NATO OTAN

NATO STANDARDIZATION OFFICE
BUREAU OTAN DE NORMALISATION

NSO

18 January 2018 NSO/0065(2018)CAP1/4494

STANAG 4494 CAP1 (EDITION 1) - ADVANCED SATCOM NETWORK MANAGEMENT AND CONTROL ENGINEERING ORDER WIRE

References:
A. AC/322(SC/6)N(2010)0025, dated 8 April 2010 (Edition 1) (Ratification Draft 1)
B. AC/322-N(2017)0168-AS1 dated 10 January 2018

1. The enclosed NATO Standardization Agreement, which has been ratified by nations as reflected in the NATO Standardization Document Database (NSDD), is promulgated herewith.

2. Reference A. is to be destroyed in accordance with local document destruction procedures.

ACTION BY NATIONAL STAFFS

3. National staffs are requested to examine their ratification status of the STANAG and, if they have not already done so, advise the NSO, through their national delegation as appropriate of their intention regarding its ratification and implementation.

4. It should be noted that this standard entered development/ratification under AAP-03(I) and therefore is promulgated in its current format as approved by the related Tasking Authority in accordance with Reference B.

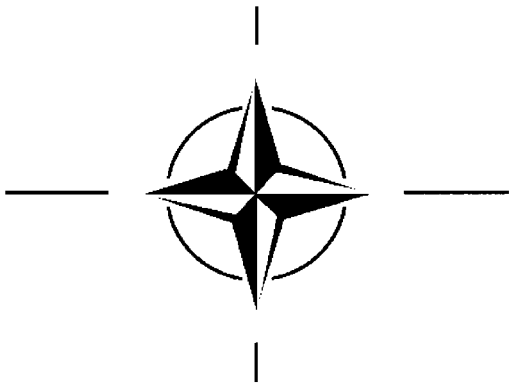
Edvardas MAŽEIKIS
Major General, LTUAF
Director, NATO Standardization Office

Enclosure:
STANAG 4494 (Edition 1)

NATO Standardization Office – Bureau OTAN de normalisation
B-1110 Brussels, Belgium Internet site: http://nso.nato.int
E-mail: nso@nso.nato.int – Tel 32.2.707.5556 – Fax 32.2.707.5718

STANAG 4494
(Edition 1)

NORTH ATLANTIC TREATY ORGANIZATION
(NATO)



NATO STANDARDIZATION OFFICE
(NSO)

STANDARDIZATION AGREEMENT
(STANAG)

SUBJECT: ADVANCED SATCOM NETWORK MANAGEMENT AND CONTROL ENGINEERING ORDER WIRE

Promulgated on 18 January 2018

Edvardas MAŽEIKIS
Major General, LTUAF
Director, NATO Standardization Office

RECORD OF AMENDMENTS

No.	Reference/Date of amendment	Date entered	Signature

EXPLANATORY NOTES

AGREEMENT

1. This STANAG is promulgated by the Director NATO Standardization Office under the authority vested in him by the North Atlantic Council.

2. No departure may be made from the agreement without informing the tasking authority in the form of a reservation. Nations may propose changes at any time to the tasking authority where they will be processed in the same manner as the original agreement.

3. Ratifying nations have agreed that national orders, manuals and instructions implementing this STANAG will include a reference to the STANAG number for purposes of identification.

RATIFICATION, IMPLEMENTATION AND RESERVATIONS

4. Ratification, implementation and reservation details are available on request or through the NSO websites (internet http://nso.nato.int; NATO Secure WAN http://nso.hq.nato.int).

RESTRICTION TO REPRODUCTION

FEEDBACK

6. Any comments concerning this publication should be directed to NATO/NSO – Bvd Leopold III - 1110 Brussels - Belgium.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

STANAG 4494
(Edition 1)

DRAFT NATO STANDARDIZATION AGREEMENT

(STANAG)

ADVANCED SATCOM NETWORK MANAGEMENT AND CONTROL ENGINEERING ORDER WIRE

1. (NU) INTRODUCTION

The purpose of this STANAG is to provide a standard for, and promote the interoperability among, providers of products for the Transport of Management and Control Information for Engineering Order Wire between a Hub and each of its Remotes in a Satellite-based network. Such products must be fully IP based and cover the transmission and reception of both data and voice. The network will be based on a Star Architecture: each Remote will have its own logical Link with the Hub, and all satellite traffic to or from that Remote will travel over that Link. The Hub will use “bent pipe” broadcast stream to all of its Remotes simultaneously, while each Remote will use an MF-TDMA access scheme (as dictated by the Hub) for its transmissions to the Hub. The Hub will dynamically assign changing amounts of bandwidth to each of its Remotes, based on each Remote’s changing amounts of data it needs to transmit. All data transmissions will be conducted under the TRANSEC methodology, thus ensuring a high level of security for the network and its users.

2. (NU) AIM

The Satellite Control and Command Service will provide NATO forces with high capacity satellite communications for strategic and tactical units. The aim of this agreement is to define a baseline set of interfaces and protocols so that system integrators are able to build and commission interoperable Satellite Control and Command communications using COTS based satcom technology.

3. (NU) AGREEMENT

The requirements specified in this STANAG are meant to ensure a basic commonality amongst NATO control and command satellite systems. Participating nations agree to use the characteristics contained in this STANAG for development and deployment of Satellite Control and Command systems.

4. (NU) GENERAL

The details of this agreement are defined in Annexes A-I. Annex A provides a list of acronyms and abbreviations.. Annex B contains definitions of frequently used terms. Appendix C is a list of other documents and standards which are referred to in this STANAG. Annex D provides an overview of an entire satellite communications system, with highlighting of the sections covered by this STANAG. Annex E provides a description of the establishment of a Link between a Hub and a coming-online Remote. Annex F discusses the steps necessary for the maintenance of a Link while it is active. Annex G describes the steps in the teardown of an existing Link. Annex H contains the detailed methodology by which data is prepared for transmission to a satellite; or is interpreted upon reception from a satellite. Annex I deals specifically with the details of TRANSEC encryption, decryption, and block headers. Annex J presents a non-binding discussion of ways that Quality of Service (QoS) could be integrated with this STANAG.

5. (NU) HIERARCHY

This STANAG is limited to the Physical and Data Link layers of the OSI model, excluding all higher levels. Figure B.1 shows the hierarchy of organization of STANAGs into which this STANAG falls.

Related documents are listed in Annex C.

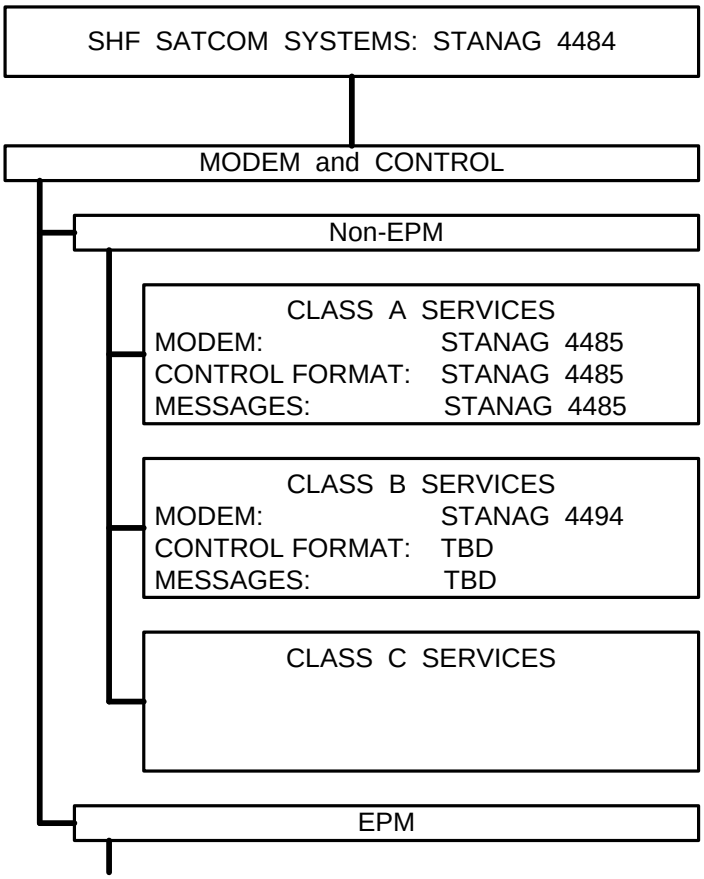


Figure B.1 Partial Organization of SHF SATCOM STANAGs

6. (NU) IMPLEMENTATION OF THIS AGREEMENT

Nations compliance to this agreement shall be satisfied when Satellite Control and Command gateway in that nation's forces are placed in service and interoperate with other nations' Control and Command terminals that comply with the characteristics detailed in this STANAG.

ANNEX A
ACRONYMS

GREEK

α	Roll-off factor
θ	deviation angle in hierarchal constellations
ρ	modulus of complex vector for modulation symbol constellation
φ	phase of complex vector for modulation symbol constellation

NUMBERS

8PSK	eight Phase Shift Keying

A

AAP	Allied Administrative Publications
ACK	ACKnowledgement
ACQ	ACQuisition
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
ANSI	American National Standards Institute

B

BPSK	Binary Phase Shift Keying
BUC	Block Up Converter
BW	BandWidth

C

CBWFQ	Class-Based Weighted Fair Queue
CCITT	Comité Consultatif International Téléphonique et Télégraphique (ITU-T)
CFB	Cipher FeedBack
CFI	Canonical Format Indicator

CSMA/CD	Carrier Sense Multiple Access / Collision Detection
COTS	Commercial Off The Shelf
CRC	Cyclic Redundancy Check

D

DAC	Digital to Analog Convertor
dB	deciBel
DH	Demand Header
DISC	DISConnect
DM	Disconnected Mode

E

EH	Encryption Header
EOW	Engineering Order Wire
EPM	Electronic Protective Measures
ET	Earth Terminal
ETSI	European Telecommunications Standards Institute

F

FDM	Frequency Division Multiplex
FDMA	Frequency Division Multiplex Access
FEC	Forward Error Correction
FED-STD	FEDeral STanDard
FH	Fragment Header
FIFO	First In, First Out
FIPS	Federal Information Processing Standards
FRMR	FRaMe Reject

H

HDLC	High level Data Link Control
Hz	Hertz

I

IBS	Intelsat Business Services
IDR	Intermediate Data Rate
IEEE	Institute of Electrical and Electronics Engineers
IESS	Intelsat Earth Station Standard
IETF	Internet Engineering Task Force
IF	Intermediate Frequency
IIOB	Interger In One Byte
INTELSAT	INternational TELecommunicatins SATellite organization
IP	Internet Protocol
IR	InRoute
IRIB	InRoute Information Block
ISO	International Organization for Standardization
ITU	International Telecommunications Union
ITU-R	International Telecommunications Union – Radio (previously CCIR)
ITU-T	International Telecommunications Union – Telecommunications (CCITT)
IV	Initialization Vector

K

KAT	Known Answer Test

L

LAN	Local Area Network
LL	Link Layer
LNB	Low Noise Block amplifier
LSB	Least Significant Bit

M

MAC	Media Access Control
MF TDMA	Multiple Frequency Time Division Multiple Access
MHz	MegaHertz
MILSATCOM	MILitary SATellite COMmunications
MIL-STD	MILitary STanDard
MODCOD	MODulation and CODing

modem	modulator-demodulator
MSB	Most Significant Bit

N

N/A	Not Applicable
NATO	North Atlantic Treaty Organization
NSA	Nato Standardization Agency
NU	Nato Unclassified

O

OOB	Out Of Band
OSI	Open Systems Interconnection (ISO 7498)

P

PCP	Priority Code Point

Q

QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying

R

RF	Radio Frequency
RNR	Receiver Not Ready
RR	Receiver Ready
RTP	Real-Time Protocol
Rx	Receiver

S

SABM	Set Asynchronous Balanced Mode
SABME	Set Asynchronous Balanced Mode Extended

SATCOM	SATellite COMmunications
SH	Segment Header
SHF	Super High Frequency 3 GHz to 30 GHz
SREJ	frame REJect
SRRC	Square Root Raised Cosine
STANAG	STANardization AGreement (NATO)

T

TCP	Transmission Control Protocol (layer four)
TCP/IP	Transmission Control Protocol (on top of) Internet Protocol
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TOS	Type Of Service (IPv4)
TPC	Turbo Product Code
TPID	Tag Protocol ID
TRANSEC	TRANsmission SECurity
Tx	Transmitter

U

UA	Unnumbered Ack (level 2)
UCP	Uplink Control Protocol
UD	Unnumbered Datagram
UDP	User Datagram Protocol
UI	Unnumbered I
UW	Unique Word

V

VH	Vlan Header
VID	Vlan IDentifier
VLAN	Virtual Local Area Network
VoIP	Voice over Internet Protocol

X

XDR	eXternal Data Representation

[This page intentionally left blank.]

