Isode

SIS Layer Extension Protocol (SLEP) (S5066-APP3)

24th August 2020

Version: 1.13

Status: Experimental

1 Purpose

This document specifies a layer protocol that can be used over SIS to provide a set of core services that can be used by different applications. This protocol is designed as a replacement for RCOP and UDOP. It also provides compression and a new stream service. It enables a single application to use any combination of these services over a single SIS connection.

This document is published in the STANAG 5066 Application (S5066-APP) series.

2 SLEP Service Summary

SLEP provides the following services that can be used in combination by an application.

2.1 Extended Addressing

Extended addressing allows multiple applications to share a SAP, removing the limitation of 16 SAPs. Use of SAP only (slightly) reduces overhead, so key applications should be assigned a SAP without extended addressing. This option can be used with any SAP.

SLEP uses a single byte to give an extra 256 addresses for each SAP. An extended address of “0” is equivalent to no extended address.

The SIS extended address capability option (in another specification) allows applications to listen on a range of extended addresses. This is necessary to make use of extended addressing.

2.2 Unreliable Datagram Service

This service provides un-acknowledged datagram transmissions of blocks of data. This provides a service similar to UDP (RFC 768).

2.3 Reliable Datagram Service

This service provides acknowledged datagram transmissions of blocks of data.

2.4 Stream Service

SLEP provides a reliable bidirectional stream service, which may be used to transfer data in one direction only. The SLEP stream service can provide a service equivalent to TCP (RFC 793). This SLEP stream service can be used to enable TCP applications to operate over STANAG 5066.

2.5 Compression

Data **may** be compressed by the SLEP user using DEFLATE as specified in RFC 1951. This compression applies across the whole datagram or stream. Use of SLEP compression is optionally selected for the stream service. Datagrams are compressed if this leads to data reduction.

3 Improvements over RCOP/UDOP

SLEP provides the following benefits over RCOP and UDOP:

- Enables only required sub-services to be selected.
- More compact encoding
- Single protocol covering both ARQ and non-ARQ.
- Optional Compression.
- Acknowledgements for reliable datagram, to ensure reliability. This addresses RCOP data loss cases.
- Identifies total number of fragments in extended data (a significant design deficiency of RCOP/UDOP).
- Deals with Unidata Reject, hiding the rejects from the SLEP user.
- Addition of a stream service.

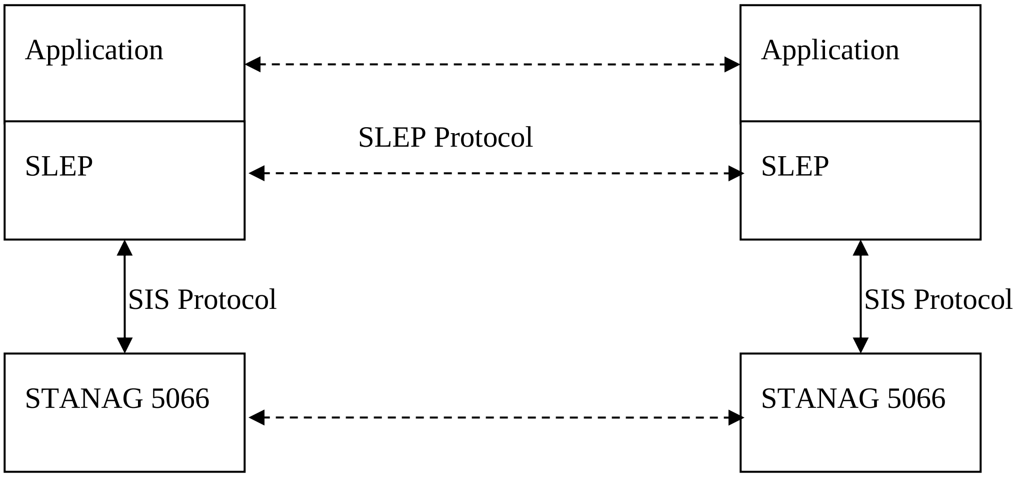
4 Motivation for SLEP

SLEP is designed to provide a general purpose building block and to address a number of deficiencies in RCOP/UDOP. This is driven by two specific application requirements.

1. XEP-0365 defines XMPP operation over RCOP. This works reasonably in practice but does not give compression which is highly desirable. The new stream service of SLEP provides a better solution for XMPP server to server communication.
2. ACP 142 provides a good messaging service but is a complex choice for point to point. It also incurs about 60 bytes overhead which could be avoided. CFTP is simpler, but functionally inadequate. SLEP enables a simple point to point messaging protocol with modern capabilities.
3. A stream service is needed to efficiently support TCP-like applications over HF.

5 SLEP Services

5.1 SLEP Model



SLEP is a layer service. It operates over STANAG 5066 and accesses the STANAG 5066 service using the STANAG 5066 SIS Protocol. SLEP defines a peer protocol and provides services to applications using SLEP.

5.2 Bind and Unbind Services

5.2.1 SLEP_BIND_REQUEST

Application -> SLEP.

Arguments:

1. SAP. Value of SAP (0-15). Same SAP is used for local and remote system.
2. Extended Address. Optional. 0-255. 0 is equivalent to no extended address. Same extended address is used for local and remote system.
3. RANK. As for S5066 S_BIND_REQUEST
4. Partial Datagrams. If this option is selected, partial unreliable datagrams may be returned as a list of complete blocks. This option is selected by unreliable datagram receiver. This option is not selected by default.
5. Block Fragments. If this is selected, partial blocks may be returned when Partial Datagrams are selected. This option needs to be selected by both unreliable datagram sender and receiver, as sender needs to use the STANAG 5066 Non-ARQ with Errors service to achieve this. This option is not selected by default.
6. Do not confirm Reliable Datagram. This reduces reliability slightly and slightly reduces protocol overhead, by eliminating the SLEP acknowledgment. Reliable Datagram is confirmed when all data has been confirmed by the peer STANAG 5066 node. This

option needs to be selected by both reliable datagram sender and receiver. There is no negotiation of this capability, so the configuration is a priori. This option is not selected by default.

5.2.2 SLEP_BIND_ACCEPTED

SLEP -> Application.

Arguments:

1. Maximum single block datagram size. This is the largest size of data that can be sent in a single block. This is particularly useful for unreliable datagram, where applications may choose to keep datagram size below this value.

5.2.3 SLEP_BIND_REJECTED

SLEP -> Application.

Arguments:

1. Reason. Values as for S5066 S_BIND_REJECTED, plus:
    a. Extended Addressing Not Supported
    b. Extended Address in use

5.2.4 SLEP_UNBIND_REQUEST

Application -> SLEP

No arguments.

5.2.5 SLEP_UNBIND_INDICATION

SLEP -> Application.

Arguments:

1. Reason. Values as for S5066 S_UNBIND_INDICATION

5.3 Unreliable Datagram Services

5.3.1 SLEP_UDP_REQUEST

Application -> SLEP

Arguments:

1. Data. Octets to be transferred.
2. ID. Integer provided by the application to correlate responses.
3. Priority. STANAG 5066 Priority.
4. Address. STANAG 5066 Unicast or Broadcast Address.
5. Minimum retransmission count.

6. No Compression. If specified, compression is never used.

5.3.2 SLEP_UDP_CONFIRM

SLEP -> Application.

Arguments:

1. ID.

Indicates that the datagram has been accepted for transfer.

5.3.3 SLEP_UDP_REJECT

SLEP -> Application.

Arguments:

1. ID.
2. Reason. The following reject values are defined:
    a. Datagram too large.
    b. ID already in use.
    c. Priority not currently allowed.

5.3.4 SLEP_UDP_INDICATION

SLEP -> Application

Arguments:

1. Data. Octets transferred.
2. Priority. STANAG 5066 Priority.
3. Sender Address. STANAG 5066 Address of Sender.
4. Receiver Address. STANAG 5066 Address of Recipient (may be unicast or broadcast).

Datagram is delivered.

5.3.5 SLEP_UDP_PARTIAL_INDICATION

SLEP -> Application

Arguments:

1. Data. The Partial Datagram
2. Priority. STANAG 5066 Priority.
3. Sender Address. STANAG 5066 Address of Sender.
4. Receiver Address. STANAG 5066 Address of Recipient (may be unicast or broadcast).

Partial datagram is delivered. This can include:

a. Blocks of SLEP data correctly delivered

b. Gaps/Missing blocks

c. Partial blocks, as returned by SIS.

5.3.6 SLEP_UDP_ABORT_REQUEST

Application -> SLEP

Arguments:

1. ID

5.3.7 SLEP_UDP_STATUS_INDICATION

SLEP -> Application

Arguments:

1. ID

2. Bytes Sent. Indicates the total number of bytes that have been sent OTA.

5.4 Reliable Datagram Services

5.4.1 SLEP_RDP_REQUEST

Application -> SLEP

Arguments:

1. Data. Octets to be transferred.
2. ID. Integer provided by the application to correlate responses.
3. Priority. STANAG 5066 Priority.
4. Address. STANAG 5066 Unicast Address.
5. Max RDP Time. A maximum time for the transfer.

Note that the protocol limits maximum datagram size to about 80 Mbytes.

5.4.2 SLEP_RDP_CONFIRM

SLEP -> Application.

Arguments:

1. ID.

Indicates that the datagram has been successfully transferred.

5.4.3 SLEP_RDP_REJECT

SLEP -> Application.

Arguments:

1. ID.
2. Reason. The following reject values are defined:
    a. Datagram too large.
    b. ID already in use.
    c. Priority not currently allowed.
    d. Broadcast address not allowed.
    e. Delivery Failure
    f. Rejection by peer application (e.g., format not recognized) encoded as two byte integer. SLEP reserves some of these values for generic rejection reasons that **may** be used by any SLEP application.
3. Reject Information. Application information as to why message was rejected.

5.4.4 SLEP_RDP_INDICATION

SLEP -> Application

Arguments:

1. ID. An identifier so that SLEP layer can correlate the response.
2. Data. Octets transferred.
3. Priority. STANAG 5066 Priority.
4. Address. STANAG 5066 Address of Sender.

Datagram is delivered.

