

spxIIMaximumRetryDelta

The value of this member is added to *spxIIMinimumRetryDelay* or to the current round-trip time to determine the maximum retry delay. Setting this value to 0 causes the default delay (5 seconds) to be used. This is a negotiable value, and may be manipulated through the *t_optmgmt*( ) function or through functions that use the **t_call** data structure.

spxIIWatchdogTimeout

This value determines the number of seconds that SPX will wait on an inactive connection before sending a still-alive query to the remote endpoint. This is a negotiable value, and may be manipulated through the *t_optmgmt*( ) function or through functions that use the **t_call** data structure.

spxIIConnectionTimeout

This value is the number of seconds that SPX will wait after a successful connect request before the first session packet must arrive. If a packet does not arrive in this interval, the connection is aborted. This is a negotiable value, and may be manipulated through the *t_optmgmt*( ) function or through functions that use the **t_call** data structure.

spxIILocalWindowSize

This value is association-related. This member sets the size of the local endpoint's receive window in packets. A zero setting requests the driver to negotiate the window size. The local driver default is 8. This is a negotiable value, and may be manipulated through the *t_optmgmt*( ) function or through functions that use the **t_call** data structure.

spxIIRemoteWindowSize

This value is association-related. This member is meaningful only after a connection has been established, that is, in states T_OUTCON, T_INCON, T_DATAXFER. It contains the number of packets in the remote endpoint's receive window. This is a non-negotiable value and is retrieved through functions that use the **t_call** data structure. Attempts to set this value with the T_NEGOTIATE flag set will have no effect.

spxIIConnectionID

This value is association-related. This member is meaningful only after a connection has been established, that is, in states T_OUTCON, T_INCON, T_DATAXFER. It contains the local endpoint connection ID. This a non-negotiable value and is retrieved through functions that use the **t_call** data structure. Attempts to set this value with the T_NEGOTIATE flag set will have no effect.

spxIIInboundPacketSize

This value is association-related. This member is meaningful only after a connection has been established, that is, in states T_OUTCON, T_INCON, T_DATAXFER. It contains the number of bytes in each incoming data packet. This value may change due to a route change in mid-connection. There is no way to notify an application that this has occurred. This is a non-negotiable value and is retrieved through functions that use the **t_call** data structure. Attempts to set this value with the T_NEGOTIATE flag set will have no effect.

spxIIOutboundPacketSize

This value is association-related. This member is meaningful only after a connection has been established, that is, in states T_OUTCON, T_INCON, T_DATAXFER. It contains the number of bytes in each outgoing data packet. This value may change due to a route change in mid-connection. There is no way to notify an application that this has occurred. This is a non-negotiable value and is retrieved through functions that use the **t_call** data structure. Attempts to set this value with the T_NEGOTIATE flag set will have no effect.

spxIISessionFlags

This value is association-related. This bit member contains flags that control physical layer characteristics of SPX packets. These may include checksums, data encryption, or data signing. The following flags have been defined:

Flag	Value	Description
T_SPX_SF_NONE	0x00	Options off
T_SPX_SF_IPX_CHECKSUM	0x01	Packet checksums on
T_SPX_SF_SPX2_SESSION	0x02	Compatibility flag

The T_SPX_SF_NONE and T_SPX_SF_IPX_CHECKSUM flags are read/write, and accessible through the *t_optmgmt*( ) function and functions using the **t_call** data structure. The T_SPX_SF_SPX2_SESSION flag is read-only and is accessible through functions that use the **t_call** data structure.

K.4 Functions

t_accept( ) No special considerations.

t_bind( ) **For IPX and SPX**:

IPX and SPX support static and dynamic port assignment. If the application does not wish to chose a port, it sets *req* to NULL or *req→addr.len* to 0. In this case, the provider assigns a dynamic port in the range 0x4000 to 0x7fff.

If the application wishes to bind to a specific port (that is, a static port), *req→addr.buf* must point to an addressing structure.

A process without sufficient privilege can only request a port in the range 0x8000 to 0xffff. If the request is for a port outside the static port range, *t_bind*( ) will fail and *t_errno* will be set to TACCESS. A process with appropriate privilege can request any port number.