ANNEX B
TERMINOLOGY

Bent Pipe: a satellite operating as a “bent pipe” receives an analog waveform on one frequency and polarization, and retransmits the same analog waveform on another frequency and polarization, without demodulating or processing the signal.

Block: a combination of payload data and the control information (Block Header) which is needed to successfully deliver that data.

Block Header: the control information needed to successfully deliver the contents (payload) of a Block.

Burst: for Upstream, the amount of data and control symbols which will fit in one Time Slot of a Time Frame; for Downstream, the transmission from the Hub to all of its Remotes during one Time Frame.

Data Path: one of four related components of communications between a Remote and its Hub: Downstream, Hub side; Downstream, Remote side; Upstream, Remote side, and Upstream, Hub side.

Data Fragmentation: a schema whereby Blocks are broken down into smaller sized fragments, thus allowing a fixed-size content to be completely filled, rather than left partially empty (because the next Block is too large to fit in the remaining space).

Data Segmentation: a schema whereby incoming packets are broken down into fixed size segments, allowing smaller packets to be intermingled with larger packets, instead of having to wait until the entirety of the larger packet has been completely transmitted.

Demand Header: request from a remote for Time Slots in an upcoming Time Frame

Downstream: transmissions from a Hub to all of its Remotes.

Earth Station: a satellite-communicating antenna and its ancillary equipment.

Forward Error Correction: a schema for detecting and correcting transmission errors, to minimize the need to retransmission of packets across the satellite link.

Hub: the equipment and software comprising the Earth Station at the center of a Star Network.

Initialization Vector: used with the X.509 encryption algorithm.

InRoute: a specific (one of many) frequency on which a Remote will transmit a Burst.

InRoute Group: a collection of InRoutes which are assigned consistent transmission attributes (e.g., FEC rate), such that a Remote may transmit on various InRoutes without changing anything except its transmission frequency.

Jitter: a measure of the variation of Latency on a packet-by-packet basis.

Latency: the total delay between events; it includes transit time, queuing, and processing delays.

Network Congestion: occurs whenever the volume of traffic exceeds the available bandwidth.

Packet Loss: a measure of the number of packets that are transmitted by a source, but not received by their destination.

Quality of Service: getting data from senders to receivers, in the most timely manner possible, as looked at from the viewpoint of all traffic in the network

Realtime Data: data whose delivery is extremely time sensitive, for example Voice (where delays would cause gaps in the received sound, degrading the ability of the listener to comprehend what was said).

Remote: the equipment and software comprising the Earth Station at a non-center node of a Star Network.

Remote Acquisition: the process whereby a Remote comes (back) up and requests admission into its Hub’s Star Network.

Service Level: an assignment of relative importance between different IP Packets as they arrive to be transmitted.

Star Network: a collection of nodes which are each connected to a central Hub: all communication between different nodes must travel from the first node to the Hub, and then from the Hub to the second node.

TDMA Burst: the amount of data and control symbols which will fit in one Time Slot of an Upstream Time Frame.

Throughput: indicates the amount of user data that is received by the end user application.

Time Frame: the fixed-size interval for which the Hub has defined a Timeplan for all of its Remotes.

Time Slot: the smallest division of a Time Division Multiplexing schema.

Timeplan: the Hub's definition, to all Remotes which belong to the same Inroute Group, of which Remote has been allocated which Time Slots on which InRoutes, during the upcoming Time Frame.

Timeplan ID: a unique identifier assigned by the Hub to each Remote, as it is acquired, which will be used in Timeplans to tell each Remote which Time Slots in which InRoutes have been assigned to that Remote.

Unique Word: a defined sequence of symbols which is sent by one Earth Station so that, upon reception by another Earth Station, the receiver will be able to synchronize with the symbol stream and accurately extract the payload blocks following the Unique Word.

Uplink Control Protocol: a schema whereby the Hub can tell a specific Remote to adjust its transmission characteristics for a specific InRoute.

Upstream: transmissions from a Remote to its Hub.

Virtual Local Area Network: a group of network nodes that communicate as if they were physically co-located, but allows the nodes to interact using LAN protocols even though some nodes are located remotely, and transparently interact with the other nodes over some non-LAN medium.

[This page intentionally left blank.]

ANNEX C
REFERENCES

Documents that are referenced from within this STANAG

STANAG 4484 – Overall SHF MILSATCOM Interoperability Standard

STANAG 4486 – FDMA non-EPM Modem for Class B Services

STANAG 4505 – SHF MILSATCOM Network Management and Control

FIPS PUB 140-2 – Security Requirements for Cryptographic Modules

FIPS PUB 197 – Advanced Encryption Standard

FIPS PUB 180-1 – Secure Hash Standard

FIPS PUB 186-2 – Digital Signature Standard

AAP-6 NATO Glossary of Terms and Definitions

FED-STD 1037C – Telecommunications Glossary of Telecommunications Terms

IESS-308 – Intelsat Earth Station Standards: Performance Characteristics for Intermediate Data Rates Digital Carriers using Convolutional Encoding / Viterbi Decoding and QPSK Modulation (QPSK/IDR)

IESS-309 – Intelsat Earth Station Standards: Performance Characteristics for Intelsat Business Services (IBS)

X.509 – International Telecommunication Union: Information Technology; Open Systems Interconnection; The Directory: Authentication Framework

IEEE 802.1D-2004 – IEEE Standard for Local and Metropolitan Area Networks Media Access Control (MAC) Bridges

IEEE 802.1Q-2005 – IEEE Standard for Local and Metropolitan Area Networks Virtual Bridged Local Area Networks

IEEE 802.3-2005 – IEEE Standard for Information technology-Specific requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CMSA/CD) Access Method and Physical Layer Specifications.

IEEE 802.16-2004 – IEEE Standard for Local and Metropolitan Area Networks; Part 16: Air Interface for Fixed Broadband Wireless Access Systems

ESTI EN 300 421 – Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services

ETSI EN 302 307 – European Standard (Telecommunications series) Digital Video Broadcasting (DVB)

IETF RFC 768 – User Datagram Protocol

IETF RFC 791 – Internet Protocol

IETF RFC 793 – Transmission Control Protocol

IETF RFC 1122 – Requirements for Internet Hosts – Communications Layers

IETF RFC 2474 – Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers

IETF RFC 4506 – XDR: External Data Representation Standard

ANNEX D
OVERVIEW

CONTENTS

D.1 (NU) Description of Entire System

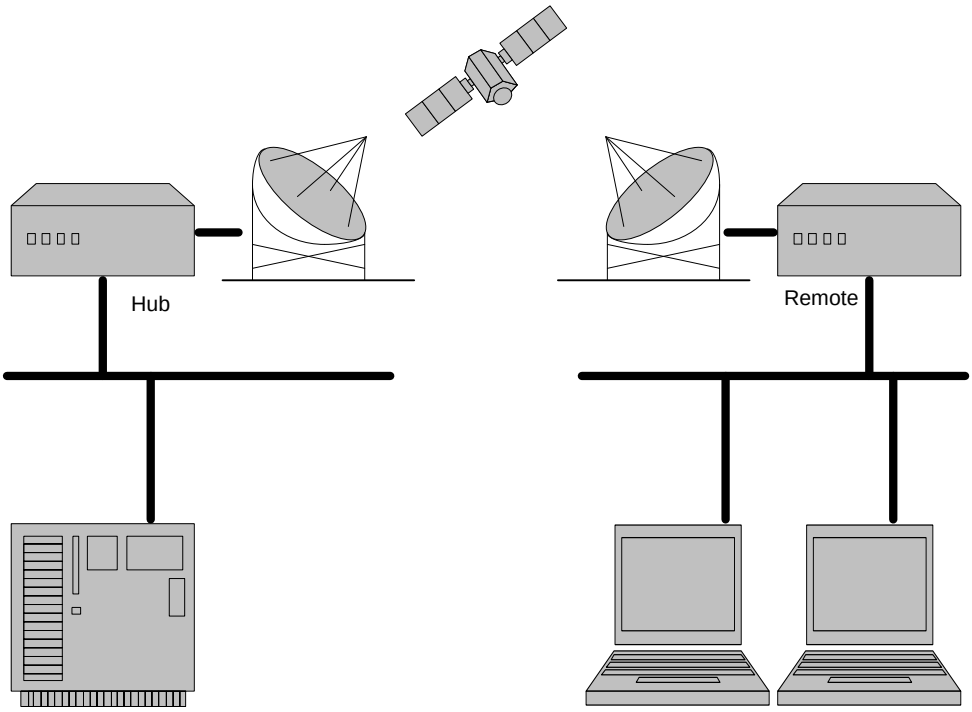


Figure D.1 Host, Hub, Dish – Satellite – Dish, Remote, Terminal

Satellite communications are used to tie together digital devices (e.g., terminals and hosts, voice over IP, etc.) under conditions where physical connectivity is not available. Digital data,

in the form of IP packets, arrive at both the Hub and the Remote stations from the outside world, where they are converted through a series of steps into an encryption-secured stream of analog symbols which are beamed up to a communications satellite. The satellite acts as a "bent pipe" which transmits that same stream of analog symbols back down to the antenna on the other end. Once there, the stream of analog symbols goes through an inverse series of steps to reconstruct the original IP packets, which are then handed off to the outside world for final delivery.

This STANAG is focused on those steps by which the data are converted, both from digital to analog and back from analog into digital format, and also the intertwined steps which assure both the integrity and the security of the data during its transmission and reception. Referring to Figure D.1, all of the steps covered by this STANAG will occur within the boxes labeled "Hub" and "Remote".

D.2 (NU) Overview of Data Paths

There are four different but related Data Paths associated with satellite communications. Two of them are sending data from the Hub to the Remote(s); this is also known as the Downstream. The other two are sending data from the Remote(s) to the Hub; this is also known as the Upstream. For both the Downstream and the Upstream, one of the Data Paths takes place inside the Hub, while the other takes place inside the Remote.

In Figure D.2, the Downstream Hub-side and Downstream Remote-side Data Paths are labeled "A" and "B", respectively. Likewise, the Upstream Remote-side and Upstream Hub-side Data Paths are labeled "C" and "D", respectfully.

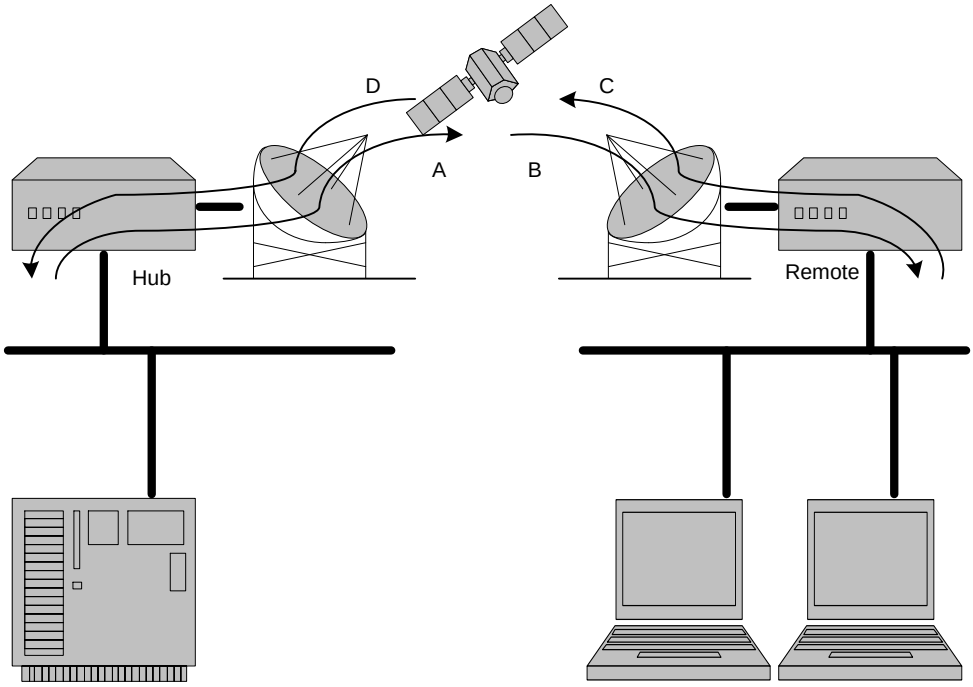


Figure D.2 Upstream & Downstream Data Paths

The Downstream Hub to satellite Data Path and the Downstream satellite to Remote Data Path are almost, but not exactly, mirror images (inverses) of each other. The same is true for the Upstream Remote to satellite Data Path and the Upstream satellite to Hub Data Stream. Likewise, the Downstream Hub to satellite Data Path and the Upstream Remote to satellite Data Path are nearly, but not exactly, parallel to each other. And again, the same is true for the Downstream satellite to Remote Data Path when compared with the Upstream satellite to Hub data path. Accordingly, each Data Path will be discussed separately throughout this document.