5.4.5 SLEP_RDP_RESPONSE

This allows the application to confirm that it has reliably received the datagram. It also allows the application to reject the datagram, for example because it cannot be parsed. Application -> SLEP.

Argument:

1. ID.
2. Result.
    a. Accepted
    b. Rejected
3. Reject Information. Application information as to why message was rejected encoded as two byte integer.

5.4.6 SLEP_RDP_ABORT_REQUEST

Application -> SLEP

Arguments:

1. ID

5.4.7 SLEP_RDP_STATUS_INDICATION

SLEP -> Application

Arguments:

1. ID

2. Percentage sent. This indicates the faction of data that has been sent to the peer.

3. Percentage received. This indicates the fraction of the data confirmed by peer.

5.5 Datagram Flow Control Services

The following flow control mechanism is used to control data flow for both datagram services.

5.5.1 SLEP_DP_CONTROL_PRIORITY_INDICATION

SLEP -> Application

Argument1:

1. Minimum Priority. Only datagrams of this priority or higher may be submitted. This argument can have a value of 16 (one higher than the maximum STANAG 5066 priority) to indicated that no datagrams may be submitted. A value of zero indicates that any datagram may be submitted.

5.5.2 SLEP_DP_CONTROL_DESTINATION_INDICATION

SLEP -> Application

Arguments:

1. Address. This control only applies to this specified STANAG 5066 address.
2. Flow ON. If set to True, datagrams may be submitted to this destination. If set to False, datagrams may not be submitted to this destination.

5.6 Stream Services

The Stream service provides a byte stream service. The service definition is block based, to reflect a model of flow control based on blocks.

5.6.1 SLEP_STREAM_INIT_REQUEST

Application -> SLEP. Starts a new stream.

Arguments:

1. ID. Application provided ID to identify stream.
2. Address. STANAG 5066 Unicast Address of peer.
3. Priority. STANAG 5066 Priority.
4. Compression.. If set, stream is compressed.
5. Compression Options. This specifies ZLIB (RFC 1950) options, including:
    a. Compression Method
    b. Compression Level
    c. Preset dictionary. Use of dictionaries must be agreed a priori.
6. Stream Timeout. This is the time that will lead to stream being terminated if no data gets through. It might typically be set to 20-30 minutes.
7. Immediate Connect. Controls if connect confirm is immediate (local). If this is set, stream may get rejected after initial confirm and data.

Use of Immediate connect avoids an initial handshake to establish the stream, which reduces latency by allowing data to be transmitted along with the stream initiation protocol.

5.6.2 SLEP_STREAM_INIT_CONFIRM

SLEP -> Application.

Arguments:

1. ID.

5.6.3 SLEP_STREAM_INIT_REJECT

SLEP -> Application.

Arguments:

1. ID.
2. Reasons:
    a. Invalid Parameter.
    b. Too many streams.
    c. STANAG 5066 Timeout
    d. Peer Not Bound
    e. Peer Not Responding

    f. STANAG 5066 Error
    g. Connection Timed Out

5.6.4 SLEP_STREAM_INIT_INDICATION

SLEP -> Application. Incoming stream being offered to responder.

Arguments:

1. ID. SLEP generated identifier to uniquely identify the stream.
2. Address. STANAG 5066 Address of Initiator.
3. Priority. STANAG 5066 Priority.

5.6.5 SLEP_STREAM_INIT_RESPONSE

Application -> SLEP. Accept or reject incoming stream. This also controls compression and timeout of the reverse stream, analogous to SLEP_STREAM_INIT_REQUEST.

Arguments:

1. ID.
2. Accept or Reject.
3. Reject Reason.
4. Compression.. If set, stream is compressed.
5. Compression Options. This specifies ZLIB (RFC 1950) options, including:
    a. Compression Method
    b. Compression Level
    c. Preset dictionary. Use of dictionaries must be agreed a priori.
6. Stream Timeout. This is the time that will lead to stream being terminated if no data gets through. It might typically be set to 20-30 minutes.

5.6.6 SLEP_STREAM_DATA_REQUEST

Application -> SLEP. Initiator or responder sends a block of data.

Arguments:

1. ID.
2. Data.
3. Push. If this is set to true, data will be sent immediately. If set to FALSE, the SLEP provider will wait for more data.

5.6.7 SLEP_STREAM_DATA_CONFIRM

SLEP -> Application.

Argument:

1. ID

This confirms data is accepted and allows client to send another data block. This provides a mechanism for SLEP to flow control the application.

5.6.8 SLEP_STREAM_DATA_REJECT

SLEP -> Application.

Argument:

1. ID
2. Reason:
   a. Data too large.
   b. Invalid Transfer ID
   c. Previous Data not confirmed.
   d. Stream Closed.
   e. Protocol Error
   f. Timed Out

This rejects data submitted.

5.6.9 SLEP_STREAM_DATA_INDICATION

SLEP -> Application.

Arguments:

1. ID.
2. Data.

Application must accept the data. Note that this is a stream service, and received blocks of data are not expected to match the blocks sent. There is no size limit on the volume of data received.

5.6.10 SLEP_STREAM_CLOSE_REQUEST

Application -> SLEP.

Arguments:

1. ID.
2. Transfer All Data. If this is chosen, data transfer to the peer is completed before the stream is closed. If this is not chosen, then the close is treated like an abort; close is immediate and data sent to the peer may be lost.

Initiator or Responder requests SLEP to close a stream.

5.6.11 SLEP_STREAM_CLOSE_CONFIRM

SLEP -> Application.

Arguments:

1. ID
2. Data Received. This communicates to the application how the received data stream was terminated with the following alternative values:
    a. All data received. Peer requested Transfer All Data and it was received.
    b. Data missing. Peer requested Transfer All Data, but not all data was received.
    c. Not known. Peer did not request Transfer All Data.

This confirms that the stream has been closed and that data has been delivered according to the request parameters.

5.6.12 SLEP_STREAM_CLOSE_INDICATION

SLEP -> Application.

Arguments:

1. ID.
2. Reason:
    a. Initiator Request
    b. SLEP failure.
    c. STANAG 5066 Timeout
    d. STANAG 5066 Failure
3. Data Received. Same values as in SLEP_STREAM_CLOSE_CONFIRM.

SLEP tells SLEP user that the (inbound part of the) stream is being closed and that no more data will be received. The application MAY send further data and MUST then confirm close of the stream using SLEP_STREAM_CLOSE_RESPONSE in order to close the outbound part of the stream

5.6.13 SLEP_STREAM_CLOSE_RESPONSE

Application -> SLEP.

Arguments:

1. ID.
2. Transfer All Data. If this is chosen, data transfer to the peer is completed before the stream is closed. If this is not chosen, then the close response is treated like an abort and data may be lost. This option is not available to the receiver of a unidirectional stream.

Application confirms close of stream and application level and requests SLEP to complete stream closure.

6 Providing the SLEP Service

6.1 SLEP PDUs

The SLEP PDUs defined below, with a block of header data preceding user data. SLEP header and user data need to fit in the SIS U_PDU. There are two variant encodings. The first format is for data PDUs.

<table>
<tr><td>MSB
7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>LSB
0</td></tr>
<tr><td colspan="2">Version=0</td><td>Data=1</td><td>Ext Addr</td><td>Compressed</td><td colspan="3">Header Length</td></tr>
<tr><td colspan="8">MSB Address (optional) LSB</td></tr>
<tr><td colspan="8">MSB Transfer ID (2 bytes, optional)
LSB</td></tr>
<tr><td colspan="8">MSB Block Info (0-4 bytes)
LSB</td></tr>
<tr><td colspan="8">MSB Data (to maximum PDU size of up to 2048)
LSB</td></tr>
</table>

This PDU is encoded as follows:

1. Version is set to 0 for this version of SLEP.
2. Data bit set to 1, to indicated this is a data PDU.
3. Ext Addr. If this bit is set, an extended address is used, and this is encoded in the Address Byte. Note that the same extended address is used for both local and remote systems.
4. Compressed. If set to 1, Specifies that the PDU is using compression.
5. Header length specifies the length of the Transfer ID plus Block Info and any future headers. For Unreliable Datagram transferred in a single block, there is no transfer ID and zero length block information. For all other transfers, there is a two byte Transfer ID, and so the length of Block Info can always be determined.

The Block Info encoding and service semantics depends on the number of bytes in the block information.

Number of Bytes	Service	Encoding
0	Datagram	Entire datagram in PDU.
1	Datagram	Block number of PDU in bits 4-7. Total block count for datagram in bits 0-3.
2	Datagram	Block number of PDU in first byte. Total block count for datagram in second byte.
4	Datagram	Block number of PDU in first two byte. Total block count for datagram in bytes 3-4.
3	Stream	This encodes the block number for the PDU.

The control PDU is encoded as follows:

MSB 7	6	5	4	3	2	1	LSB 0
Version=0		Data=0	Ext Address	Type			
MSB				Extended Address (optional)			
MSB				Transfer ID (2 bytes)			LSB
MSB				Block Info (1-3 bytes - optional)			

		LSB
MSB	Block Upper Bound (1-3 bytes – optional, same size as Block Info)	LSB
MSB	Reject Reason (2 bytes – optional)	LSB

The values of this PDU are set as follows:

1. Version is set to 0 for this version of SLEP.
2. Data bit set to 0, reflecting this is a control PDU.
3. If Ext Addr is set to 1, the Extended Address byte is included.
4. Type has the integer values defined in the following tables.
5. The Transfer ID identifies the Datagram or Stream to which the control message refers. Note that Transfer ID is defined by initiator, so direction of each of these PDUs is only used in one direction. Transfer ID is always present.
6. The size of the optional data can be inferred from the overall PDU length. Most PDUs have zero or one optional elements. If Upper Bound Block Info is present, Block Info will also be present and the optional data is split equally between them.
7. Block Info identifies a specific block.
8. Block Upper Bound specifies the highest block in a range.

The following table sets out the different control types. Datagram only values come first, then Stream only values.

