Network Working Group Brian Bidulock
INTERNET-DRAFT The OpenSS7 Project

Expires in six months May 2001

Stream Control Transmission Protocol (SCTP)
Stream Control Transmission Provider Interface (SCTPI)
<draft-bidulock-sctpstreams-sigtran-00.txt>

Status of this Memo

Abstract

This Internet Draft provides a specification mapping the Stream Control Transmission Protocol RFC 2960[1] into a SVR 4.2 STREAMS provider interface. The benefit of this specification includes compatibility to UNIX International TLI (Transport Layer Interface)[2] and XOpen/XPG4 XTI (XOpen Transport Interface)[3] for maximum compatibility with OSI and IP transport provider applications based on SVR 4.2 STREAMS. In addition, this specification provides access to the features of SCTP which extend beyond those of existing transport level interfaces.

1. Introduction

2. Service Primitives

2.1. Transport Service Management Primitives

2.1.1. T_INFO_REQ − Get transport protocol parameter sizes

This primitive requests the transport provider to return the sizes of all relevant protocol parameters, plus the current state of the provider. This message consists of a M_PCPROTO mesage block formatted as follows:

```
struct T_info_req {
   t_scalar_t  PRIM_type;  /* always T_INFO_REQ */
};
```

Where *PRIM_type* indicates the primitive type. Note that the T_INFO_REQ and T_INFO_ACK primitive have no effect on the state of the transport provider and do not appear in the state tables.

This primitive requirest the transport provider to generate one of the following acknowledgements on receipt of the primitive and that the transport user wait for the acknowledgement before issuing any other primitives:

Successful
Acknolwegement of the primitive via the T_INFO_ACK.

Non-fatal errors
There are no errors associated with this primitive.

Transport Primitives	Stream Msg Type	IS 8072 Transport Primitives
T_CONN_REQ	M_PROTO	T-CONNECT request
T_CONN_IND	M_PROTO	T-CONNECT indication
T_CONN_RES	M_PROTO	T-CONNECT response
T_CONN_CON	M_PROTO	T-CONNECT confirm
T_DATA_REQ	M_PROTO	T-DATA request
T_DATA_IND	M_PROTO	T-DATA indication
T_EXDATA_REQ	M_PROTO	T-EXPEDITED-DATA request
T_EXDATA_IND	M_PROTO	T-EXPEDITED-DATA indication
T_DISCON_REQ	M_PROTO	T-DISCONNECT request
T_DISCON_IND	M_PROTO	T-DISCONNECT indication
T_UNITDATA_REQ	M_PROTO	T-UNITDATA request
T_UNITDATA_IND	M_PROTO	T-UNITDATA indication
T_ORDREL_REQ	M_PROTO	not defined in ISO
T_ORDREL_IND	M_PROTO	not defined in ISO
T_BIND_REQ	M_PROTO	not defined in ISO
T_BIND_ACK	M_PCPROTO	not defined in ISO
T_UNBIND_REQ	M_PROTO	not defined in ISO
T_OK_ACK	M_PCPROTO	not defined in ISO
T_ERROR_ACK	M_PCPROTO	not defined in ISO
T_INFO_REQ	M_PCPROTO	not defined in ISO
T_INFO_ACK	M_PCPROTO	not defined in ISO
T_UDERROR_IND	M_PROTO	not defined in ISO
T_OPTMGMT_REQ	M_PROTO	not defined in ISO
T_OPTMGMT_ACK	M_PCPROTO	not defined in ISO
T_ADDR_REQ	M_PROTO	not defined in ISO
T_ADDR_ACK	M_PCPROTO	not defined in ISO

Figure 1. Mapping of ISO IS8072 and RFC 2960 to Kernel-level Transport Service Primitives