These four Data Paths are the heart of the standardization covered by this STANAG. They are described in detail, step by step, in Annex H. Section H.1 of that Annex contains an overview of each of the four Data Paths, listing all of the individual steps of each Data Path. The next four sections of Annex H walk through the details of the necessary actions at each step, along each Data Path. Finally, section H.6 contains descriptions and information that are common to all four Data Paths.

D.3 (NU) Internal Exchanges Between Hub and Remotes

In addition to acquiring, transmitting, receiving, and sending on the data which is received from / sent to the outside world, a Hub and its Remotes will also be transferring data between themselves.

There are five types of data which need to be transferred between the Hub and the Remote, as briefly described below.

The first is a Timeplan, which tells each Remote which slots it may transmit into, where a slot is defined as being a particular time slice along a particular InRoute frequency, within the approximately125ms that constitutes a time frame.

A large number of physical world factors – rain, temperature, satellite drift, etc. – can cause a Remote's transmission parameters to deviate from the ideal (or for the ideal to change from the parameters that had been appropriate). The Hub can instruct each remote, individually, and on a per-InRoute frequency basis, to make minor adjustments to how it transmits on that frequency. In particular, the parameters which can be changed are: transmission power, symbol offset, and frequency offset.

Every Remote which is a member of a common InRoute Group shares common characteristics which are defined across all the InRoutes of that InRoute Group. Periodically, the Hub sends a reaffirmation of these characteristics to every one of its Remotes. The information sent out consists of the InRoute Group ID, a collection of pairs of InRoute IDs and the Frequency of each such InRoute, and a three coordinate definition of the location of the satellite.

Each Remote monitors its own internal queues of data waiting to be sent to the Hub, and counts the number of slots it would like to burst transmit into, in order to send that data. It conveys this request for slots to the Hub by prepending a Demand Header to the front of every TDMA burst it transmits. Note that the number of desired slots is likely to change, even over the short time from one TDMA burst to the next (within the same Time Frame); thus the contents of this Demand Header will not remain the same from slot to slot. The Demand Header can also be used to request slots at different priorities.

The final required data exchange between Hub and Remote is the periodic updating of TRANSEC keys.

D.4 (NU) Activities Outside the Scope of this STANAG

- Registration of a Remote at its Hub: definition procedures for telling the Hub about the existence and parameters of a new Remote, prior to any attempts to actually acquire that Remote into the network

- Pre-Site Preparation before a Remote can become productive.
- Antenna Pointing as part of physically readying a Remote.
- Determining of a Remote's Transmit Initial Power
- The means by which the Hub monitors the accuracy with which each Remote adheres to the criteria for broadcasting into each InRoute
- How a Remote determines the amount of bandwidth it needs, and how it translates that into the number of slots it will ask the Hub for
- How the Hub determines the distribution of available bandwidth (slots per InRoute) among all of its Remotes
- How the Hub responds to claims that a given Remote's data should be provided a higher priority
- Other exchanges between the Hub and a Remote, besides the ones described herein
- How the Hub partitions InRoutes among various InRoute Groups
- How the Hub determines the frame size for fragments in an InRoute Group
- How the Hub assigns a FEC rate to an InRoute Group
- How the Hub selects which FEC rate it will use for its transmissions
- How the Hub assigns an ID to an InRoute
- How the Hub calculates the number of time slots in a timeframe, for a given InRoute Group
- Selecting the number of downstream FEC frames to transmit before inserting the next Unique Word
- Level 3 (and above) addressing and delivery of packets
- Shifting between IF and RF, and similar steps outside of the modem, to and from the antenna

- How the Hub separates the reception of data from each individual Remote out of all the time slots in all the InRoutes that each individual Remote could be using for data transmission
- Whether a Hub can assign different transmission priorities to different packets, and if so how it makes those choices
- Whether a Remote can assign different transmission priorities to different packets, and if so how it makes those choices
- Creation and maintenance of a Hub-resident routing table for determining to which Remote an incoming packet needs to be sent
- TCP Acceleration
- Whether the Hub performs Packet level encryption on none, all, or some of its packets; and, if some, how it determines which packets will receive the encryption
- Whether the Remote performs Packet level encryption on none, all, or some of its packets; and, if some, how it determines which packets will receive the encryption
- Whether or not the Hub performs Data Segmentation when transmitting to a specific Remote
- The selection of which bit mapping schema will be used is determined during network installation
- The decision of whether or not a Remote and the Hub will use VLAN is determined during network installation.
- When VLAN is in use, the assignment of VLAN Identifiers and Priority Code Points.
- The selection of the Intermediate Frequency used at the Hub
- The selection of the Intermediate Frequency used at a Remote
- The length of Guard Times used between burst transmissions by a Remote

ANNEX E
LINK ESTABLISHMENT

CONTENTS

E.1 (NU) Overview

This Annex covers the steps required in the establishment of a link between an in-operation Hub and a coming-on-line Remote.

E.1.1 (NU) Outside the Scope for this Document

There are several actions which must take place prior to a given Remote's coming on-line. Mostly these are associated with the establishment and definition of an entire network (Hub plus its Remotes). As such, they fall outside the scope of this document. The most notable such required but out-of-scope steps are listed here.

E.1.1.1 (NU) Registering of Remote at Hub

Link establishment is controlled from the Hub. Each Remote that may potentially join a network is registered at the Hub via a Network Management Function. The Hub establishes

and controls which Remotes may attempt to establish a link on a particular network and when they may attempt to do so.

When a Remote is registered with the Hub, the Network Operator selects a particular InRoute Group for the remote to belong to. An InRoute Group consists of a set of one or more InRoutes. All InRoutes within an InRoute Group are homogeneous with respect to signal rate, modulation, and FEC scheme. The InRoute Group that a Remote belongs to determines the set of possible InRoutes that a Remote may potentially acquire into as described in Section E.2.

Also at the time that a Remote is registered with the Hub, a decision is made as to whether or not that Hub-Remote link will support Virtual Local Area Network (VLAN) Tagging. If VLAN Tagging is supported, then the data payload of each transmitted block will be reduced in size by two bytes, which will be used instead for transmitting the VLAN information.

E.1.1.2 (NU) Pre-Site Preparation

The physical steps of this preparation may include providing power to the Remote, establishing the location for the outdoor unit (Antenna, BUC, LNB), establishing the RF connections between the indoor unit (Remote Router) and the outdoor unit, and connecting the Remote to the networking infrastructure (typically a LAN) present at the site.

E.1.1.3 (NU) Antenna Pointing

This step requires that all the pre-site preparation work be completed. It typically requires coordination with the Network provider. This step establishes that the outdoor unit is correctly pointed at the satellite and that the antenna has been finely tuned for optimum operation with respect to receiving and transmitting RF signals.

E.1.1.4 (NU) TX Initial Power

This step requires the determination of the correct initial operating transmit power for the Remote. This initial power setting is used by the remote while acquiring into the network as described in Section E.2. This value is typically established by calculation based on the parameters of a particular Remote installation such as BUC and Antenna size.

E.1.2 (NU) Network Byte Ordering

Most types of data – floating point numbers, strings, large integers, etc. – require several bytes for their storage. The order in which these bytes are strung together (in memory, on hard disk, etc.) is dependent on the computer being used. If both the computer installed as the Hub, and the computer installed as one of its remotes, have a different – opposite – ordering of the bytes used to store these data, then the data transmitted by one of them will not be recognized on the other side.

To prevent this problem from occurring, all transmissions and receptions of Hub-to-Remote and Remote-to-Hub messages, plus all headers added to data packets on one side and read then removed on the other side, will conform to a network standard of sending the most significant byte first, then proceeding in order to the least significant byte. This is frequently referred to as “Big Endian”.

E.1.3 (NU) Remote Acquisition

When a Remote has been off-line, it must be *acquired* into the network, that is, it must be brought into synchronization with the hub with respect to network timing, transmit frequency offset, and acceptable power settings in order for the Hub to begin including it in the mix of upstream slots. As an initial step in this process, the Remote uses its stored configuration tables to determine: that its antenna is pointed properly to listen to the satellite; what downstream symbol rate at which the Hub will be sending; and the frequency center for the downstream frequency at which the satellite will be transmitting. The remote uses these configuration parameters to begin listening to the satellite, and thus to the Hub. In particular, it is waiting to receive, in clear text, a reiteration from the Hub of the Common Characteristics for the InRoute Group (as described in Annex F, Section F.1.4). This will tell the Remote what all of the InRoute frequency centers are, for the InRoute Group to which it belongs.

With this information in hand, the Remote now waits for the Hub to invite this particular remote to join the network. Each upstream timeframe contains, following all the defined traffic slots, an acquisition slot during which the only permitted activity is for a Remote that has been invited by the Hub to send a request for Link Establishment. These acquisition slots, one per InRoute, are assigned, in rotation, as part of the Timeplan, each to an individual Remote which is not currently active in the network. Once a Remote uses the Acquisition Slot assigned to it and is successfully detected by the Hub, the Hub then conducts a synchronization exchange with that Remote, until the new Remote is adequately in step with the rest of the network. At that point the Hub’s broadcast Timeplan commences to include traffic slots for the new Remote, and that acquisition slot becomes available for the acquisition invitation to a different off-line Remote.

E.2 (NU) Remote Acquisition Details

This section describes the individual steps that need to be performed, on both the Remote and the Hub sides, in order to bring a previously off-line Remote into active status.

E.2.1 (NU) Listening for Remote Acquisition Slot

The Timeplan broadcast by the Hub to all its Remotes (see Figure F.1 in Annex F) covers a ~125ms timeframe, and is to be used by all Remotes within an InRoute Group. As explained in detail in Annex F, “Link Management”, this Timeplan contains a variety of data, including directives to each individual Remote, plus the slots – frequency combined with time offset – that each Remote is to use for its upstream transmission during that timeframe. At the end of the timeframe, the Hub also defines an Acquisition slot, one for each InRoute, so that a Remote which is waiting for Link Establishment can transmit its acquisition burst into that slot.

The Acquisition slot is between 1.5 and 2.0 times the normal length of an upstream slot, and constitutes the entire time between the end of the last “data” slot and the end of this 125ms timeframe.

An off-line Remote which needs to establish a link with its Hub must listen for the broadcast Timeplan; use that broadcast to synchronize its internal timing with the 125ms time frames for uploading; wait for the Hub to transmit an Acquisition Invitation message for its particular Remote ID, and then transmit an Acquisition Response block into the Acquisition slot.

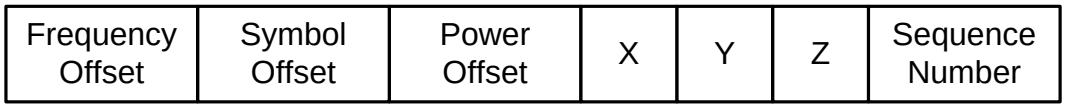


Figure E.1. Acquisition Response Block

IX	NAME	SIZE bytes	XDR	USE
1	Frequency Offset	4	long	most recent InRoute freq. adjustment
2	Power Offset	4	short	most recent signal gain adjustment
3	Symbol Offset	4	short	most recent burst start time adjustment
4	X	4	float	X-axis location of satellite
5	Y	4	float	Y-axis location of satellite
6	Z	4	float	Z-axis location of satellite
7	Seq. Num.	4	uns short	of most recently received timeplan

Table E.1 Composition of Acquisition Response Block

1. Frequency Offset: when received from the Hub in a timplan, this field's value, positive or negative, directed the Remote to adjust the baseline for this InRoute's frequency. The field is calibrated in units of Hz. The Remote will copy this number, from the latest timeplan it has received, so that the Hub can verify that the acquiring Remote has received and acted upon the Hub's most recent adjustment directive.