Name	**Type**	**Block Info**	**UB Block Info**	**Reject Reason**	**Notes**
Datagram Ack	0	No	No	No	This confirms delivery of Reliable Datagram. Transfer ID identifies the datagram. Receiver -> Sender

Name	Type	Block Info	UB Block Info	Reject Reason	Notes
Datagram NACK	1	No	No	Yes	Peer rejects datagram (e.g., cannot parse). Application defined reason for the rejection is encoded in Reject Reason Receiver -> Sender
Datagram Discard	2	No	No	No	Request by sender to discard an un-acknowledged transfer. Sender -> Receiver
Datagram Discard Ack	3	No	No	No	Responder acknowledging discard. Receiver -> Sender
Datagram Probe	4	Yes	No	No	Sender sends when it has not received Ack in expected timeframe. Block Info shows block number of final datagram block, so that receiver can request resend. Sender -> Receiver
Datagram Block Repeat Request	5	Yes	Yes	No	Datagram receiver requests stream sender to send again the set of blocks identified by the range Block Info (lowest block) and Block Upper Bound (highest block). Receiver -> Sender

Name	Type	Block Info	UB Block Info	Reject Reason	Notes
Stream Block Repeat Request	6	Yes	Optional	No	Stream receiver requests stream sender to send again the set of blocks identified by the inclusive range Block Info (lowest block) and Block Upper Bound (highest block). Block length is always 3, so upper bound may be omitted if only one block is requested Receiver -> Sender
Stream Init Request	7	No	No	No	This is sent by the stream initiator at start of stream. Initiator -> Responder
Stream Init Confirm	8	No	No	No	This is sent by the stream responder. Responder -> Initiator.
Stream Reject	9	No	No	Yes	Peer refuses a stream init request. Responder -> Initiator
Stream Ack	10	Yes	No	No	This is sent at intervals by a stream receiver. Transfer ID identifies the stream. Block Info identifies the last “contiguous” block received. Receiver -> Sender

Name	Type	Block Info	UB Block Info	Reject Reason	Notes
Stream Close	11	Yes	No	No	Sent by a stream sender. This is sent each way on stream close. Th Block Info indicates the final block sent. The responder must wait for this to arrive. Sender -> Receiver.
Stream Error	12	No	No	Yes	Sent to indicate an error or reference to an unknown Stream Transfer ID. Stream is always closed after this message is used. Either direction
Blocks Rejected	13	Yes	Yes	No	Used by both Datagram and Stream Services. Communicates from sender to receiver a list of blocks rejected for reason "destination not responding" as sender cannot determine whether or not block has been delivered. Sender -> Receiver

SLEP defines the following Reject Reasons. All values from 0-255 are reserved for future standard reject reasons. Values of 256 and greater are for use in applications defined to use SLEP. These values are application specific.

Name	Value	Notes
Temporary Reject	0	Failure due to temporary resource error or other non-

		permanent reason. Sender should try again later.
Datagram Too Large	1	Too large. Sender should reject.
Datagram Parse Error	2	Receiver could not interpret datagram. Sender should reject.
Permanent Reject	3	Other failure where repeating is not expected to work. Sender should reject.
Not Confirmed	4	Datagram arrived but was not confirmed by the application.
Abort Connection	5	Condition has led to connection being aborted and local connection should be aborted
Unknown Transfer ID	6	Failure due to transfer ID
Invalid Ack Block	7	Reference to non-existent block
Invalid Protocol	8	Protocol Error
Timed Out	9	Operation timed out

6.2 Bind Services

The SLEP_BIND service is mapped directly onto the S5066 S_BIND service.

Extended addressing requires a new SIS service that is not currently defined. Prior to this service being defined, extended addressing requests must be rejected.

A SLEP session starts with SLEP_BIND_REQUEST. After this is confirmed datagram and/or stream services may be used. The session is ended with SLEP_UNBIND. Note that this does not lead to any end to end protocol. Both ends must be bound in order for communication to take place.

6.3 Non-use of in-order Delivery

Use of STANAG 5066 in-order delivery superficially appears desirable, as it simplifies message re-assembly. However, STANAG 5066 in order delivery is implemented in the DTS layer. This means that if multiple applications are using in order delivery, they will interact, and one application can be delayed by another. For this reason, in order delivery **shall not** be used with SLEP.

This means that datagram and stream fragments may arrive out of order and the receiving SLEP needs to order them.

6.4 Achieving Reliability STANAG 5066 ARQ

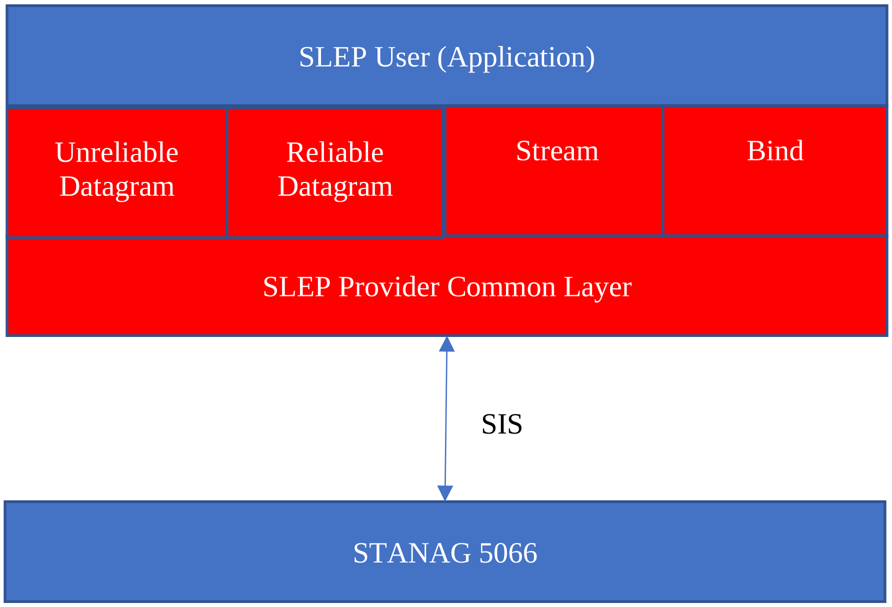
STANAG 5066 provides S_UNIDATA with an ARQ option that provides reliable transfer at the STANAG 5066 level. The ARQ service is accessed through the SIS protocol. There are a number of situations when SIS failure (e.g., application not connected over SIS for a period) will lead to loss of data between applications, such as SLEP, communicating over ARQ. These situations will be rare in practice but do need to be addressed. This issue can cause problems with some current protocols (e.g., CFTP). SLEP treats S_UNIDATA over ARQ as an “at most once” service.

The Steam Service includes mechanisms to deal with this potential loss in a timely manner.

The Reliable Datagram services incudes and end to end acknowledgement, so it can provide an “at least once” service. It also includes timers and retransmissions to address potential loss.

SLEP has confirmation on completion of Reliable Datagram, and occasional confirmation for Stream Service. This allows the receiver to identify missing blocks and request retransmission.

6.5 Supporting Multiple SLEP Services



The SLEP model is that the application using SLEP (SLEP User) will have the SLEP service integrated as a library and this will communicate locally with STANAG 5066 using the SIS protocol.

SLEP has three services: Unreliable Datagram; Reliable Datagram; and Stream. An application may use one, two or all three of these services, as shown in the diagram above. The SLEP specification is written to allow for a SLEP user to use all services. If only one or two services are used, some simplification is possible.

6.6 Directing S_UNIDATA_INDICATION to Correct Service

SLEP Bind services map directly to SIS and the bind to a single SAP is shared by all of the SLEP services.

The primary SIS service used by SLEP is S_UNIDATA. When an S_UNIDATA_REQUEST is passed over SIS, any S_UNIDATA_CONFIRM or S_UNIDATA_REJECT contains the data sent. This will enable the SLEP Provider Common Layer to pass this up correctly to the SLEP service that sent the data.

Incoming data will arrive a that SLEP Provider Common Layer as S_UNIDATA_INDICATION. The SLEP Common Layer will need to analyse the arriving data, to determine which service to pass it to, using the following rules.

1. If the data is non-ARQ, then it will be passed to the Unreliable Datagram Service.

2. ARQ data may be for Reliable Datagram or Stream, if both services are being used. The PDU is parsed.

a. If it is SLEP Data, Block Info size is examined. If this is three, the data is for stream service. Otherwise it is for Reliable Datagram.
b. If it is SLEP Control, the Control Types are unique for Reliable Datagram and Stream. The Control Type is used to determine the correct service.

Only data following these rules is passed to the corresponding service. If S_UNIDATA_INDICATION does not match the rules for any active service, it is discarded.

6.7 SLEP Message Queue Sublayer and Datagram Flow Control

The SLEP layer maintains a queue of UNIDATA blocks to be submitted over SIS using S_UNIDATA_REQUEST. This is modelled as a set of queues, with a separate queue for each peer. UNIDATA will be submitted until flow control from the SIS layer prevents further submission. Queued UNIDATA blocks are selected for submission on the following basis:

1. Only UNIDATA blocks from active queues are considered.
2. Highest priority UNIDATA blocks are submitted first, irrespective of queue.
3. For UNIDATA blocks of the same priority, the oldest is submitted first.

6.7.1 Sending UNIDATA

This queueing service is provided using a SLEP_QUEUED_UNIDATA_REQUEST, which has the same arguments as the S_UNIDATA_REQUEST that will be used to send the data over SIS.

When a message is sent over SIS, the message queue sublayer issues an SLEP_QUEUED_UNIDATA_CONFIRM. Correlated to the original request.

6.7.2 Removing Queued UNIDATA

There is an associated SLEP_QUEUED_UNIDATA_REMOVE, which allows the service to remove a submitted message from the queue (if it is still present in the queue).

6.7.3 General Flow Control

STANAG 5066 will use S_DATA_FLOW_ON and S_DATA_FLOW_OFF to control flow of traffic to STANAG 5066. The SLEP layer will only send messages using S_UNIDATA_REQUEST when data flow is ON. Flow control for the Stream service is managed using SLEP_QUEUED_UNIDATA_CONFIRM.

Flow control for Datagram services is managed using SLEP_DP_CONTROL_PRIORITY_INDICATION, which is generated by the SLEP message queue sublayer. When this layer receives S_DATA_FLOW_OFF from STANAG 5066, it will send SLEP_DP_CONTROL_PRIORITY_INDICATION to the SLEP user, with priority set to one higher than the highest priority message in the queue.