2.1.2. T_INFO_ACK − Protocol information acknolwedgement

This primitive indicates to the transport user any relevan protocol-dependent paramters. It should be initiated in response to the T_INFO_REQ primitive described above. The format of this message is one M_PCPROTO message block formatted as follows:

```
struct T_info_ack {
   t_scalar_t  PRIM_type;      /* always T_INFO_ACK    */
   t_scalar_t  TSDU_size;      /* max TSDU size        */
   t_scalar_t  ETSDU_size;     /* max ETSDU size       */
   t_scalar_t  CDATA_size;     /* connect data size    */
   t_scalar_t  DDATA_size;     /* disconnect data size */
   t_scalar_t  ADDR_size;      /* TSAP size            */
   t_scalar_t  OPT_size;       /* Options size         */
   t_scalar_t  TIDU_size;      /* TIDU size            */
   t_scalar_t  SERV_type;      /* service type         */
   t_scalar_t  CURRENT_state;  /* current state        */
   t_scalar_t  PROVIDER_flag;  /* provider flag        */
};
```

Where the fields of this message have the following meanings:

PRIM_type

This indicates the primitive type and is always T_INFO_ACK.

TSDU_size

A value greater than zero specifies the maximum size of a transport service data unit (TSDU); a value of zero specifies that the transport provider does not suppor the concept of TSDU, although it does support the sending of a data stream with no logical boundaries preserved across a connection; a value of -1 specifies that there is no limit on the size of a TSDU; and a value of -2 specifies that the transfer of normal data is not supported by the transport provider.

Stream Control Transmission Protocol (SCTP) transport service providers must support the concept of a TSDU to provide the capabilites described in [1]. Therefore, SCTP transport providers should never return 0 or -2 in the *TSDU_size* parameter of the T_INFO_ACK. It is advised that SCTP providers which are capable of segmentation and reassembly of packets return -1 in this parameter.

ETSDU_size

A value greater that zero specified the maximum size of an expedited transport service data unit (ETSDU); a value of zero specifies that the transport provider does not support the concept of ETSDU, although it does support the sending of an expedited data stream with no logical boundaries preserved across a connection; a value of -1 specifies that there is no limit on the size of an ETSDU; and a value of -2 specifies that the transfer of expedited data is not supported by the transport provider.

SCTP as specified in RFC 2960 does not support the concept of expedited data directly; however, some SCTP service providers may support this concept in an implementation dependent fashion. It is advised, therefore, that SCTP transport provider return -1 if it supports expedited data, and -2 otherwise.

CDATA_size

A value greater that or equal to zero specifies the maximum amount of data that may be associated with the connection establishment primitives; and a value of -2 specifies that the transport provider does not allow data for be sent with connection establishment primitives.

DDATA_size

A value greater than or equal to zero specifies the maximum amount of data that may be associated with the disconnect primitives; and a value of -2 specifies that the transport provider does not allow data to be sent with the disconnect primitives.

ADDR_size

A value greater than or equal to zero indicates the maximum size of a transport protocol address; and a value of -2 specifies that the transport provider does not provide user access to transport protocol addresses.

OPT_size

A value greater than or equal to zero indicates the maximum number of bytes of protocol-specific options supported by the provider; and a value of -2 specifies that the transport provider does not support user-settable options.

SCTP providers have user-settable options which are mandated by RFC 2960[1] and the SCTP provider is advised to return a non-zero value for the *OPT_size* in the T_INFO_ACK.

TIDU_size

This is the amount of user data that may be presented in a single T_DATA_REQ or T_EXDATA_REQ primitive. This is the size of the transport protocol interface data unit, and should not exceed the tunable system limit, if non-zero, for the size of a STREAMS message.

This value is STREAMS specific and must be determined independent of the SCTP protocol.

SERV_type

This field specifies the service type supported by the transport provider, and is one of the following:

T_COTS The provider service is connection oriented with no orderly release support.

This service type may be returned by *streams* which are stream associated rather than association associated. That is, the stream head corresponds to an SCTP stream rather than an SCTP association. SCTP streams do not have orderly release.

T_COTS_ORD

The provider service is connection oriented with orderly release support.

This service type may be returned by *streams* which are association associated rather than stream associated. That is, the stream head corresponds to an SCTP association rather than an SCTP stream. SCTP associations do have orderly release.

T_CLTS The provider service is a connectionless transport service.

This service type may be returned by *streams* which are not associated with a given stream but which can access any stream in an association with T_UNITDATA_REQ and T_UNITDATA_IND messages.

CURRENT_state

This is the current state of the transport provider.