2. Power Offset: when received from the Hub in a timeplan, this field's value, positive or negative, directed the Remote to adjust its gain. The field is calibrated in units of dBm. The Remote will copy this number, from the latest timeplan it has received, so that the Hub can verify that the acquiring Remote has received and acted upon the Hub's most recent adjustment directive.

3. Symbol Offset: when received from the Hub in a timeplan, this field's value, positive or negative, directed the Remote to adjust its definition of the beginning of the acquisition frame. The field is calibrated in units of UpStream symbols. The Remote will copy this number, from the latest timeplan it has received, so that the Hub can verify that the acquiring Remote has received and acted upon the Hub's most recent adjustment directive.

4. X-axis Location of Satellite: This is a value which identifies the location of the Satellite along the X-axis in units of kilometers, from an origin at the center of the earth.

5. Y-axis Location of Satellite: This is a value which identifies the location of the Satellite along the Y-axis in units of kilometers, from an origin at the center of the earth.

6. Z-axis Location of Satellite: This is a value which identifies the location of the Satellite along the Z-axis in units of kilometers, from an origin at the center of the earth.

7. Sequence Number: The Hub will increment this number, rolling over when applicable, for each IRIB that it constructs (see Annex F, "Link Maintenance"). The Remote will copy this number, from the latest timeplan it has received, so that the Hub can verify that the acquiring Remote has received and acted upon the Hub's most recent adjustment directives.

E.2.2 (NU) Synchronizing

Once the Hub has received the coming on-line Remote's Acquisition Response block, it begins to include that Remote in its broadcast Timeplan, but still in the clear text, not the TRANSEC encrypted, partition. At this point it does not assign any data slots to the new Remote. Instead, it uses the normal directives format (see Figure F.1 and Table F.2) to issue commands to the new Remote, directing it to fine tune its transmission parameters, and thus adjust how perfectly the Remote's broadcast – the Acquisition Response block being used as a trial data block – fits into the Acquisition slot. This is an iterative process which will take several cycles of the Remote transmitting the Acquisition Response block, and then the Hub responding with adjustments.

At this time, the Hub also assigns a unique "Timeplan ID" to the new Remote. The mechanism for this is covered in detail in Annex F, "Link Maintenance", in section F.2, as part of the description of the information sent from the Hub to its Remotes. Hereafter, this new Remote will use this Timeplan ID to determine which from-Hub Timeplan data applies to it (and discard everything else).

E.2.3 (NU) Incorporating new Remote into non-Acquisition Slots

Once the Hub is satisfied with the accuracy of the new Remote's timing, frequency and power level, it will broadcast its next Timeplan with actual slots assigned to the new Remote (using the new Remote's Timeplan ID). However, because the new Remote is still not able to send or receive TRANSEC encrypted data, this is still sent in the clear text partition of the Timeplan. At this time the Remote ceases to use the Acquisition slot, and the Hub returns to listening for further new Remotes in that slot.

E.2.4 (NU) Initial TRANSEC Exchange

Please see Annex I, section I.6, "Initial TRANSEC Exchange" for the details of this step.

ANNEX F
LINK MAINTENANCE

CONTENTS

F.1 (NU) Overview

This Annex covers the steps required for maintaining a link between an in-operation Hub and an in-operation Remote.

Figure F.1 shows an expansion of an InRoute Information Block into its various components; and those components are then further expanded into their fields. Tables F.1, F.2, and F.3, and their accompanying text, then provide all the details as to the content and meaning of each field.

Figure F.2 shows an expansion of a Reiteration Block into its various components and fields. Table F.4, and its accompanying text, then provide all the details as to the content and meaning of each field.

Many of the fields included in these handshakes between Hub and Remote are encoded with the External Data Representation (XDR) schema. One characteristic of this schema is that all

data types (e.g., unsigned short) which are less than four bytes in length, are encoded into a four byte field for transmission; and then upon reception decoded back to their original data type and size. The tables in this section show, where applicable, what data type has been encoded into the four byte size of that data's field.

F.1.1 Network Byte Ordering

Most types of data – floating point numbers, strings, large integers, etc. – require several bytes for their storage. The order in which these bytes are strung together (in memory, on hard disk, etc.) is dependent on the computer being used. If both the computer installed as the Hub, and the computer installed as one of its remotes, have a different – opposite – ordering of the bytes used to store these data, then the data transmitted by one of them will not be recognized on the other side.

To prevent this problem from occurring, all transmissions and receptions of Hub-to-Remote and Remote-to-Hub messages, plus all headers added to data packets on one side and read then removed on the other side, will conform to a network standard of sending the most significant byte first, then proceeding in order to the least significant byte. This is frequently referred to as "Big Endian".

F.1.2 (NU) InRoute Information Block

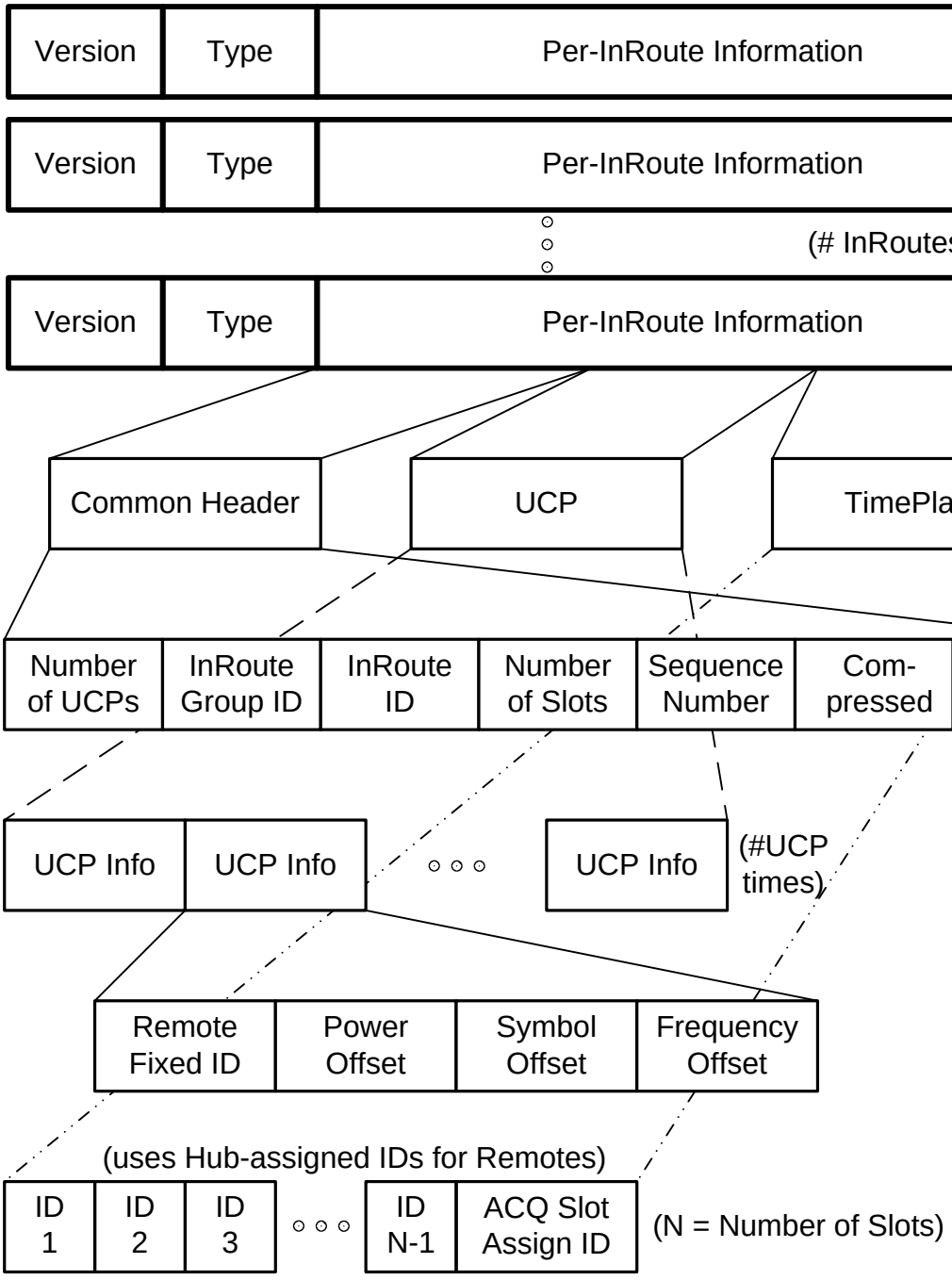


Figure F.1 Hub to Remotes: InRoute Information

The Hub broadcasts to all Remotes a series of InRoute Information Blocks (IRIBs). There is one such block for each InRoute controlled by the Hub. In Figure F.1, this is shown as the rows of fields at the top of the figure. After a Header, each InRoute's Information consists of: a Common Header (containing fields that are the same for all InRoutes, but whose field contents differ InRoute by InRoute); a set of Uplink Control Protocol (UCP) fields for use in minor adjustments of the Remote's antenna; and a Timeplan detailing to which Remote each timeslot (in the 125ms frame) of this frequency has been allocated. Overviews of the UCP and the Timeplan can be found in sections F.1.2 and F.1.3, below, respectively. Details on the Headers are given in section F.2; of the Timeplan in section F.3; and of the UCP in section F.4.

F.1.3 (NU) Minor, per-Remote Adjustments from Hub

The Hub will need to monitor the quality of each Remote's ability to adhere to the criteria defining the slots that the Hub has assigned to that Remote. The exact methodology and frequency of this monitoring is outside the scope of this document. If the Hub determines that a particular Remote has drifted away from its optimum settings, it will send a command, broadcast but specifically targeted to that one Remote, for the Remote to make an adjustment to the criterion (or criteria) which needs to be fine tuned.

In Figure F.1, the first Common Header field, Number of UCPs, defines how many groups of UCP information are included in this InRoute's Information. That number of UCP groups will be found immediately after the end of the Common Header. Every Remote will need to read all of these UCP groups, to see if any of the adjustment commands have been directed at it. The number of UCP groups within each IRIB is variable: there may be anywhere from zero to 16 UCP groups within one IRIB.

Within each UCP group there are four fields. The first identifies the Remote for which this information is targeted. The other three define the amount of minor adjustment which should be made for the three characteristics of Power, Symbol Start Time, and Frequency. Figure F.1 shows the UCP area placed immediately after the Common Header; initially expands the UCP area into a number of UCP groups; and then expands one of the UCP Groups into the four fields which make up each group.

F.1.4 (NU) Timeplan from Hub

The Hub sends a Timeplan, covering each next125ms timeframe, for each of its InRoutes, as part of its Downstream broadcast.. Under some circumstances, it will need to send the

Timeplan for an InRoute twice. The first is TRANSEC encrypted for security, and is used to convey minor adjustments and assigned data slots to each active Remote. The second is in clear text (as far as TRANSEC is concerned) and is used exclusively for the purpose of acquisition of a previously off-line Remote as it comes on-line. This second InRoute Timeplan, when present, will contain none of the information for existing Remotes, but only those data items which are needed by a being-acquired new Remote.

Each Remote must listen for the entirety of all Timeplans: the TRANSEC encrypted ones for active Remotes, and (when present) the TRANSEC clear text ones for Remotes requesting acquisition. Each Remote must then discard all timeplan information that does not apply to both its InRoute Group, and to itself as an individual Remote.

In Figure F.1, the Timeplan for a single frequency is shown at the very bottom of the figure. Each field in that row represents a time slot within the 125ms frame, for this InRoute's frequency. The number of such slots can be varied by the Hub, and is included in the Common Header so that the Remote will know how many to look at.

There is one additional field, which is only used when a new Remote is being acquired by the network. This field contains the Hub-assigned "Timeplan" ID for the new Remote (which otherwise would have no means of knowing which ID the Hub had assigned to it).

The Remote (when it isn't being acquired) scans the first N fields. Any field which contains that Remote's Timeplan ID represents a slot which that Remote has been allocated for burst transmission into. For any given InRoute, there may be anywhere from zero to all of the fields showing slots when that Remote may broadcast. [It should be noted that the Timeplan should never assign the same time slot, under two different InRoutes, to a single Remote.]