After S_DATA_FLOW_ON is received, SLEP_DP_CONTROL_PRIORITY_INDICATION is sent with the following rules:

- If the priority of the highest priority queued messages changes, send SLEP_DP_CONTROL_PRIORITY_INDICATION to the SLEP user, with priority set to one higher than the highest priority message in the queue.

- If the queue becomes empty, send SLEP_DP_CONTROL_PRIORITY_INDICATION to the SLEP user, with priority set to zero, which allows any message to be added to the queue.

This algorithm will enable SLEP users to hold data until the latest possible time. The highest priority allowed is also shared with the Datagram services.

6.7.4 Peer Flow Control

The STANAG 5066 service ARQ service can indicate that a destination is not responding. When this happens, it is sensible to back off sending data to this destination in order to give priority in the SIS queue to other destinations.

The SLEP_QUEUED_DESTINATION_NOT_RESPONDING_REQUEST service is used to indicate to the SLEP message queue sublayer that a specific STANAG 5066 peer is not responding.

For a configurable QUEUED_PEER_DELAY interval, the queue sublayer will not send any data to this peer.

If there is not traffic for any other destination, the SLEP Layer MAY choose to not set this delay.

When traffic is not being sent to a destination, the SLEP Layer will use SLEP_DP_CONTROL_DESTINATION_INDICATION (Flow ON = FALSE) to indicate to the SLEP User that datagrams may not be sent to this destination. After the delay, SLEP_DP_CONTROL_DESTINATION_INDICATION (Flow ON = TRUE) is used to turn flow on again.

6.7.5 Assigning Transfer IDs for Datagram Services

The two byte Transfer IDs for datagram services are assigned based on the 16 least significant bits of the Unix Time value. This gives a value that is unique for an approximately 18 hour period and one second apart. This approach enables two things:

1. When a SLEP system starts, it can guarantee to be able to pick a number that will not conflict with any recently allocated IDs.
2. A receiver can detect IDs that are 12-18 hours old and does not otherwise know about. A receiver should discard based on a time slightly ahead of the local clock time, to allow for clock skew relative to sender.

These prevent certain bad interactions when systems are restarted.

Transfer IDs are allocated uniquely and independently for each peer and for each SAP/Extended address used for that peer

The mechanism limits allocation to one per second. If there is a need for more transfer IDs, the following may be considered. Transfer IDs in the near future (up to an interval reflecting system restart time) may be used. Transfer IDs in the recent past (typically a few hours) that have not been used are also safe.

6.8 Unreliable Datagram Services

6.8.1 General Procedure

There is a single module for unreliable datagram. When SLEP_UDP_REQUEST is received, the procedure for Unreliable Datagram Sender is followed. When S_UNIDATA_INDICATION for the Unreliable Datagram Service are received they are handled by the Unreliable Datagram Receiver procedure.

S_UNIDATA_CONFIRM and S_UNIDATA_REJECT are handled by Sender and will always be associated with a specific datagram.

No SLEP Control messages are used by Unreliable Datagram service.

6.8.2 Procedure for Unreliable Datagram Sender

This section sets out the procedure for Unreliable Datagram Service to send a datagram,

6.8.2.1 Handling SLEP_UDP_REQUEST

The SLEP User will use SLEP_UDP_REQUEST to send an unreliable datagram.

1. If the priority of the request is lower than that of the current priority limit for the request, a SLEP_UDP_REJECT(priority) is passed to the User. The procedure terminates.

2. The User's request ID is checked that it does not conflict with another Request for the same application ID. If there is a conflict, a SLEP_UDP_REJECT(ID in use) is passed to the User. The procedure terminates

3. The data **is** compressed using the DEFLATE algorithm, specified in RFC 1951 "DEFLATE Compressed Data Format Specification version 1.3". If the compressed data is smaller than the original data, then the compressed data is used, and the SLEP data PDUs are marked as compressed.

4. Then the data will be broken into a number of blocks, ensuring that the resulting SLEP data PDUs do not exceed the STANAG 5066 MTU size, which will typically be 2048.

5. If the data exceeds the maximum for the largest collection of UDP blocks, a SLEP_UDP_REJECT(too large) is generated. The procedure terminates.

6. If the datagram can be sent with a single U_PDU, then a Transfer ID is not used, and block info is not needed. This is expected to be the most common mode for Unreliable Datagram.

7. A unique two byte Transfer ID will be generated for datagrams that need multiple U_PDUs. The value of the Transfer ID is generated using the Procedure for Datagram Transfer ID Assignment. Transfer ID is needed for Unreliable Datagram to ensure fragments are correctly associated.

8. A block info encoding of 1, 2 or 4 bytes will be chosen, dependent on whether the total number of blocks can be encoded in 4 bits, one byte or two bytes. U_PDUs **shall** be sent in order, with Block Info encoding the number of the U_PDU and the total number of U_PDUs. The first block has number 0.

9. Each block is sent using S_QUEUED_UNIDATA_REQUEST:

    a. Address and priority taken from SLEP_UDP_REQUEST.

    b. The non-ARQ service is requested, unless the Block Fragments service is requested, in which case non-ARQ with errors mode is used.

    c. Minimum retransmission count is set to the value supplied in SLEP_UDP_REQUEST.

    d. TTL is set to a large value. It is anticipated that non-ARQ data will usually be transmitted shortly after transmission. As there is no handling of TTL expiry, this needs to be set long enough to be confident that it will not expire.

10. The PDU Count is set to the total number of PDUs sent.

At this point, the datagram has been sent to the message queue sublayer and the submission procedure is complete. The Unreliable Datagram is considered pending, until the PDU count is reduced to zero.

6.8.2.2 Handling S_UDP_ABORT_REQUEST

If Abort is requested, remove all queued PDUs associated with the datagram from the queue using SLEP_QUEUED_UNIDATA_REMOVE.

6.8.2.3 Handling S_UNIDATA_REJECT

STANAG 5066 may reject a submitted PDU using S_UNIDATA_REJECT. A rejection can be correlated to a specific S_UNIDATA_REQUEST and thus to a pending datagram. The following actions will be taken:

1. Remove other queued PDUs associated with the same datagram from the queue using SLEP_QUEUED_UNIDATA_REMOVE.

2. Set the PDU count to zero and remove the pending transfer.

3. Inform the SLEP user of the failure using SLEP_UDP_REJECT, using the reason from S_UNIDATA_REJECT.

6.8.2.4 Handling Confirmations

The SLEP layer has two options for confirming PDU submission:

1. When a PDU has been written over SIS, the message queue layer will return SLEP_QUEUED_UNIDATA_CONFIRM.
2. Some STANAG 5066 implementation provide a mechanism to indicate when non-ARQ data is sent over the air, using S_UNIDATA_CONFIRM. This is non-standard but is an accepted industry convention. If this is available, it should be used as the mechanism to confirm PDU submission, and SLEP_QUEUED_UNIDATA_CONFIRM is ignored.

When a PDU is confirmed the PDU Count for the Datagram is reduced by 1. The volume of data transmitted is calculated and reported to the SLEP User using SLEP_UDP_STATUS_INDICATION.

If the PDU count reaches zero, inform the SLEP user of success using SLEP_UNIDATA_CONFIRM.

6.8.3 Procedure for Unreliable Datagram Receiver

This section sets out the procedure for Unreliable Datagram Service to receive a datagram.

This uses a timer UDP_DATA_TIMEOUT. This timer is necessary as some blocks may not be delivered. This timer is set whenever a block of data arrives. The value of this timer should be set to a value which is long enough so that it is very unlikely that block of data will arrive after the timer has expired. This will typically be a few minutes.

Note that this UDP_DATA_TIMEOUT needs to be chosen to ensure that discard happens prior to Transfer ID wrapping.

6.8.3.1 Handling S_UNIDATA_INDICATION

The Unreliable Datagram Service will be passed any non-ARQ UNIDATA addressed to the SLEP User. The UNIDATA User Data is parsed following the SLEP rules.

1. If the data is a Control PDU or does not parse, it is discarded.

2. If there is no transaction ID or Block Info, the data from the PDU is uncompressed (if compression bit is set) delivered to the SLEP user using SLEP

3. The (source address, SAP/Extended Address, transfer ID) is used to find an existing Transfer. If not found, one is created, unless the transfer ID time is 12-18 hours in the past, in which case it is discarded.

4. If the new block is inconsistent with previously received blocks, the Transfer is deleted and the procedure terminates. The following possible inconsistencies are noted:

    a. Block compression does not match the previous blocks; or
    b. Different maximum block count; or
    c. Duplicate block number with different data.

5. If the block number has not previously been inserted, the block data is inserted into PDU list.

6. If there are still missing blocks, the UDP_DATA_TIMEOUT timer is (re)started for this Transfer, and the procedure terminates.

7. At this point, all data blocks have been received. The data from the blocks is assembled into the complete datagram.

8. If compressed, the data is decompressed. If this fails, the Transfer is discarded, the error noted and the procedure terminates.

9. A SLEP_UDP_INDICATION is generated and passed to the application.

10. The Transfer is deleted and the procedure terminates.

6.8.3.2 Handling UDP_DATA_TIMEOUT

This procedure is followed when UDP_DATA_TIMEOUT timer goes off, indicating that a partial datagram has been received:

1. If the SLEP User has not selected the Partial Datagrams service, all of the received blocks and transfer are discarded and the procedure finishes

2. If the SLEP User has selected the Partial Datagrams service, the delivered blocks will be delivered using SLEP_UDP_PARTIAL_INDICATION. No attempt is made at decompression. This service will generally be used without compression. If Partial Blocks Handling is selected, the additional steps set out in the following section are followed. All received blocks are discarded and the procedure terminates.

    a. If the Partial Blocks service is used, S_UNIDATA_INDICATION may contain partial blocks. If a block is received twice, use of full bock is preferred. A server may attempt to combine a block received twice with different fragments.

6.9 Reliable Datagram Services

6.9.1 General Procedure for Reliable Datagram Service

There is a single module for Reliable Datagram. When SLEP_RDP_REQUEST is received, the procedure for Reliable Datagram Sender is followed.