PROVIDER_flag

This field specifies additional properties specific to the transport provider and may alter the way the transport user communicates. The following flags may be set by the provider:

SENDZERO This flag indicates that the transport provider supports the sending of zero-length TSDUs.

XPG4_1 This flag indicates that the transport provider supports XPG4 semantics.

The following rules apply when the type of service is T_CLTS:

- The ETSDU_size, CDATA_size and DDATA_size files should be -2.
- The TSDU_size should equal the TIDU_size.

2.1.3. User-Originated Primitives

2.1.3.1. T_INFO_REQ - protocol sizes and capabilities

2.1.4. T_BIND_REQ - bind protocol address request

This primtive requests that the transport provider bind a protocol address to th *stream*, negotiate the number of connect indications allowed to be outstanding by the transport provider for the specified protocol address, and activate the *stream* associated with the protocol address. The format of the message is one M_PROTO message block. The format of the M_PROTO message block is as follows:

```
struct T_bind_req {
   t_scalar_t  PRIM_type;      /* always T_BIND_REQ */
   t_scalar_t  ADDR_length;    /* length of address */
   t_scalar_t  ADDR_offset;    /* offset of address */
   t_uscalar_t CONIND_number;  /* requested number of connect
                                  indications to be queued */
}
```

Where the fields of this message have the following meanings:

PRIM_type
This parameter indicates the primitive type.

ADDR_length
This field is the length of the protocol address to be bound to the *stream*.

SCTP permits binding to multiple local addresses. This can be accomplished using the T_BIND_REQ in two ways:

(1) One local destination transport address is specified for each call to T_BIND_REQ and multiple calls are made to establish multiple local destination transport addresses.

(2) Multiple local destination transport addresses are concatenated into a single *ADDR_length* and *ADDR_offset* parameter, with indicators in each of the destination transport addresses as to the transport address size and type.

The second method is the advised method for all SCTP transport providers because it is more consistent with the semantics of the socket library.

If the SCTP transport user wants the SCTP association to bind to all local destination transport addresses known to the SCTP provider, the *ADDR_length* parameter can be set to 0. In this case, the SCTP provider will attempt to bind to all local destination transport addresses which are known to the SCTP provider.

ADDR_offset
This field is the offset from the beginning of the M_PROTO block where the protocol address begins.

CONIND_number
This field is the requested number of connect indications allowed to be outstanding by the transport provider for the specified protocol address. The proper alignment of the address in the M_PROTO message block is not guaranteed. The address in the M_PROTO message block is, however, aligned the same as it was received from the transport user.

This primitive requires the transport provider to generate one of the following acknolwedgements upon receipt of the primitive, and the transport user must wait for the acknowledgement before issuing any other primitive:

− Successful
Correct acknowledgement of the primitive is indicated via the T_BIND_ACK primitive.

− Non-fatal errors
These errors will be indicated via the T_ERROR_ACK primitive (described in section detailing the T_ERROR_ACK primitive). The allowable errors are as follows:

TBADADDR This indicates that the protocol address was in an incorrect format or the address contained illegal information. It is not intended to indicate protocol errors.

TNOADDR This indicates that the transport provider could not allocate an address.

TACCES This indicates that the user did not have proper permissions for the use of the requested address.

TOUTSTATE The primitive would place the transport interface out of state.

TSYSERR A system error has occurred and the UNIX System error is indicated in the primitive.

TADDRBUSY
This indicates that the requested address is already in use.

2.1.5. T_BIND_ACK − Bind protocol address acknolwedgement

This primitive indicates to the transport user that the specified protocol address has been bound to the stream, that the specified number of connection indications are allowed to be queued by the transport provider for the specified protocol address, and that the stream associated with the specified protocol address has been activated. This message consists of one M_PCPROTO message block formatted as follows:

```
struct T_bind_ack {
   t_scalar_t  PRIM_type;     /* always T_BIND_ACK */
   t_scalar_t  ADDR_length;
   t_scalar_t  ADDR_offset;
   t_uscalar_t CONIND_number;
};
```

Where the parameters of the T_BIND_ACK message are as follows:

PRIM_type
This parameter indicates the primitive type and is always T_BIND_ACK.

ADDR_length
This parameter is the length of the protocol address(es) bound to the stream.