F.1.5 (NU) Common Characteristics Reiteration from Hub

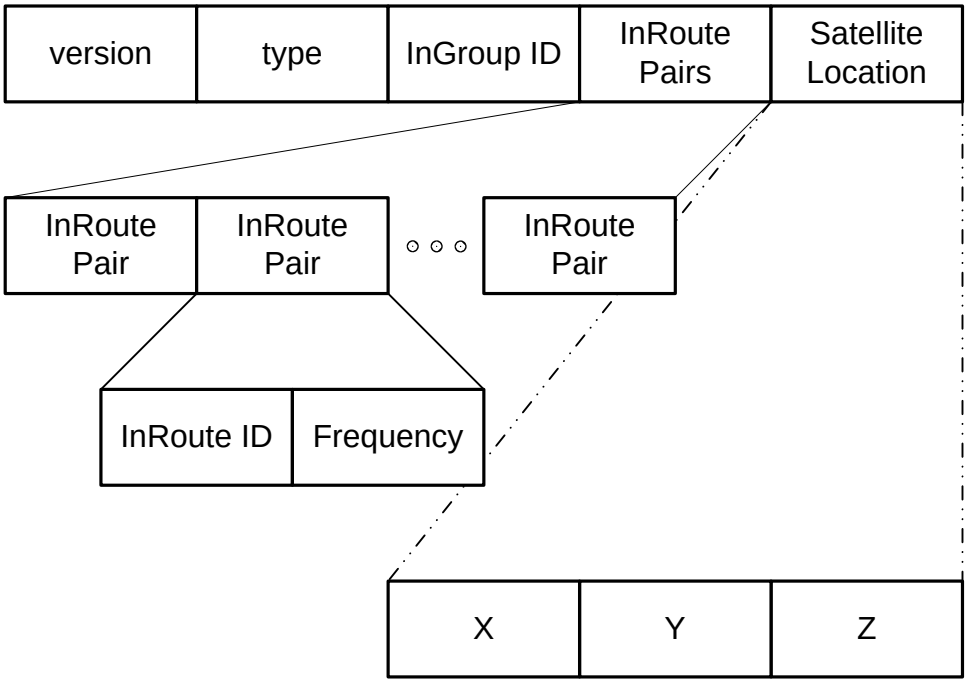


Figure F.2 Common Characteristics

Periodically, the Hub should send out a reiteration, for each InRoute Group, of the common characteristics that the Remotes within each InRoute Group share: the frequencies for each InRoute in the InRoute Group; plus the location of the Satellite.. Figure F.2 shows how this message is composed. See section F.5 for the details of the field contents. The determination of how often this message should be sent is outside the scope of this document.

F.1.6 (NU) Bandwidth Request from Remote

Each Remote should use some internal mechanism (outside the scope of this document) to monitor its own internal queues, and determine from that how much bandwidth to request that the Hub provide it. This request is included in every Time Slot burst. It should be noted that with a Timeframe of 125ms, there will be a 250ms lag between when a Remote asks for a change in bandwidth, and when the Hub can respond with a new Timeplan that takes that request into account. It should also be noted that the Hub will need some algorithm (outside the scope of this document) for how to distribute the finite resource of the available bandwidth, across multiple Remotes, when the total requested bandwidth from all Remotes exceeds the total available bandwidth.

Section F.6 shows the details for this request.

Included in this are a pair of schemas by which a Remote can request that the Hub treat its requests as a higher priority (under conditions where there are requests for more slots than the bandwidth can support). The exact circumstances under which a Remote will determine that some of its waiting data are deserving of having slot allocation expedited by the Hub is outside the scope of this document. Likewise, the algorithm by which the Hub will allocate slots, based on each Remote's normal and priority requests, is outside the scope of this document.

F.1.7 (NU) Updating of TRANSEC Keys

Please see Annex I, section I.7, "Updating of TRANSEC Keys", for the details of this step.

F.1.8 (NU) Provider Specific or Proprietary Messages

There may well be other messages which, while not necessary for the continued maintenance of the link between the Hub and any of its Remotes, are none the less useful at some other level. Such possibilities might include Status Reporting or Statistics Tracking. It is not the purpose of this STANAG to forbid or prevent such messages. So long as they are handled in the same manner as other internally generated data, and as long as they do not conflict with the message identification of the earlier listed required messages, any such other message handshakes between Hub and Remote are permitted; but are clearly outside the scope of this document.

F.2 (NU) InRoute Information Block

This section describes in detail the fields which make up the InRoute Information Block, along with the meaning of the contents of each field, and how their values should be determined (at the Hub) or interpreted (at the Remote).

The Hub will generate and send out a set of IRIBs, one for every InRoute it controls, every 125ms; interweaving this internally created data with the from-outside data which it is also sending, but giving any IRIB priority over other data. Each IRIB will be wrappered in a standard UDP packet and then fed into the Hub's transmission queues, and is thereafter treated no differently from other IP packets, except for having a higher priority.

Table F.1 lists all the fields in the InRoute Information Block, giving each field's size, name, and use, along with an indication of how many instances of the field will appear in any one IRIB. (The Index column is only for use for references within this document.)

IX	NAME	SIZE bytes	XDR	REPEAT	USE
1	Version	1		once	reserved for future compatibility
2	Type	1		once	identification of message type
3	UCP Count	4	uns short	once	how many UCP groups
4	IR Group ID	4	uns short	once	which IR Group is this
5	InRoute ID	4	uns short	once	which InRoute is this
6	Slot Count	4	uns short	once	how many slots in 125ms frame
7	Seq. Num.	4	uns short	once	for spotting missed timeplans
8	Compressed	4	bool	once	size of timeplan slots
9	Last	4	bool	once	set TRUE to show last InRoute
10	UCP Group	16		ix #3	commands to Remote
11	Timeplan	2 or 1		ix #6	shows slots allocated to Remote

Table F.1 Composition of InRoute Information Block

1. Version: this is a one byte field which is reserved for future use. The expectation is that it will be defined, in a future standard, for identifying components running different versions, so as to provide the possibility of forwards- and backwards- -compatibility. For the current implementation this byte should be set to zero.

2. Type: this is a one byte identifier used to distinguish between different types of messages. The value of Type for an InRoute Information Block is 220 (decimal).

3. UCP Count: for each InRoute, the Hub may issue Upstream Control Protocol commands to anywhere between zero and sixteen Remotes. This field holds the number of UCP Groups which are appended to the IRIB (index #10).

If the $H8000 bit of the field is set, then the first UCP Info group is not used to issue UCP commands, but instead is used to inform an off-line Remote that it may, during this Timeframe, burst into the Acquisition Timeslot as a request to be acquired. When the first UCP Info group is used in this manner, its first field, the “Remote Fixed ID”, is written as normally, but the other three fields are all left blank (zero). [Please note that when the $H8000 bit is set, this must occur after the XDR encoding of the original unsigned short data type of the UCP Count value. Likewise, the $H8000 bit must be examined, on the Remote side, prior to the XDR decoding of this field.]

If the $H4000 bit of the field is set, then the very last field in the IRIB, the ACQ Slot Assign ID, contains the Hub-generated Timeplan ID which the being-acquired Remote is to use, after acquisition is complete, for recognizing its allocated slots in future Timeplans. See Section F.3 for more details. [Please note that when the $H4000 bit is set, this must occur after the XDR encoding of the original unsigned short data type of the UCP Count value. Likewise, the $H4000 bit must be examined, on the Remote side, prior to the XDR decoding of this field.]

4. InRoute Group ID: This is an Identifier which the Hub has assigned to collect together a set of InRoutes (frequencies) for common use by a set of Remotes. A given Hub can have one or more InRoute Groups, and may assign its InRoutes among the InRoute Groups however it wants [the exact algorithm used is outside the scope of this document]. The only constraint is that a single InRoute may not belong to more than one InRoute Group.

Once assigned, an InRoute Group ID does not change. Also, once assigned to an InRoute Group, and InRoute is never reassigned to a different InRoute Group.

5. InRoute ID: This is an Identifier which the Hub has assigned to a particular InRoute – a particular Upstream frequency. The exact algorithm used to select an ID number is outside the scope of this document. The only constraint is that a single InRoute ID may only be assigned to a single InRoute, across all of the InRoutes under the control of this Hub.

Once assigned, an InRoute ID does not change.

6. Slot Count: for each InRoute, the Hub calculates the number of slots which will be available during a ~125ms frame, including the Acquisition Slot. This is also the number of Timeplan fields appended to the IRIB (index #11). This number is stored in this field.

Note: this calculation must be the same for all InRoutes of a single InRoute Group. The exact algorithm used for this calculation is outside the scope of this document.

7. Sequence Number: The Hub will increment this number, rolling over when applicable, for each IRIB that it constructs. The Remote will use these numbers to detect events wherein they have missed an InRoute Information Block. [There is no recovery procedure beyond the

Remote simply waiting to read that InRoute's IRIB during the next 125ms frame. However, statistics collecting and status reporting procedures, if any, may want to track these events.]

8. Compressed: If this field is set to True, this means that there are less than 256 Remotes belonging to this Hub. Under those conditions, the slot fields in the Timeplan can be compressed from two bytes each down to one byte each. If this value is False, then the slot fields remain at their default two byte size.

9. Last: This field normally has a value of False. The last IRIB broadcast by the Hub, in each 125ms frame, will have this field set to True. Each Remote will use this flag to tell it that there are no more InRoute Information Blocks (until the next frame starts).

10. UCP Group: This portion of the InRoute Information Block is covered in section F.3, Minor per-Remote Adjustments from Hub.

11. Timeplan: This portion of the InRoute Information Block is covered in section F.4, Timeplan from Hub.

F.3 (NU) Minor, per-Remote Adjustments from Hub

This section describes in detail the means by which a Hub directs one of its Remotes to make minor adjustments to one or more of the controllable characteristics of the Remote's antenna. These directives are collectively referred to as the Upstream Control Protocol (UCP).

Table F.2 lists all the fields in the UCP Block, giving each field's size, name, and use. (The Index column is only for use for references within this document.)

IX	**NAME**	**SIZE bytes**	**XDR**	**USE**
1	Remote Fixed ID	4	uns int	Remote recognizes data which apply to it
2	Power Offset	4	short	adjust the gain applied to signal
3	Symbol Offset	4	short	adjust the start time of each burst
4	Frequency Offset	4	long	adjust the baseline of InRoute frequency

Table F.2 Composition of Upstream Control Protocol Block

1. Remote Fixed ID: Each Remote, when installed, is assigned a Fixed ID, and this ID is registered with the Hub. While reading each IRIB, a Remote must read all of the UCP Blocks broadcast from the Hub, and ignore all except those which are labeled, through the value of this four byte field, as being for it.

2. Power Offset: This field's value, positive or negative, directs the Remote to adjust its gain. The field is calibrated in units of dBm.

3. Symbol Offset: this field's value, positive or negative, directs the Remote to adjust its definition of the beginning of each 125ms frame. The field is calibrated in units of UpStream symbols.

4. Frequency Offset: this field's value, positive or negative, directs the Remote to adjust the baseline for this InRoute's frequency. The field is calibrated in units of Hz.

F.4 (NU) Timeplan from Hub

This section describes in detail the means by which a Hub tells all of its Remotes which of those Remotes have been allocated with which of this IRIB's frequency's timeslots.

Table F.3 lists all the fields in the Timeplan, giving each field's size, name, and use, along with an indication of how many instances of the field will appear in any one Timeplan. (The Index column is only for use for references within this document.)

INDEX	NAME	SIZE (bytes)	REPEAT	USE
1	Timeplan ID	2 or 1	#slots	slot allocated to this Remote
2	ACQ Slot ID	2 or 1	once	for newly-acquired Remote

Table F.3 Composition of Timeplan

1. Timeplan ID: There is one field of this type for each upstream slot in a 125ms frame. The cardinality of the field's position in the Timeplan corresponds exactly with the cardinality of the slot in the frame: first field with first slot, second field with second slot, etc. The content of each field is the Timeplan ID of the Remote which has been allocated use of this slot during this frame. (Recall that this Timeplan is part of a particular IRIB, and thus the Remote already knows with which frequency these slots are associated.)

The use of a Timeplan ID, assigned by the Hub as each Remote is acquired, allows for multiple space savings. The Remote's Fixed ID is a four byte value, while the Timeplan ID, requiring uniqueness only across a single Hub's associated Remotes, needs to be only two bytes in size.