When S_UNIDATA_INDICATION for the Reliable Datagram Service are received they are parsed. Data messages are handled by the procedure for Reliable Datagram Receiver. control messages all have a specified Sender/Receiver direction, and so can be passed to the correct direction. The Transfer ID of the control message will enable the message to be associated with the correct transfer.

If a Control message arrives with an unknown Transfer ID the following actions are taken:

1. Datagram Probe. This is a request to sender for status of a datagram for which nothing has been received. Send a Datagram Block Repeat Request from 1 to 0, which requests sending of all blocks.

2. Datagram Block Repeat Request. This is a request to sender for repeat of part of an unknown datagram. Send a Datagram Discard for this datagram.

3. Other Control messages are discarded.

S_UNIDATA_CONFIRM and S_UNIDATA_REJECT will be correlated to the transfer that issues the associated S_UNIDATA_REQUEST.

6.9.2 Timers

Use of ARQ data means that in normal operation reliable datagram will be sent without need for timers.   There is a very small possibility of data loss, and reliable datagram uses timers to protect against these.   The potential losses are:

1. ARQ data is transferred, but not delivered to the peer SLEP user. This can happen if the SIS connection between SLEP user and STANAG 5066 is dropped for a period.
2. ARQ data is lost in transit. This can happen if a STANAG 5066 server fails or resets.
3. ARQ receiver loses state. This can happen if there is a full reset.

Use of timers in the SLEP layer means that in the event of loss, recovery can be achieved by resending only the lost data. This can be important for HF, for example if a transfer taking over an hour loses a few blocks. The model is that the sender is under control, so most timers are on sender side.

Because HF activity can have delays of many may minutes these timers will have minimum values of a few minutes.

The following sender timers are used:

1. DATAGRAM_DISCARD. This is set by the Max RDP Time in service request and is the time at which a datagram is discarded.

2. PROBE_TIMER.   This is timer is set whenever data is acknowledged. It is used to trigger sending a probe in the event that blocks are not acknowledged.

   It is also used to trigger sending a probe in the event that an Ack is not received.

3. REPEAT_DATAGRAM_DISCARD.   This timer is used to control repeat of datagram discard, in the event that a datagram discard is not acknowledged.

The following receiver timers are used

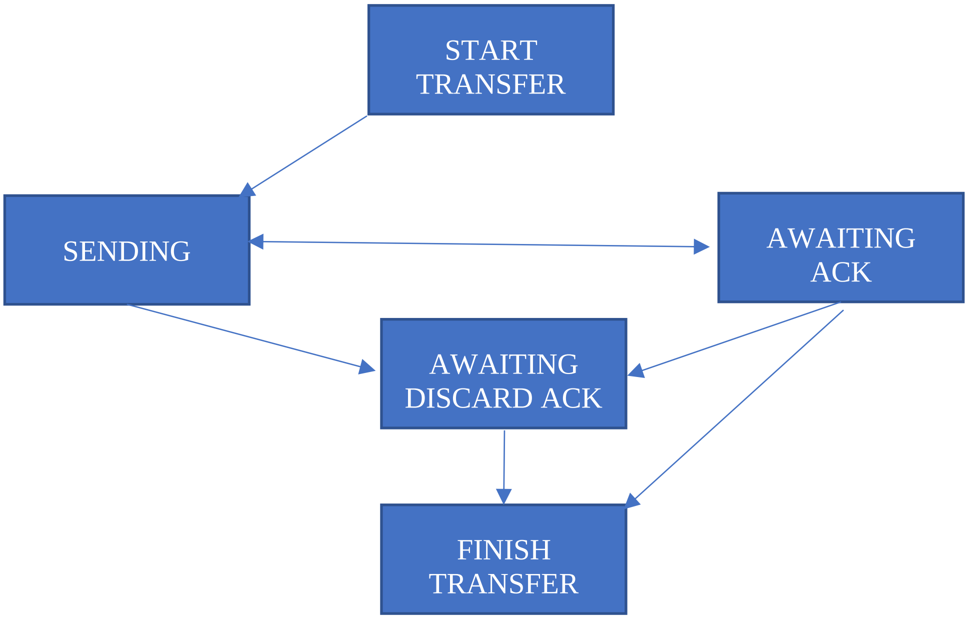
1. RDP_RECEIVE_DISCARD. This timer enables discard of partially received datagrams. It is set whenever data is received. It is expected to be a long timer.

2. RDP_RESPONSE. This timer controls how long a SLEP responder waits for the application to confirm delivery of a datagram.

3. RETAIN_TRANSFER. This is how long to retain information on a transfer after the datagram has been delivered. This ensures resilience of the protocol in certain situations.

6.9.3 Procedure for Reliable Datagram Sender

This section sets out the procedure for Reliable Datagram Service to send a datagram. This follows the state machine shown below.



6.9.3.1 START TRANSFER State

A new datagram is requested by the SLEP User with SLEP_RDP_REQUEST. The following procedure is followed to handle this:

1. If the priority of the request is lower than that of the current priority limit for the request, a SLEP_RDP_REJECT(priority) is passed to the User. The procedure terminates.

2. The User’s request ID is checked that it does not conflict with another Request for the same application ID. If there is a conflict, a SLEP_RDP_REJECT(ID in use) is passed to the User. The procedure terminates

3. If a broadcast address is requested, reject with SLEP_RDP_REJECT(Broadcast address not allowed).

4. The data **is** compressed using the DEFLATE algorithm, specified in RFC 1951 “DEFLATE Compressed Data Format Specification version 1.3”. If the compressed data is smaller than the original data, then the compressed data is used, and the SLEP data PDUs are marked as compressed.

5. Then the data will be broken into a number of blocks, ensuring that the resulting SLEP data PDUs do not exceed the S5066 MTU size, which will typically be 2048.

6. If the data exceeds the maximum for the largest collection of UDP blocks, a SLEP_RDP_REJECT(too large) is generated. The procedure terminates.

7. A unique two byte Transfer ID will be generated, following the Datagram Procedure for Transaction ID allocation.

8. A block info encoding of 1, 2 or 4 bytes will be chosen, dependent on whether the total number of blocks can be encoded in 4 bits, one byte or two bytes. U_PDUs **shall** be sent in order, with Block Info encoding the number of the U_PDU and the total number of U_PDUs. The first block is number 0.

9. Each block is sent using S_QUEUED_UNIDATA_REQUEST:

   a. Address and priority from SLEP_RDP_REQUEST.
   b. ARQ service is requested with Node confirmation.
   c. TTL is set to a value greater than the requested Max RDP Time, so that the UNIDATA will remain valid during allowed datagram transfer time.

10. The PDU Count is set to the total number of PDUs sent. A list of PDUs is also maintained, as this will be needed to correlate S_UNIDATA_CONFIRM and S_UNIDATA_REJECT to the datagram.

11. Set DATAGRAM_DISCARD timer to the requested Max RDP Time.

12. Move to SENDING state for this Transfer ID.

At the end of successful completion of the transfer start, the datagram will be fragmented into blocks and each block queued using the message queue sublayer.

The sender will maintain state for this transfer, until the transfer information is discarded. This state will include the state for each block, and a copy of each block so that it can be retransmitted if there is a need.

6.9.3.2 SENDING State

In this state the Datagram is being sent, with blocks being acknowledged at STANAG 5066 Level. In normal operation, the next state is AWAITING ACK. Set PROBE_TIMER on entering state.

Handling of the following events is considered.

1. SLEP_QUEUED_UNIDATA_CONFIRM is used to correlate data sent.

2. S_UNIDATA_CONFIRM.

   a. The PDU count is decremented by 1 if this S_UNIDATA_CONFIRM is acknowledging an S_UNIDATA that has not previously been confirmed.
   b. Set PROBE_TIMER.
   c. Calculate volume confirmed as percentage of compressed data acknowledged and indicate this and the percentage of data sent to SLEP User using SLEP_RDP_STATUS_INDICATION.
   d. If count is zero then:

      i. Default is to move to AWAITING ACK state.
      ii. If Do not confirm Reliable Datagram is selected for the session, indicate success to SLEP USER using SLEP_RDP_CONFIRM and move to FINISH TRANSFER state.

3. S_UNIDATA_REJECT. This is handled based on the reject reason:

   a. If "TTL Expired", the PDU is re-queued using S_QUEUED_UNIDATA_REQUEST and PDU count remains unchanged. TTL expired will only happen if the STANAG 5066 service has set it to a shorter value than requested.
   b. If "Destination Not Responding", any controlPDU is re-queued using S_QUEUED_UNIDATA_REQUEST. This error is usually associated with a CAS-1 soft link break, and sender cannot determine if PDU has been delivered. Send a "Blocks Rejected" control message to the peer. Note that a group of PDUs will typically all be rejected at the same time, and responses should be merged. The PDU count remains unchanged. The SLEP_QUEUED_DESTINATION_NOT_RESPONDING_REQUEST to the message queue sublayer is used to indicate the destination that is not responding.
   c. If "Destination SAP ID not Bound" is received, data is being transferred but not delivered to the peer. This error is only expected in response to a probe (which uses client acknowledgement). The datagram is rejected using

SLEP_RDP_REJECT(Temporary Reject), and it is anticipate that the layer above will try again later. The Transfer information is removed and go to FINISH TRANSFER state. Remove other queued PDUs associated with the same datagram from the queue using SLEP_QUEUED_UNIDATA_REMOVE. The SLEP_QUEUED_DESTINATION_NOT_RESPONDING_REQUEST to the message queue sublayer is used to indicate that traffic to this destination should be delayed.

    d. For other reject reasons the datagram is rejected using SLEP_RDP_REJECT(Permanent Reject). The Transfer information is removed and go to FINISH TRANSFER state. Remove other queued PDUs associated with the same datagram from the queue using SLEP_QUEUED_UNIDATA_REMOVE.

4. SLEP_RDP_ABORT_REQUEST. SLEP user requests to abort a datagram:
    a. Remove other queued PDUs associated with the same datagram from the queue using SLEP_QUEUED_UNIDATA_REMOVE.
    b. The Transfer information is removed and go to FINISH TRANSFER state.