ADDR_offset
This parameter is the offset from the begninning of the M_PCPROTO block where the protocol address begins.

CONIND_number
This parameter is the accepted number of connect indications allowed to be outstanding by the transport provider for the specified protocol address.

Note that this filed does not apply to connectionless transport providers.

The proper alignment of the address in the M_PCPROTO message block is not guaranteed.

The following rules apply to the binding of the specified protocol address to the stream:

- If the ADDR_length field in the T_BIND_REQ primitive is 0, then the transport provider is to assign a transport protocol address to the user.
- The transport provider is to bind the transport address as specified in the T_BIND_REQ primitive.
- If the transport provider cannot bind the specified address(es) the provider will return TADDRBUSY.

3. Transport Provider States

This section provides a mapping of TLI (Transport Layer Interface) kernel states onto SCTP states.

3.1.1. TS_UNBND

T_BIND_REQ
When the user requests a bind with *T_BIND_REQ*, move to the **TS_WACK_BREQ** state.

T_PASS_CONN

3.1.2. TS_WACK_BREQ

T_BIND_ACK
If the user's *T_BIND_REQ* was valid, bind the stream and move to the **TS_IDLE** state.

TLI State	Usr State	SCTP State
	T_UNINIT	
TS_UNBND	T_UNBND	SCTP_CLOSED
TS_WACK_BREQ	T_UNBND	SCTP_CLOSED
TS_WACK_UREQ	T_IDLE	SCTP_CLOSED
TS_IDLE	T_IDLE	SCTP_CLOSED
TS_WACK_OPTREQ	T_IDLE	SCTP_CLOSED
TS_WACK_CREQ	T_OUTCON	SCTP_COOKIE_WAIT
TS_WCON_CREQ	T_OUTCON	SCTP_COOKIE_WAIT
TS_WRES_CIND	T_INCON	SCTP_COOKIE_ECHOED
TS_WACK_CRES	T_INCON	SCTP_COOKIE_ECHOED
TS_DATA_XFER	T_DATAXFER	SCTP_ESTABLISHED
TS_WIND_ORDREL	T_OUTREL	SCTP_SHUTDOWN_PENDING, SCTP_SHUTDOWN_SENT, SCTP_SHUTDOWN_ACK_SENT1
TS_WREQ_ORDREL	T_INREL	SCTP_SHUTDOWN_RECEIVED
TS_WACK_DREQ6	T_OUTCON	SCTP_COOKIE_WAIT
TS_WACK_DREQ7	T_INCON	SCTP_COOKIE_ECHOED
TS_WACK_DREQ9	T_DATAXFER	SCTP_ESTABLISHED
TS_WACK_DREQ10	T_OUTREL	SCTP_SHUTDOWN_PENDING, SCTP_SHUTDOWN_SENT, SCTP_SHUTDOWN_ACK_SENT1
TS_WACK_DREQ11	T_INREL	SCTP_SHUTDOWN_RECEIVED
TS_NOSTATES	T_FAKE	SCTP_UNREACHABLE

Figure 2. Mapping of TLI kernel states to RFC 2960 states.

T_ERROR_ACK
If the user's *T_BIND_REQ* was invalid, return a *T_ERROR_ACK* primitive to the user and return to the **TS_UNBND** state.

3.1.3. TS_WACK_UREQ

3.1.4. TS_IDLE

T_CONN_REQ
When the user request a connection with a *T_CONN_REQ*, move to the **TS_WACK_CREQ** state.

T_CONN_IND
When the peer requests a connection (receipt of a valid **COOKIE-ECHO** chunk), move to state **TS_WRES_CIND**.

3.1.5. TS_WACK_OPTREQ

3.1.6. TS_WACK_CREQ

T_ERROR_ACK
If the user *T_CONN_REQ* was invalid, return a *T_ERROR_ACK* primitive to the user and stay in the **TS_IDLE** state.

T_OK_ACK1
If the user *T_CONN_REQ* was valid, send a **INIT** chunk to the peer and return a *T_OK_ACK* primitive to the user and move to the **TS_WCON_CREQ** state.