For SPX:

SPX allows only a single transport endpoint to be bound to a port. If a requested port is already bound to another endpoint, *t_bind*( ) will fail and set *t_errno* to [TADDRBUSY].

t_connect( ) SPX does not support connect user data.

t_getinfo( ) The default characteristics returned by *t_getinfo*( ) in the **info** structure for an IPX endpoint are:

Parameters	Before call	After call
info→addr	/	x
info→options	/	x
info→tsdu	/	x (1)
info→etsdu	/	–2
info→connect	/	–2
info→discon	/	–2
info→servtype	/	T_CLTS
info→flags	/	0

(1) ''x'' is the smallest of the maximum size of transport data units supported by connected LANs.

The default characteristics returned by *t_getinfo*( ) in the **info** structure for an SPX endpoint are:

Parameters	Before call	After call
info->addr	/	x
info->options	/	x
info->tsdu	/	–1
info->etsdu	/	–2
info->connect	/	–2
info->discon	/	–2
info->servtype	/	T_COTS_ORD
info->flags	/	0

t_listen( ) SPX does not support connect user data.

An option buffer, as described in Section K.3.2, ''SPX-level Options'' will be passed to the user (unless *call→opt.maxlen* is zero.

t_open( ) The default characteristics returned by *t_open*( ) in the **info** structure for an IPX endpoint are the same as those listed in the table above for *t_getinfo*( ).

Similarly, the default characteristics returned by *t_open*( ) in the **info** structure for an SPX endpoint are the same as those listed in the table above for *t_getinfo*( ).

t_optmgmt( ) **For IPX**:

No IPX options are currently available through the *t_optmgmt*( ) function.

For SPX:

Options may be set or retrieved using the *t_optmgmt*( ) function only when the endpoint is in the state T_IDLE.

SPX supports the standard options buffer using the **t_opthdr** structure. The **t_opthdr** structure is followed by a **T_SPX2_OPTIONS** structure, as described in Section K.3, Options.

SPX supports options that may be examined and negotiated with the *t_optmgmt*( ) function. The options are listed and described in Section K.3, Options.

t_rcv( ) SPX does not support expedited data, so the T_EXPEDITED flag will not be set.

SPX allows logical units of data to be unlimited in length.

If the SPX watchdog (see Section K.3.2, ''SPX-level Options'' determines that the remote transport endoint is no longer participating in the connection, the SPX watchdog generates a disconnect indication which causes *t_rcv*( ) to return with a [T_LOOK] error.

t_rcvdis( ) SPX does not support disconnect user data.

On return, the *discon→reason* member is set to one of the following:

T_SPX_CONNECTION_TERMINATED
Indicates that no error occurred and an SPX terminate connection packet was received from the remote endpoint. This reason code indicates success.

T_SPX_CONNECTION_FAILED
Indicates that the remote endpoint failed to acknowledge a transmission.

t_rcvudata( ) The *t_rcvudata* function is issued on an IPX endpoint.

On return from the function, *unitdata→addr.buf* points to the address information for the remote endpoint, and *unitdata→opt.buf* points to an options buffer as described in Section K.3.1, ''IPX-level Options''.

If a packet received by a *t_rcvudata*( ) request did not have the checksum calculated and verified, the *t_ipx_checksum* member of the **struct t_ipxOptions**, is set to 0xFFFF. Any other value indicates that the provider has calculated and verified the checksum of the received packet. For all received packets, the *t_ipx_packet_type* member is set to the packet type of the received packet.

t_snd( ) SPX does not support expedited data so the T_EXPEDITED flag must not be set.

t_snddis( ) SPX does not support the sending of user data or options with a disconnect request.

t_sndudata( ) If *unitdata→opt.len* is zero, then a packet type of T_IPX_NULL_PACKET_TYPE is used and no checksum is generated.

To send any other packet type, *unitdata->opt* must reference an options buffer as defined in Section K.3.1. ''IPX-level Options''. The IPX packet type must be defined. A checksum will be generated if *t_ipx_checksum"* is set to T_IPX_CHECKSUM_TRIGGER.