5. Control Messages. These arrive as S_UNIDATA_INDICATION, which is parsed to identify the transaction ID. Invalid control messages are discarded. The following control messages are valid:
    a. Datagram Ack. Two options:
        i. If all the blocks have been sent, it is inferred that the Ack has arrived before all of the PDU Confirms and that the datagram has been successfully delivered. Inform SLEP User of success using SLEP_RDP_CONFIRM and go to FINISH TRANSFER state.
        ii. If some of the blocks have not been sent, this should not happen and is a serious protocol error. Follow procedure to discard this transfer and then start datagram transfer again with a new Transfer ID.
    b. Datagram Nack. . Two options:
        i. If all the blocks have been sent, it is inferred that the Nack has arrived before all of the PDU Confirms and that the datagram has not been delivered. Inform SLEP User of failure using SLEP_RDP_REJECT with the reason provided in the Nack. Then go to FINISH TRANSFER state.
        ii. If some of the blocks have not been sent, this should not happen and is a serious protocol error. Follow procedure to discard this transfer and then start datagram transfer again with a new Transfer ID.

    c. Datagram Block Repeat Request. Send the requested blocks using S_QUEUED_UNIDATA_REQUEST and PDU count is incremented by the number of PDUs sent and the PDUs are marked as not confirmed.

6. DATAGRAM DISCARD timer. Send Datagram Discard control message using S_QUEUED_UNIDATA_REQUEST with same priority as the datagram and move to AWAITING DISCARD ACK state.

7. PROBE_TIMER. This timer is started when entering state and reset when any data is confirmed. It is used to drive resend of data that is not confirmed and may not have been delivered. Send a Datagram Probe control message using S_QUEUED_UNIDATA_REQUEST with same priority as the datagram. Increment PDU Count by 1. Set PROBE_TIMER.

6.9.3.3 AWAITING ACK State

In this state end to end Datagram acknowledgement is waited for. In normal operation, the acknowledgement is received and the next state is INITIAL.

This state is reached when all PDUs comprising the datagram have been confirmed, so only Ack/Nack control messages are expected.

Set PROBE_TIMER, which will lead to a PROBE being sent if no valid control message is received.

Events are handled in the following way.

1. SLEP_QUEUED_UNIDATA_CONFIRM is ignored.

2. S_UNIDATA_CONFIRM is ignored.

3. S_UNIDATA_REJECT is ignored.

4. Control Messages. These arrive as S_UNIDATA_INDICATION, which is parsed to identify the transaction ID. Invalid control messages are discarded. The following control messages are valid:
   a. Datagram Ack. Inform SLEP User of success using SLEP_RDP_CONFIRM and go to FINISH TRANSFER state.
   b. Datagram Nack. Inform SLEP User of failure using SLEP_RDP_REJECT with the reason provided in the Nack, Then go to FINISH TRANSFER state.
   c. Datagram Block Repeat Request. Move to SENDING state and process.

5. SLEP_RDP_ABORT_REQUEST. SLEP user requests to abort a datagram:
   a. Remove other queued PDUs associated with the same datagram from the queue using SLEP_QUEUED_UNIDATA_REMOVE.
   b. The Transfer information is removed and go to FINISH TRANSFER state.

6. DATAGRAM DISCARD timer. Send Datagram Discard control message using S_QUEUED_UNIDATA_REQUEST with same priority as the datagram and move to AWAITING DISCARD ACK state.

7. PROBE_TIMER. Send a Datagram Probe control message using S_QUEUED_UNIDATA_REQUEST with same priority as the datagram. The probe control message is sent with Client Confirmation, so that the situation where the peer SLEP application is not connected can be detected. Note that data and other control messages use the more efficient Node Confirmation. Set PDU Count to 1. Move to SENDING state.

6.9.3.4 AWAITING DISCARD ACK State

This state is used when the requested Max RDP Time (DATAGRAM_DISCARD) is exceeded. Set REPEAT_DATAGRAM_DISCARD timer.

Handling of the following events is considered.

1. Control Messages. These arrive as S_UNIDATA_INDICATION, which is parsed to identify the transaction ID. Invalid control messages are discarded. The following control messages are valid:
   a. Datagram Discard Ack. Move to INITIAL state.

2. SLEP_RDP_ABORT_REQUEST is ignored.

3. REPEAT_DATAGRAM_DISCARD timer. Send Datagram Discard control message using S_QUEUED_UNIDATA_REQUEST with same priority as the datagram and reset REPEAT_DATAGRAM_DISCARD timer. The number of Datagram Discard control messages sent is counted. After this reaches a configurable maximum limit and the final timer goes off, the procedure terminates and move to INITIAL state.

4. Other Events are ignored.

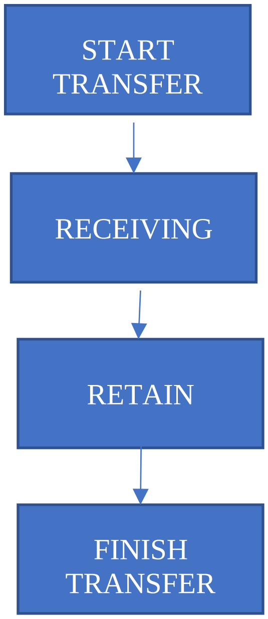
6.9.3.5 FINISH TRANSFER State

The transfer is finished and information discarded.

6.9.4 Procedure for Reliable Datagram Receiver

.

This section sets out the procedure for Reliable Datagram Service to receive a datagram. This follows the state machine shown below.



6.9.4.1 START TRANSFER State

The Reliable Datagram Service will be passed any ARQ UNIDATA addressed to the SLEP User. Data arrives as S_UNIDATA_INDICATION. The UNIDATA User Data is parsed following the SLEP rules to extract control and data messages. If parse fails, the data is discarded. The (source address, SAP/Extended Address, transfer ID) tuple is used to find an existing Transfer.

Control and Data messages will usually be associated with an active transfer in either RECEIVE or RETAIN state. Handling of control and data messages in the context of these states is described below.

Control messages and those with unknown Transfer ID are discarded.

If a new incoming Transfer ID is 12-18 hours in the past, it is discarded. Otherwise a new receive transfer is created for the (source address, SAP/Extended Address, transfer ID) tuple and the data block is processed in RECEIVING state.

6.9.4.2 RECEIVING State

Data blocks arriving in RECEIVING state are handled using the following procedure.

1. If the new block is inconsistent with previously received blocks, the moves to RETAIN state. The following possible inconsistencies are noted:

    a. Block compression does not match the previous blocks; or
    b. Different maximum block count; or
    c. Duplicate block number with different data.

2. If the block number has not previously been inserted, the block data is inserted into PDU list.

3. If there are still missing blocks, the RDP_RETAIN_TRANSFER timer is (re)started for this Transfer, and the procedure terminates.

4. At this point, all data blocks have been received. The data from the blocks is assembled into the complete datagram.

5. If compressed, the data is decompressed. If this fails, the Transfer is discarded, the error noted and the procedure terminates.

6. Deliver the complete datagram to SLEP USER using SLEP_RDP_INDICATION.

7. If “Do not confirm Reliable Datagram” is selected for the session move to RETAIN state..

8. In normal operation, procedure now waits for the SLEP User to confirm reception using SLEP_RDP_RESPONSE and setting a RDP_RESPONSE timer. The following choices are made, and the result of the choice recorded with the transfer:

    a. If the RDP_RESPONSE timer goes off, the SLEP user has not confirmed the transfer. Send a “Datagram NACK” control message with reason “Not Confirmed” at priority of the datagram.
    b. If the RDP_RESPONSE is “Accepted”, send a “Datagram Ack” control at priority of the datagram.
    c. If the RDP_RESPONSE is “Rejected”, send a “Datagram Nack” control with reason taken from RDP_RESPONSE at priority of the datagram.

9. Move to RETAIN state.

The following events in RECEIVE state are considered:

1. RDP_RETAIN_TRANSFER Timer: When this timer goes off, the transfer is discarded. This timer enables discard of datagrams that have been abandoned by the sender. Move to FINISH TRANSFER state.

2. Datagram Discard Control Message: If this message is received, the Transfer associated with the Transfer ID is deleted, and a Datagram Discard Ack is sent at the same priority as the datagram. Move to RETAIN state.

3. Datagram Probe Control Message: The receiver identifies missing blocks and send one or more Block Repeat Request messages to the sender.

4. Blocks Rejected Control Message. Each of the blocks indicated should be examined. If any blocks have not been received send one or more Block Repeat Request messages to the sender.

5. Other Control Messages are discarded.

6. S_UNIDATA_REJECT for any control message.. This is handled based on the reject reason:

    a. If "TTL Expired", "Destination Not Responding" or "Destination SAP ID not Bound". the PDU is re-queued using S_QUEUED_UNIDATA_REQUEST.

For other reasons, there is no action.

6.9.4.3 RETAIN State

On entering this state, set RDP_RETAIN_TRANSFER timer. This Transfer ID is retained for an extended period. This is to deal with certain data loss scenarios. It should be shorter than the minimum anticipated period for Transfer ID reuse.

The following events are handled in RETAIN State:

1. Data Blocks: Any data blocks received in RETAIN STATE are ignored

2. Datagram Discard Control Message: If this message is received, a Datagram Discard Ack is sent at the same priority as the datagram

3. Datagram Probe Control Message: The response previously sent for the datagram is sent again.

4. Other control Messages are ignored.

5. RETAIN_TRANSER Timer: When the RDP_RETAIN_TRANFER timer goes off, record of the Transfer ID is removed. Move to FINISH TRANFER state.

6.10 Streaming Services

SLEP provides a bidirectional stream service between a pair of applications that is initiated by one application. The SLEP layer will not time out a stream.   Stream close is always handled by the application.

6.10.1 Overview of how Streaming Service works

The streaming service is provided by two underlying unidirectional streams.   Blocks are numbered, as in the Datagram protocol, but there is no upper bound and the block number cycles.

Each stream has a 15 bit Stream Transfer ID, which is allocated by the stream initiator. The SLEP Transfer IDs used in PDUs contain the Stream Transfer ID, plus one additional bit to indicate the direction. PDUs sent by the Initiator have a high bit of 0, and PDUs sent by the Responder have a high bit of 1.