If, additionally, a given Hub has less than 256 remotes defined as belonging to it, then the Timeplan ID needs to be only one byte in size. This possible additional savings in size is supported through the use of the Compressed flag (see Table F.1, index #8). When that flag is set, each Timeplan ID field is only one byte in size.

2. ACQ Slot ID: When a new Remote is successfully acquired by the Hub, the Hub assigns to it a Timeplan ID, for the purposes described above. Since this is a dynamic assignment, the Hub must communicate this information to the Remote. This field is used to convey the Timeplan ID to this newly-acquired Remote. The Remote, which has been reading an in-clear text (per TRANSEC) IRIB, will notice that the UCP Count field (see Table F.1, index #3) has its $H4000 bit set: this flag is used to tell this Remote that it should look in the ACQ Slot ID field to find its Timeplan ID.

F.5 Common Characteristics Reiteration from Hub

This section describes the means by which a Hub periodically reaffirms, or, sometimes, changes, the characteristics which are common across all the Remotes of a given InRoute Group.

Table F.4 lists all the fields in the Reiteration, giving each field's size, name, and use, along with an indication of how many instances of the field will appear in any one Reiteration. (The Index column is only for use for references within this document.)

IX	**NAME**	**SIZE bytes**	**XDR**	**REPEAT**	**USE**
1	Version	1		once	reserved for future compatibility
2	Type	1		once	identification of message type
3	IR Group ID	4	uns int	once	which IR Group is this
4	InRoute Pair	4 & 4	unsigned & double	32 pairs	InRoute ID & Frequency
5	X	4	float	once	X-axis location of satellite
6	Y	4	float	once	Y-axis location of satellite
7	Z	4	float	once	Z-axis location of satellite

Table F.4 Reiteration Block

1. Version: this is a one byte field which is reserved for future use. The expectation is that it will be defined, in a future standard, for identifying components running different versions, so as to provide the possibility of forwards- and backwards- -compatibility. For the current implementation this byte should be set to zero.

2. Type: this is a one byte identifier used to distinguish between different types of messages. The value of Type for a Reiteration Block is 64 (decimal).

3. InRoute Group ID: This is an Identifier which the Hub has assigned to collect together a set of InRoutes (frequencies) for common use by a set of Remotes. A given Hub can have one or more InRoute Groups, and may assign its InRoutes among the InRoute Groups however it wants [the exact algorithm used is outside the scope of this document]. The only constraint is that a single InRoute may not belong to more than one InRoute Group.

Once assigned, an InRoute Group ID does not change. Also, once assigned to an InRoute Group, and InRoute is never reassigned to a different InRoute Group.

4. InRoute Pair: This is a pair of fields, repeated 32 times, each consisting of a InRoute ID and then the Frequency defined for that InRoute. If a particular InRoute Group contains less than 32 InRoutes, the trailing pairs are set to zero. The Hub will never assign more than 32 InRoutes to any one InRoute Group.

The InRoute ID is an Identifier which the Hub has assigned to a particular InRoute – a particular Upstream frequency. The exact algorithm used to select an ID number is outside the scope of this document. The only constraint is that a single InRoute ID may only be assigned to a single InRoute, across all of the InRoutes under the control of this Hub.

Once assigned, an InRoute ID does not change.

The Frequency is a double-word value in units of Hz.

5. X-axis Location of Satellite: This is a value which identifies the location of the Satellite along the X-axis in units of kilometers, from an origin at the center of the earth.

6. Y-axis Location of Satellite: This is a value which identifies the location of the Satellite along the Y-axis in units of kilometers, from an origin at the center of the earth.

7. Z-axis Location of Satellite: This is a value which identifies the location of the Satellite along the Z-axis in units of kilometers, from an origin at the center of the earth.

F.6 Bandwidth Request from Remote

This section describes the means by which a Remote requests bandwidth – slots in the upcoming Timeplan – from the Hub.

At the beginning of every burst that a Remote transmits into an allocated slot (per the Timeplan), the Remote prepends a two byte Demand Header which contains its most current count of a need for slots in the upcoming Timeplan.

Figure F.3 shows a schematic of this Demand Header; while Table F.5 gives a detailed breakdown into the sizes and meanings of each of the fields.

Because each burst transmission into an upstream slot happens at a slightly later time than the previous transmission, the values in the Demand Header are likely to change with every burst. If nothing else changes, then the transmission of one burst into a slot in of itself reduces the Remote's queues by that one slot's worth of data, while additional IP packets coming in from the outside world will increase the size of the Remote's queues.

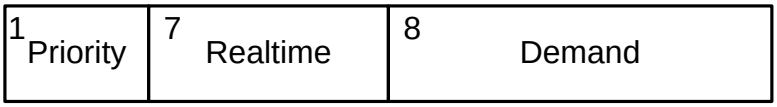


Figure F.3 Demand Header

INDEX	NAME	SIZE (bits)	USE
1	Priority	1	request expedited allocation of slots
2	Realtime	7	request for VoIP & similar applications
3	Demand	8	request for normal data

Table F.5 Demand Header

1. Priority: if this bit is set, then the Remote is requesting that the Hub expedite the allocation of the number of slots in Demand (Index #3).

2. Realtime: some data, for example Voice over IP, can suffer if it is not delivered in a timely manner. The Remote uses this field to tell the Hub how many slots worth of realtime data it has queued up. Ideally, the Hub will allocate slots such that realtime data has priority over normal data. The value of this field is divided by 128 (seven bits) to determine a percentage

(0 to 100). This is the percentage of the number of requested slots in the Demand (Index #3) field which are classified as being for realtime data.

3. Demand: the Remote uses this field to tell the Hub how many slots worth of data, of all types, it has queued up. The value in this field may range from zero to 255. However, this value has a non-linear meaning. Table F.6 shows the relationship between the value in this field and the number of slots requested.

VALUE RANGE	MULTIPLIER	REQUEST RANGE
0	none	none
1 – 63	single	1 – 63
64 – 127	double	128 – 254
128 – 255	quadruple	512 – 1020

Table F.6 Demand Definition Curve

F.7 Updating of TRANSEC Keys

Please see Annex I, section I.7, “Updating of TRANSEC Keys”, for the details of this step.

[This page intentionally left blank.]

ANNEX G
LINK TEARDOWN

CONTENTS

G.1 (NU) Overview

There is no explicit methodology by which a Remote may remove itself from the network, or by which a Hub may remove one of its Remotes. The assumption is that, once a Remote is successfully acquired by its Hub, then that Link stays up forever. This is, of course, not a valid assumption. This Annex briefly describes the two most common ways by which a Link may go down, and how the other components of the network detect and react to that event.

G.2 (NU) Loss of Remote

A Remote may go down because of some external event, such as loss of power. When this happens, the only means of detection which the Hub has, is that it is no longer receiving any data in the InRoute Group slots which it is assigning to that Remote in its Timeplan.

It is recommended that the Hub be assigned a timer value related to a timeout on waiting for new transmissions from a Remote. If the assigned time has elapsed, and no new transmissions have been received, then the Hub could remove that Remote entirely from consideration, when the Hub builds its Timeplan for the relevant InRoute Group.

Since this means that this Remote, even if it is still alive, will no longer have any way of transmitting to the Hub, the Hub will now need to begin periodically offering an Acquisition Slot to that Remote, as discussed in Annex E, “Link Establishment”.

Once that Remote does come back up, it will be able to use the standard Link Establishment procedures to regain access to the network, and establish a new Link with the Hub.

Notice that the Hub will assign a new, and likely different, Timeplan ID to this Remote, as part of that acquisition process.

G.3 (NU) Loss of Hub

Should the Hub itself go down, then when it comes back up all of its links to its Remotes will have been destroyed. At this point the Hub will need to cycle through its entire list of Remotes, offering Acquisition Slots to all of them. As there are limits to how many Remotes may be acquired simultaneously – in particular only one Remote may be acquired per InRoute, at any one time – there will likely be several rounds of acquisitions before the network is fully re-established.

The order in which the Hub offers acquisitions to its Remotes, and the time allotted before it gives up on one Remote and offers that Acquisition Slot to another Remote instead, are issues which are outside the scope of this document.

Eventually the Hub will attain a steady-state condition, in which either all of its assigned Remotes are again part of the network, or else there are still some Remotes which have not yet rejoined, and the Hub is cycling through them and offering each an Acquisition Slot.

ANNEX H
DATA TRANSFER

CONTENTS

H.1 (NU) Overview

This Annex describes the step-by-step process by which data is taken from the outside world, prepared for transmission to the satellite, received from the satellite, and presented once again to the outside world. There are four Data Paths that are followed, at different times, during this process:

- Data flowing from the Hub side to the Remote side; while transversing the Hub;
- Data flowing from the Hub side to the Remote side; while transversing the Remote;
- Data flowing from the Remote side to the Hub side; while transversing the Remote;
- Data flowing from the Remote side to the Hub side; while transversing the Hub;

While there are a number of similarities, parallels, and mirror images between these four Data Paths, there are also significant differences between them. Rather than describing one or two of them, and then saying that, “the others are a parallel or a mirror image (except for x, y, and z),” it is more appropriate to describe each Data Path in its entirety. This means that there will be some subsections which are word-for-word the same between the major sections; but this duplication will also prevent the reader from flipping back and forth, while trying to keep place in two parts of the document at once.

Where the content is sufficiently large, such that repetition is more of a distraction than an aid, that content has been pulled out of the individual Data Path sections, and placed in section H.6, “Common Details”.

H.1.1 (NU) Downstream: Hub to Remote, Hub side

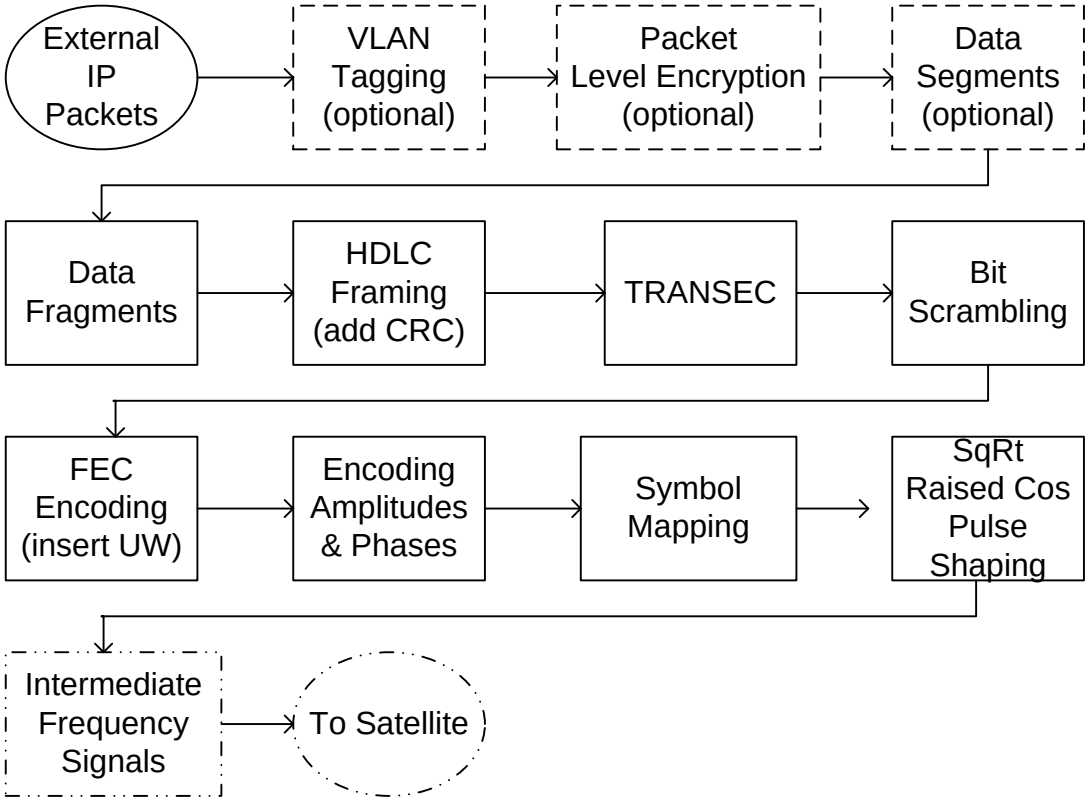


Figure H.1.1 Downstream, Hub-side

Figure H.1.1 displays the steps through which Hub-side originating data must pass, before they are ready for transmission to the satellite, on their way to their eventual destinations. The oval steps at either end of the chain represent additional activities (e.g., creation of IP packets, shifting from IF to RF for actual transmission, etc.) which are outside the scope of this document. The rectangular steps are those which are covered by this standard, and are each discussed in detail under Section H.2. Steps outlined in dot-dot-dash are ones where the data are in analog form; steps outlined by dotless borders – solid or dashed – are ones where the data are in digital form. The dashed boxes indicate steps which do not apply to all data packets.