3.1.7. TS_WCON_CREQ

T_DISCON_REQ

T_CONN_CON

T_DISCON_IND1

3.1.8. TS_WRES_CIND

In this state, the provider is awaiting a response to a connection indication. SCTP is in the **SCTP_CLOSED** state after having received a **COOKIE-ECHO** chunk.

T_CONN_RES
The user has decided to accept the connection. Send a **COOKIE-ACK** and move to the **SCTP_ESTABLISHED** state. The provider moves to the **TS_DATA_XFER** state.

T_DISCON_REQ
The user has decided not to accept the connection. Send an **ABORT** and move to the **SCTP_CLOSED** state. The provider moves to the **TS_IDLE** state.

T_CONN_IND

T_DISCON_IND2

3.1.9. TS_WACK_CRES

T_ERROR_ACK

T_OK_ACK2

T_OK_ACK3

T_OK_ACK4

3.1.10. TS_DATA_XFER

T_DISCON_REQ

T_DATA_REQ

T_EXDATA_REQ

T_ORDREL_REQ

T_DATA_IND

T_EXDATA_IND

T_ORDREL_IND

T_DISCON_IND1

3.1.11. TS_WIND_ORDREL

This is a stable state while waiting for a *T_ORDREL_IND* from the peer. This corresponds to a state where the local sender has shut down transmission of data and we are waiting for the peer endpoint to shut down transmission of data in repsonse to a **SHUTDOWN** chunk which was previously sent when entering this state. Data transmission in this state is not valid.

For SCTP, this corresponds to the SCTP **SHUTDOWN-SENT** state.

T_DISCON_REQ

T_DATA_IND

T_EXDATA_IND

T_ORDREL_IND

T_DISCON_IND1

3.1.12. TS_WREQ_ORDREL

This is a stable state while waiting for a *T_ORDREL_REQ* from the user. This corresponds to a state where the peer has shut down transmission of data and we are waiting for the local endpoint to shut down transmission of data in response to a *T_ORDREL_IND* which was previously delivered. Data transmission in this state is not valid.

For SCTP, this state corresponds to the SCTP **SHUTDOWN-RECEIVED** state.

T_DISCON_REQ

T_DATA_REQ

T_EXDATA_REQ

T_ORDREL_REQ

T_DISCON_IND1

3.1.13. TS_WACK_DREQ6

This is a transient state while waiting for acknowledgement of a *T_DISCON_REQ* issued while in state 6 (**TS_WCON_CREQ**).

T_ERROR_ACK

T_OK_ACK_1

3.1.14. TS_WACK_DREQ7

This is a transient state while waiting for acknowledgement of a *T_DISCON_REQ* issued while in state 7 (**TS_WRES_CIND**).

T_ERROR_ACK

T_OK_ACK2

T_OK_ACK3

T_OK_ACK4

3.1.15. TS_WACK_DREQ9

This is a transient state while waiting for acknowledgement of a *T_DISCON_REQ* issued while in state 9 (**TS_DATA_XFER**).

T_ERROR_ACK

T_OK_ACK_1

3.1.16. TS_WACK_DREQ10

This is a transient state while waiting for acknowledgement of a *T_DISCON_REQ* issued while in state 10 (**TS_WIND_ORDREL**).

T_ERROR_ACK

T_OK_ACK_1

3.1.17. TS_WACK_DREQ11

This is a transient state while waiting for acknowledgement of a *T_DISCON_REQ* issued while in state 11 (**TS_WREQ_ORDREL**).

T_ERROR_ACK

T_OK_ACK_1

Security Considerations

4. Acknowledgements

5. References

[1] Stream Control Transmission Protocol, RFC 2960 February 2000

[2] Transport Provider Interface Specification, Unix International - OSI Special Interest Group, Revision 1.5, December 10, 1992.

[3]

[4]

[5]

[6]

[7]

[8]

[9]

[10]

[11]

[12]

[13]

[14]

[15]

[16]

[17]

[18]

[19]

6. Author's Addresses

Brian F. G. Bidulock Tel: +1-972-839-4489
The OpenSS7 Project EMail: bidulock@openss7.org
4701 Preston Park Boulevard, Suite 424
Plano, TX 75093
USA

This Internet Draft expires September 2001.