MSB 7	6	5	4	3	2	1	LSB 0
Responder	Stream Transfer ID						

To initiate a stream, the initiator sends the allocated Stream Transfer ID to the responder. The responder confirms this request.

Streams are identified to the SLEP user by an ID, and the Transfer IDs are hidden from the SLEP user.   This ID is referenced as “Stream ID”.

6.10.2 Use of STANAG 5066

Each Stream Control and Data PDU is mapped onto the STANAG 5066 ARQ service. Node Confirmation is requested, except where otherwise noted.

TTL **should** be set to a configurable value. This is likely to be a value somewhat longer than the STREAM_BROKEN timer, to deal with very unreliable links. TTL **may** be omitted and default system value used.

6.10.3 General Procedure for Stream Service

There is a single module for the Stream Service. When SLEP_STREAM_REQUEST is received, the procedure for Stream Initiation is followed.

When S_UNIDATA_INDICATION for the Stream Service are received they are parsed.

Data messages are handled as follows:

- If the Stream Transfer ID is known, the block is handled by the procedure for Stream Receiver.
- If the Stream Transfer ID is not known, it is possible that Immediate Connect has been used and that data has arrived ahead of the stream creation. To address this, the data message should be held for a configurable time, to see if the Stream Transfer ID is created. If it is created, the block is handled by the procedure for Stream Receiver. If the Stream Transfer ID is not created, a Stream Error control message is sent using SLEP_QUEUED_UNIDATA_REQUEST at the priority of the received S_UNIDATA_INDICATION. The transfer ID of this message is the unknown Stream Transfer ID. This message MUST NOT be sent in response to a Stream Error control message. Where multiple messages arrive with the same unknown Transfer ID, a single Stream Error control message MAY be sent.

If a Stream Init Request control message is received, the procedure for Stream Response is followed.

The Stream Transfer ID of each control message will enable the message to be associated with the correct stream.

If a Control message arrives with an unknown Stream Transfer ID a Stream Error control message is sent

Note that Stream Transfer IDs are generated by Stream Initiator for each peer and SAP/extended address, so there will be a separate list of active Transfer IDs for each direction, associated with each peer address and SAP/Extended address for each peer.

S_UNIDATA_CONFIRM and S_UNIDATA_REJECT will be correlated to the transfer that issues the associated S_UNIDATA_REQUEST.

6.10.4 Which Transfer ID to Use

A Stream has a pair of streams with data flowing in each direction, each with the same Stream Transfer ID, but with the streams distinguished by the Responder bit, which is set for PDUs sent by the responder. The stream PDUs identify the stream pair with a single Transfer ID.

6.10.5 Timers

Use of ARQ data means that in normal operation streams will work without need for timers. There is a very small possibility of data loss, and the stream service uses timers to protect against these. The potential losses are:

1. ARQ data is transferred, but not delivered to the peer SLEP user. This can happen if the SIS connection between SLEP user and STANAG 5066 is dropped for a period.
2. ARQ data is lost in transit. This can happen if a STANAG 5066 server fails or resets.
3. ARQ receiver loses state. This can happen if there is a full reset.

Use of timers in the SLEP layer prevents streams from hanging in the event of data loss.

The following timers are used:

1. STREAM_INIT_TIMER. This protects against the Stream Init Request or Stream Init Response getting lost.

2. EMPTY_BLOCK. This timer is used to control sending of empty blocks when there is no user traffic.

3. WAIT_FOR_NEXT_DATA. It is most efficient to send data in maximum size blocks. This is the time to wait for more data to arrive, before sending a short block.

4. STREAM_BROKEN. This timer is used by stream sender to close a stream where no data is getting through. It is set to the larger of a configurable minimum and the requested Stream Timeout.

5. STREAM_CLOSE. This protects against loss of the Stream Close control message and leads to resend.

6. FINAL_DATA_WAIT. This is the time to wait for data that has not arrived after a close has been received.

7. TTL. The time to set TTL for STANAG 5066 Unidata (optional).

8. HOLD_ID. This is a timer used when a connection has been closed. The transfer ID for the connection remains valid for this period. This is to enable correct discard of duplicate PDUs that may arrive after a stream has been correctly closed.

6.10.6 Application Considerations on Sending Data

An application using SLEP will send blocks of data using SLEP_STREAM_DATA_REQUEST. SLEP will process this data and acknowledge it. Then the application can send more data. The choice of data size is not specified here. The ideal data chunk would result in one stream packet being generated, after adding whatever was left over from the previous chunk. That should be enough data to generate approximately one MTU of data (typically 2048 bytes). This is after compression if compression is active. So providing chunks of data on the order of 1000-3000 bytes for an uncompressed stream, or 5000-10000 bytes for a compressed stream would be reasonable. Larger chunks are acceptable so long as the amount is not excessive. Much smaller chunks are also acceptable, but the compression ratio may suffer in this case, depending on the compressor implementation.

6.10.7 Procedure for Stream Initiation

Stream initiation is started with SLEP_STREAM_INIT_REQUEST call to initiate a stream. The following procedure is followed.

1. If any of the parameters are invalid (e.g., a broadcast address is requested), the request is rejected with SLEP_STREAM_INIT_REJECT(Invalid Parameter). If there are too many streams already open, the request is rejected with SLEP_STREAM_INIT_REJECT(Too many streams).

2. A new Stream Transfer ID is allocated for the sending stream. This will generally be one larger than the previous one allocated and MUST be different to any active Stream Transfer ID for the peer and SAP/extended address initiated by the local node. For the first stream, a random value is chosen, to minimize risk of conflict with any stream not previously closed cleanly.

3. Procedure for Stream Sender is started for the Stream Transfer ID.

4. If the Immediate Connect option is considered.
    a. If Immediate Connect is not selected, Stream State Pair is set to PENDING. This will prevent SLEP User data being sent on the stream before the stream is confirmed.
    b. If Immediate Connect is selected, Stream State Pair is set to ACTIVE, and SLEP_STREAM_INIT_CONFIRM is passed to SLEP user.

5. A Stream Init Request control messages is sent over the stream using SLEP_QUEUED_UNIDATA_REQUEST. This exchange ensures any previous use of the stream is cleared.

    a. The Steam Init Request MUST be sent with “client confirmation”, as opposed to “node confirmation” which is used for most other transactions. This ensures that a check is made for that the peer SLEP system is listening on SIS.

6. Set STREAM_INIT_TIMER. If this timer goes off, send the Stream Init Request control message again. This MUST be repeated a configurable number of times, after which the connection is rejected with SLEP_STREAM_INIT_REJECT, indicating that the connection attempt timed out.

7. If a Stream Reject control message is received, the request is rejected with SLEP_STREAM_INIT_REJECT. The stream is discarded and the procedure terminates.

8. If a Stream Init Confirm control Message is received, Set STREAM_BROKEN timer. If Immediate Connect option is not selected, the following actions are also taken

    a. State for the Stream Pair is set to ACTIVE.
    b.
    c. Stream initiation is completed with SLEP_STREAM_INIT_CONFIRM.

Note that when immediate connect is used, an implementation may choose to limit the amount of data sent prior to reject or confirm of the stream.

6.10.8 Procedure for Stream Response

This procedure is triggered by receiving a Stream Init Request control message.

1. If the inbound Transfer ID is associated with an existing Stream Pair:

    a. Close the stream pair using SLEP_STREAM_CLOSE_INDICATION. The pair request infers that peer has reset.
    b. Ignore any subsequence SLEP_STREAM_DATA_REQUESTS and expected SLEP_STREAM_CLOSE_RESPONSE.
    c. Discard the existing Stream Pair.

2. Set Stream Pair state to PENDING.

3. Generate a unique Stream ID

4. Indicate new Stream and Stream ID to SLEP User using SLEP_STREAM_INIT_INDICATION

5. Wait for SLEP_STREAM_INIT_RESPONSE. If rejected, send Stream Reject Control message with provided reject reason and procedure terminates.

6. Compression and timeout values for the second stream are taken from SLEP_STREAM_INIT_RESPONSE.

7. Send a Stream Init Confirm control message to the initiator using SLEP_QUEUED_UNIDATA REQUEST.

8. Start STREAM_BROKEN timer.

9. Start procedure for Stream Sender on the new Transfer ID.

10. Stare procedure for Stream Receiver on the first Transfer ID.

The Stream Pair will be activated when data is received from the peer, which indicates that the initiator has correctly received the Stream Init Confirm.

6.10.9 Procedure for Stream Sender

This section sets out the procedure for Stream Service to send data. Actions for possible events and timers are set out.

Any other events should not happen. Any other control messages should be discarded.

6.10.9.1 Receive SLEP_STREAM_DATA_REQUEST

1. If Stream Pair state is PENDING, wait until the state changes.

2. If Stream Pair state is CLOSED, reject request with SLEP_STREAM_DATA_REJECT(Stream Closed). Finish procedure.

3. If Sending State is "Waiting for Confirm", reject request with SLEP_STREAM_DATA_REJECT(Previous Data not confirmed). Finish Procedure.

4. Clear WAIT_FOR_NEXT_DATA timer.

5. Data is now prepared for sending. Data from the request is appended to the stream of uncompressed data to be sent. If compression is requested, this data stream is compressed using the ZLIB algorithm as specified in RFC 1950 "ZLIB Compressed Data Format Specification version 3.3", and this option is selected in the SLEP data PDUs. The ZLIB parameters are specifies in SLEP_STREAM_INIT_REQUEST. ZLIB defines the header of the stream. The stream is unbounded and no final checksum is ever sent.

6. Data is now taken from this stream to build a sequence of SLEP Data PDUs of maximum size. Each Data PDU is sent with the following sub-procedure:

    a. Set Sending State "Waiting for Confirm".
    b. Allocate the block number as one higher than the previous block or zero for the first block. Block number is encoded in three bytes.

c. Send the block using SLEP_QUEUED_UNIDATA_REQUEST:
    i. Peer address from stream
    ii. ARQ mode
    iii. Requested priority.
    iv. TTL set as specified in Section 6.10.2~~6.10.2~~.
    v. “Node Confirmation” is chosen.,

d. Wait for SLEP_QUEUED_UNIDATA_CONFIRM

e. Record submitted block and retain with transfer.

f. Clear Sending State.