IP packets arrive from the outside world, and go through a series of steps to turn them into a constant stream of analog symbols, which are beamed up to the satellite for broadcast to all Remotes assigned to this Hub. Other packets, generated internally by the Hub (see Annex

F), are treated in the same manner as the external packets. During these conversion steps, packets are split into manageable sized segments, encapsulated in protocols which minimize transmission errors and security breaches, and formed into a constant stream of digital bits. This stream is then converted from digital bits into analog symbols, which are sent to the satellite.

Figure H.1.2 depicts a simplified step-through of these conversions being applied to incoming data on the Hub side of the Downstream.

An incoming packet is (optionally) assigned an appropriate VLAN Identifier (VID), and a VLAN Header (VH) is prepended. It is then (also optionally) provided with packet-level encryption, and an Encryption Header (EH) is prepended. The next (also optional) step splits the packet – encrypted or not, but if encrypted then also with its header – into constant sized segments, each of which has a Segment Header (SH). This step allows Segments from different IP Packets to be intermingled: allowing smaller packets to not need to wait for the entire transmission of a very large preceding packet. The Block – in whichever state the choice of optional steps have left it in – is then split into fragments, each with its own Fragment Header (FH). Fragments may be of varying sizes, thus allowing exactly the correct, constant number of bits to be provided to the TRANSEC encryption routine, later on. HDLC Bit-stuffing is applied to this multi-fragment Block; a High-level Data Link Control (HDLC) Header is prepended, and the Header plus Payload have a Cyclic Redundancy Check (CRC) run on them and added as a trailer. The resulting TRANSEC Content is prepended with a TRANSEC Header. Bit-scrambling is applied to this entire Block. The resulting bit stream is split into fixed-size Forward Error Control (FEC) data chunks, and the FEC check values for each data chunk are calculated and appended. Every so often a Unique Word (UW) is inserted, to make synchronization on the Remote side, during reception, easier. This FEC bitstream is turned into a Symbol Stream; then into a stream of Intermediate Frequency (IF) signals, which are handed off to hardware for sending up to the satellite as a Downstream Burst.

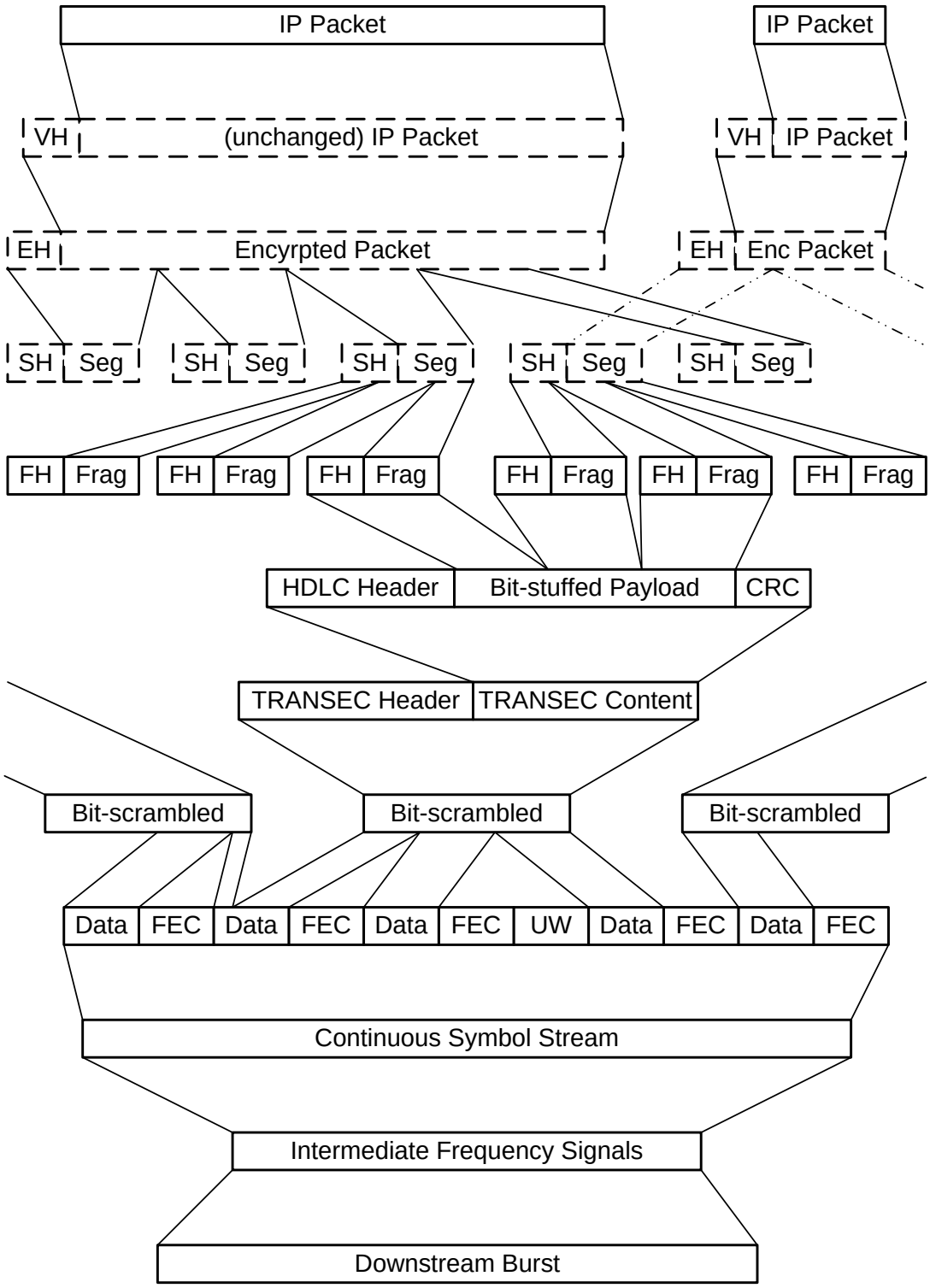


Figure H.1.2 Data Conversions, Downstream Hub Path

H.1.2 (NU) Downstream: Hub to Remote, Remote Side

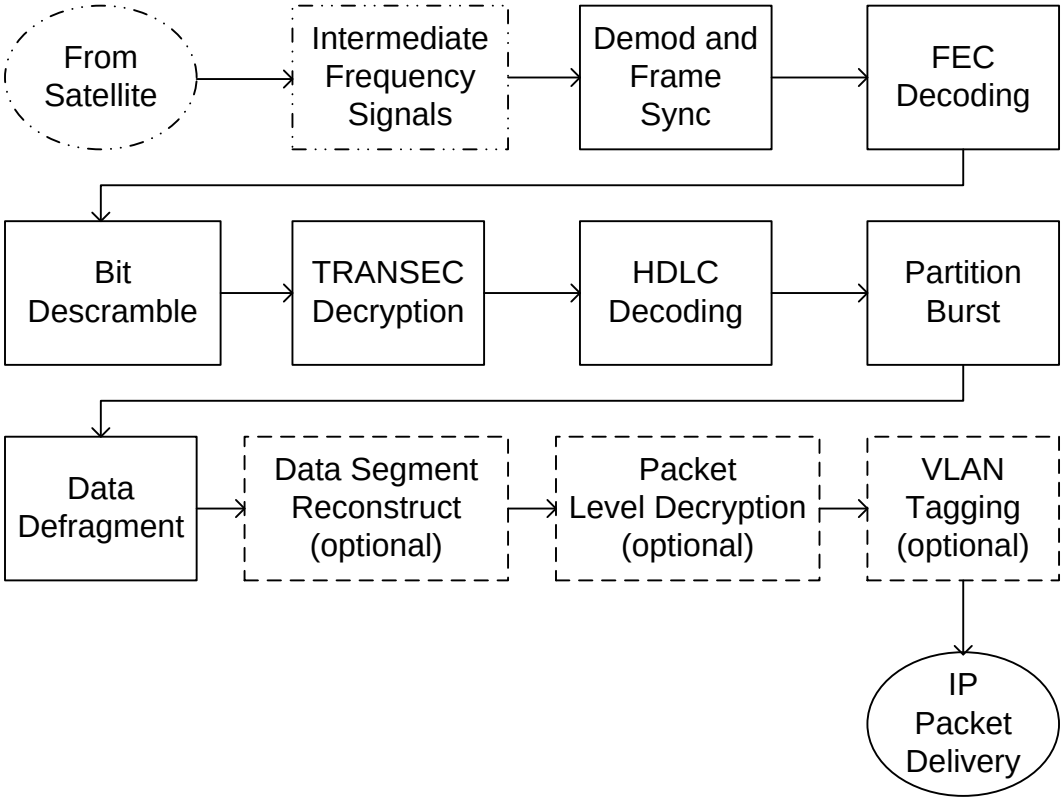


Figure H.1.3 Downstream, Remote-side

Figure H.1.3 displays the steps through which Hub-side originating data must pass, after they have been received from the satellite by the Remote, on their way to their eventual destinations. The oval steps at either end of the chain represent additional activities (e.g., shifting from RF to IF) which are outside the scope of this document. The rectangular steps are those which are covered by this standard, and are each discussed in detail under Section H.3. Steps outlined in dot-dot-dash are ones where the data are in analog form; steps outlined by dotless borders – solid or dashed – are ones where the data are in digital form. The dashed boxes indicate steps which do not apply to all data packets.

The Remote receives the data stream from the satellite and, through several steps, converts that from a stream of analog symbols into a stream of binary bits. The bit stream is then decapsualized to remove the wrappers used for transmission quality and network security.

The next step is to select, from the entire broadcast data stream, only those portions which have this individual Remote as their interim destination. Once the rest of the data stream has been discarded, the payload bits are recombined into their original packets and delivered to the outside world.

H.1.3 (NU) Upstream: Remote to Hub, Remote Side

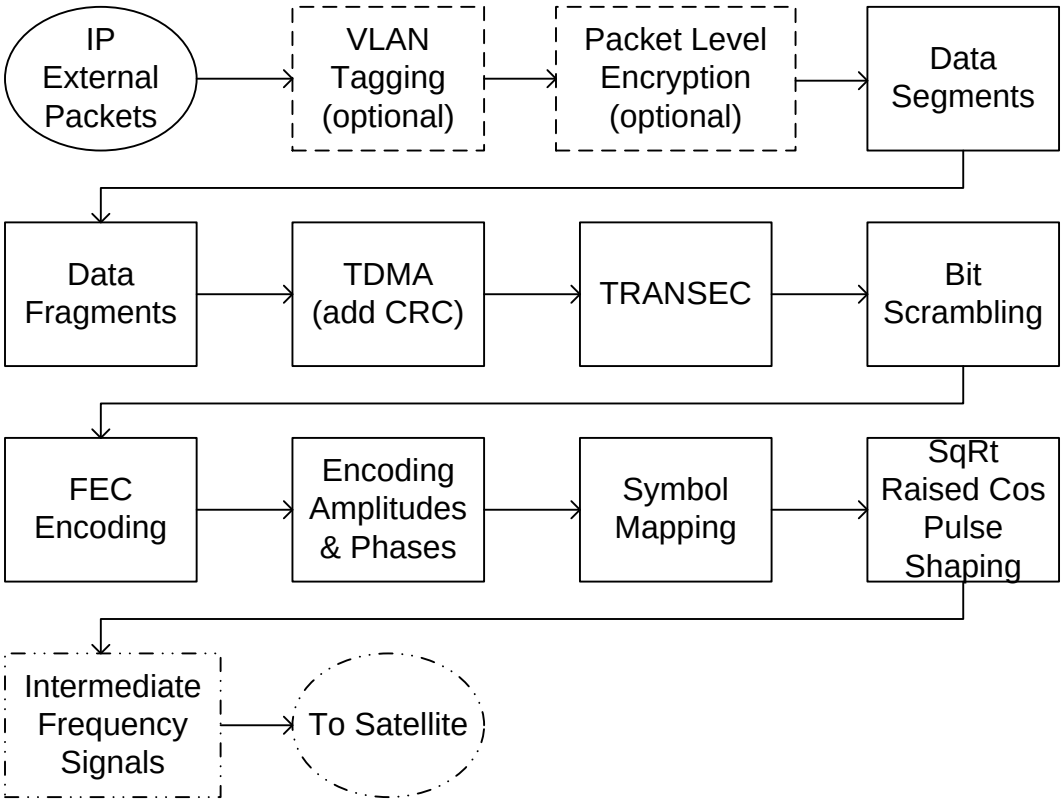


Figure H.1.4 Upstream, Remote-side

Figure H.1.4 displays the steps through which Remote-side originating data must pass, before they are ready for transmission to the satellite, on their way to their eventual destinations. The oval steps at either end of the chain represent additional activities (e.g., creation of IP packets; shifting from IF to RF for actual transmission, etc.) which are outside the scope of this document. The rectangular steps are those which are covered by this standard, and are each discussed in detail under Section H.4. Steps outlined in dot-dot-dash are ones where the data are in analog form; steps outlined by dotless borders – solid or dashed – are ones where the data are in digital form. The dashed boxes indicate steps which do not apply to all data packets.