7. If there is data in the stream not large enough for a full sized PDU:
    a. If Push was set on the last SELP_STREAM_DATA_REQUEST, send a partial block using SLEP_QUEUED_UNIDATA_REQUEST and wait for SLEP_QUEUE_UNIDATA_CONFIRM; or
    b. If Push was not set on the last SLEP_STREAM_DATA_REQUEST set timer WAIT_FOR_NEXT_DATA.

8. If there is no data left, set EMPTY_BLOCK_TIMER

9. Return SLEP_STREAM_DATA_CONFIRM, to indicate to the SLEP User that more data can be sent on the stream.

6.10.9.2 Receive S_UNIDATA_CONFIRM

This is not used for recording delivery.

However, STREAM_BROKEN timer is reset when this is received.

6.10.9.3 Receive S_UNIDATA_REJECT

If a block is rejected with S_UNDATA_REJECT, this is handled based on the reject reason.

If the stream has not been confirmed, the stream should be rejected using SLEP_STREAM_INIT_REJECT with value as follows:

a. STANAG 5066 Timeout for TTL Error.

b. Peer Not Bound for Peer Not Bound

c. Peer Not Responding for Peer Not Responding.

d. STANAG 5066 Error for other values.

If the stream is established, for reasons other than Destination Not Responding, the stream will be terminated. To do this a SLEP_STREAM_CLOSE_INDICATION will be given to the user with reasons:

a. STANAG 5066 Timeout for TTL

b. STANAG 5066 Failure for other Reasons.

Then send a Stream Close control PDU.

If the reason for the Reject is Destination Not Responding, the a “Blocks Rejected” control message is sent to the receiver. This is because the primary reason for this error is when a CAS-1 soft link is broken. The sender cannot determine if the block has been delivered or not. This control message requests the receiver to check and to ask for resend of missing blocks.

6.10.9.4 WAIT_FOR_NEXT_DATA Timer

If WAIT_FOR_NEXT_DATA goes off, there has been no new data and all of the pending data is sent as new Data PDU, following the sub-procedure of SLEP_STREAM_DATA_REQUEST.

Set EMPTY_BLOCK_TIMER.

6.10.9.5 EMPTY_BLOCK Timer

EMPTY_BLOCK timer enables an empty block to be sent if there is a period where there is no user data. This helps the receiver to clearly distinguish between the situation where there is no user data and when conditions are preventing data from getting through.

If EMPTY_BLOCK timer goes off, send a new Data PDU with zero length data, following the sub-procedure of SLEP_STREAM_DATA_REQUEST. Do not send a Data PDU if the close procedure has started.

Set EMPTY_BLOCK_TIMER.

6.10.9.6 Receive SLEP_STREAM_CLOSE_REQUEST

SLEP_STREAM_CLOSE_REQUEST will close both streams of a stream pair, using the following procedure.

1. If "Transfer All Data" is not set, then a Stream Error with "Abort Connection" shall be sen, and go to step 7 of this procedure.

2. Set Stream State to HALF-CLOSED. In this state, no more data shall be sent, but data may continue to be received.

3. Send Stream Close control message using SLEP_QUEUED_UNIDATA_REQUEST with ARQ and Priority of the stream. If Transfer All Data is requested, include the final block number in the control message.

4. Set STREAM_CLOSE timer. If this goes off, resend the Stream Close control message. Note that the STREAM_CLOSE timer needs to be set long enough to allow the peer to send an appropriate amount of “final data”, in addition to the time taken to respond. The timer shall be reset and a configurable number of retransmissions made.

5. Wait for Stream Close control message. When it arrives clear STREAM_CLOSE timer.

6. If the Block Number is set wait for the earlier of:
    a. All data up to the block number to arrive; or
    b. FINAL_DATA_WAIT timer

7. Report close using SLEP_STREAM_CLOSE_CONFIRM

8. Retain the stream information as valid for HOLD_ID time. Discard any PDUs that arrive referencing the Transfer ID.

9. Discard information on this stream and allow re-use of the Transfer ID.

6.10.9.7 Receive Stream Error Control Message

This is not expected to happen. The Transfer ID can be used to identify the stream that is no longer recognized by the peer.

1. Set Stream State to CLOSED.

2. Discard information on Stream.

6.10.9.8 Receive Stream Reject Control Messages

If Stream Reject control message is received, inform the SLEP user using SLEP_STREAM_INIT_REJECT with the value from the control message.

Discard information in the stream.

6.10.9.9 Receive Stream Block Repeat Request

Receive Stream Block Repeat Request control messages are used by the receiver to request resend of blocks that have not been received.

When these messages are received, send each requested block using SLEP_QUEUED_UNIDATA_REQUEST with the same parameters as the original send. Where acknowledgement has already been received for a requested block, the block should not be sent.

6.10.9.10 Receive Stream Ack Control Message

The Receive Stream Ack control message communicates the highest block number which has been received with all earlier blocks received. This is stored as a Window Edge marker. When allocating block numbers, this must be done to ensure that there is no wrap around.

When this control message is received, the sender can discard information on blocks which this control message acknowledges as being received.

6.10.10 Procedure for Stream Receiver

A Stream Receiver will get data and control messages as S_UNIDATA_INDICATION. Handling these is described below.

6.10.10.1 Received Data Blocks

The following procedure is followed for data received on a stream.

1. If Stream Pair State is PENDING, set Stream Pair State to ACTIVE.

2. If "Send Ack Counter" not set, set to zero. Otherwise increment.

3. If "Send Ack Counter" is equal to "Count of Blocks to Ack":

    a. Set Send Ack Counter to zero.
    b. Send Stream Ack control message with the block counter set to the Highest Block Received Counter.

4. If block number is before Highest Block Received counter, discard the block (it is a duplicate) and end procedure.
5. Add the received block to a list of received blocks, indexed by block number.

6. Set the Highest Block Received counter to the number of the highest block received, where all lower-numbered blocks have been received. This includes wrapping at the maximum block number.

7. Generate a data stream from the blocks held, up to and including block referenced by Highest Block Received counter. Then discard these blocks.

8. If compression is selected, decompress the stream, reversing the compression applied on send.

9. Deliver data received using SLEP_STREAM_DATA_INICATION.

6.10.10.2 Receive Stream Close Control Message

When a Stream Close control message is received, the following procedure is used.

1. Set Stream State to HALF CLOSED.

2. Wait for all data up to the reported block (from Block Info) to arrived, if necessary requesting retransmission following the process described in Section 6.10.10.5~~6.10.10.5~~.;
3.

4. Communicate stream close to SLEP User using SLEP_STREAM_CLOSE_INDICATION.

5. Handle any data submitted on the return stream using SLEP_DATA_REQUEST and wait for SLEP_STREAM_CLOSE_RESPONSE.

6. Send Stream Close control message to peer using SLEP_QUEUED_UNIDATA_REQUEST. If Transfer All Data is requested in SLEP_STREAM_CLOSE_RESPONSE, Block Info MUST be set in this message to the number of the last block of data sent.

7. Retain the stream information as valid for HOLD_ID time. Discard any PDUs that arrive referencing the Transfer ID.

8. Discard information on this stream and allow re-use of the Transfer ID.

6.10.10.3 Receive Error Control Message

This is not expected to happen. The Transfer ID can be used to identify the stream that is no longer recognized by the peer.

1. Set Stream State to CLOSED.

3. Discard information on Stream.

6.10.10.4 Receive Blocks Rejected Control Message

This control message is received when the STANAG 5066 layer rejects a block transfer and the sender cannot determine if a set of blocks have been delivered.

The receiver shall examine each of the blocks listed to determine if it has been delivered. If any of the listed blocks have not been received, each of the missing blocks is requested using a Stream Block Repeat control message.

6.10.10.5 Requesting Block Resend

It is the responsibility of the receiver to identify data loss and to request that the sender resend.

The receiver can request retransmission of blocks using the Stream Block Repeat control message to request one block or a range of blocks. This is used to address block loss, noting that this is unlikely but possible.

The heuristics chosen to do this are chosen by the implementor. If this is requested too early, it will lead to duplication of transfer. If requested too late, it will lead to delays.

Considerations:

- Blocks may be delayed due to poor HF conditions.
- Blocks may be delayed due to other traffic of higher priority.
- Zero length blocks suggest that the link is not being used to full capacity, and so delays are less likely.
- Occasional delivery of “early” packets suggest that conditions are poor and that the system is catching up..
- If there are just a few missing packets, it may be preferable to request early and take the hit on possible duplication.

7 Protocol Specifications and Address Assignments

7.1 SLEP SAP

While SLEP can be used with any SAP, this specification recommends use of default value of 10, prior to any official assignments.

7.2 STANAG 4406 Annex E

STANAG 4406 Annex E defines a compressed file format that is normally used with ACP 142. SLEP offers a more compact option for point to point transfer. The following SLEP options are used:

SLEP Capability	**Choice**
Type	Reliable Datagram
Acknowledgement	Yes
Extended Addressing	Deployment choice
Extended Data	Use for messages that will not fit in one block

Default address: SAP: 10; Extended Address: 1

7.3 MULE

MULE defines a compressed file format that is normally compressed with STANAG 4406 Annex E compression and used with ACP 142. SLEP offers a more compact option for point to point transfer. The following SLEP options are used:

SLEP Capability	Choice
Type	Reliable Datagram
Acknowledgement	Yes
Extended Addressing	Deployment choice
Extended Data	Use for messages that will not fit in one block

STANAG 4406 Annex E compression **may** be used for MULE or the MULE format message carried directly over SLEP

Default address (STANAG 4406 Annex E compression): SAP: 10; Extended Address 2.

Default address (direct MULE format): SAP 11; No Extended Address.

7.4 XEP-0365

XMPP and XEP-0365 transfer messages with RCOP. SLEP may be used instead to carry XEP-0361 protocol.

SLEP Capability	Choice
Type	Stream
Extended Addressing	Deployment choice

Compression	Yes

Default address: SAP: 10; No Extended Address