IP packets arrive from the outside world, and go through a series of steps to turn them into a set of burst streams of analog symbols, which are beamed up (intermingled with all other Remotes' own bursts) to the satellite for delivery to this Remote's Hub. Other packets, generated internally by the Remote (see Annex F), are treated in the same manner as the external packets. During these conversion steps, packets are split into manageable sized segments, encapsulated in protocols which minimize transmission errors and security breaches, and formed into a constant stream of digital bits. This stream is then converted from digital bits into analog symbols, which are sent to the satellite.

Figure H.1.5 depicts a simplified step-through of these conversions being applied to incoming data on the Remote side of the Upstream.

An incoming packet is (optionally) assigned an appropriate VLAN Identifier (VID), and a VLAN Header (VH) is prepended. It is then (also optionally) provided with packet-level encryption, and an Encryption Header (EH) is prepended. The next step splits the packet – encrypted or not, but if encrypted then also with its header – into constant sized segments, each of which has a Segment Header (SH). This step allows Segments from different IP Packets to be intermingled: allowing smaller packets to not need to wait for the entire transmission of a very large preceding packet. The Block – in whichever state the choice of encryption has left it in – is then split into fragments, each with its own Fragment Header (FH). Fragments are of a fixed, pre-determined size (thus allowing exactly the correct, constant number of bits to be provided to the TRANSEC encryption routine at a later step). A Time Division Multiple Access (TDMA) Header is prepended; then Remote calculates a Demand Header (DH), requesting future bandwidth from the Hub, and this is also prepended, in front of the TDMA Header. Next the two Headers plus Payload have a Cyclic Redundancy check (CRC) run on them and added as a trailer. Because the DH, TDMA, and CRC are always the same size, the resulting Block has the correct, constant number of bits to be provided to the TRANSEC encryption routine; the resulting TRANSEC Content is prepended with a TRANSEC Header. Bit-scrambling is applied to this TRANSEC Block. The resulting bit stream is placed into a fixed-size Forward Error Control (FEC) data chunk, and the FEC check values for each data chunk are calculated and appended. A Unique Word (UW) is then prepended to make synchronization, during reception, easier. This FEC bitstream is turned into a Symbol Stream; then into a stream of Intermediate Frequency (IF) signals, which are handed off to hardware for sending up to the satellite as a burst into a particular time slot on a particular InRoute (frequency).

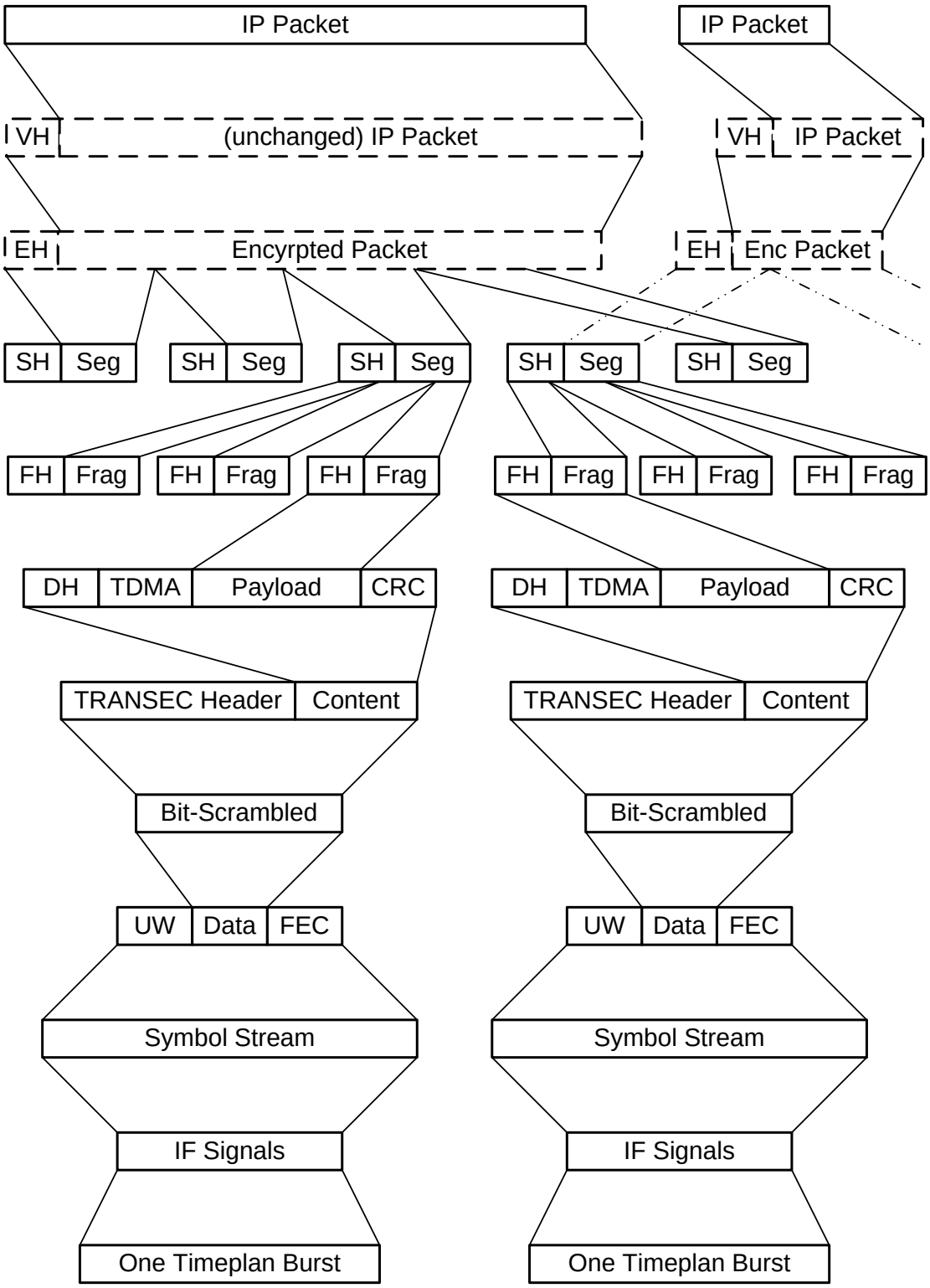


Figure H.1.5 Data Conversions, Upstream Remote Path

H.1.4 (NU) Upstream: Remote to Hub, Hub Side

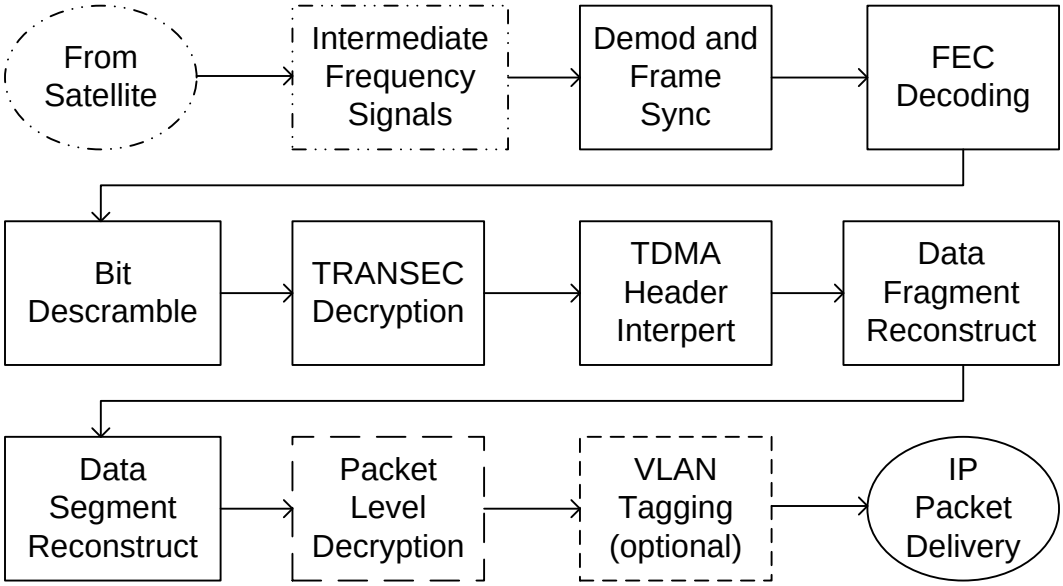


Figure H.1.6 Upstream, Hub-side

Figure H.1.6 displays the steps through which Remote-side originating data must pass, after they have been received from the satellite by the Hub, on their way to their eventual destinations. The oval steps at either end of the chain represent additional activities (e.g., shifting from RF to IF) which are outside the scope of this document. The rectangular steps are those which are covered by this standard, and are each discussed in detail under Section H.5. Steps outlined in dot-dot-dash are ones where the data are in analog form; steps outlined by dotless borders – solid or dashed – are ones where the data are in digital form. The dashed boxes indicate steps which do not apply to all data packets.

As the analog symbol stream is received from the satellite, the Hub uses its knowledge of the timeplan (see Annex F), which it had previously broadcast to its Remotes, to identify which slots – which time segments and frequencies – are being used by each Remote in this upstream data burst. The exact methods by which the Hub separates the data burst into its different frequencies, pulls apart the slots along each frequency, and combines multiple slots – potentially across multiple frequencies – so as to recreate and partition the bit stream from each Remote, is outside the scope of this document.

Once a Remote's data stream has been recovered and recombined, that bitstream is passed, step by step, through a series of inverse processes to decapsulize the data from its

transmission quality and security wrappers; and then recombine the parts of each IP packet together. The individual packets are then sent into the outside world.

H.1.5 (NU) Common Details

A few of the steps contain details that are both fairly long to relate, and also exactly the same for the parallel or mirrored steps in all four Data Paths. In these cases, the common details have been pulled out of the four steps in the four Data Paths, and placed instead into section H.6, “Common Details”.

H.2 (NU) Downstream: Hub to Remote, Hub Side

This section describes the necessary actions that the Hub must take at each step, from when it first receives a digital data packet (whether from the outside world or internally generated), until it hands off an Intermediate Frequency stream of analog symbols to the hardware connected to the antenna.

H.2.1 (NU) Packets

Packets may either arrive from the outside world, routed to this Hub as the next step in their path to their eventual destination; or they may be created internally, by the Hub, for delivery to its associated Remotes.

Different packets may be assigned differing levels of priority for transmission. How packets are identified and marked for each different priority level, and what the Hub does to honor those priority requests, are both outside the scope of this document.

H.2.1.1 (NU) External Packets (IP only)

All externally generated packets must be IPv4 packets complying with IETF RFC 791. Any non-IP data which arrives at the Hub will be discarded.

The destination IP address is looked up in a Routing Table to determine to which Remote this packet should be transmitted – that is, which Remote's ID should be attached to this packet. If there is no match for the destination IP address, then this packet is rejected.

The creation and maintenance of this Routing Table is outside the scope of this document.

IP packets come in a variety of flavors. For purposes of this STANAG, these flavors may be collapsed into only three categories: TCP packets (as per IETF RFC 793); UDP packets (as per IETF RFC 768); and all other IP packets.

H.2.1.1.1 (NU) TCP Packets

After this point, the Hub treats the entire TCP packet as a Data Block, thus indistinguishable from other IP packet types, as per Figure H.2.1.

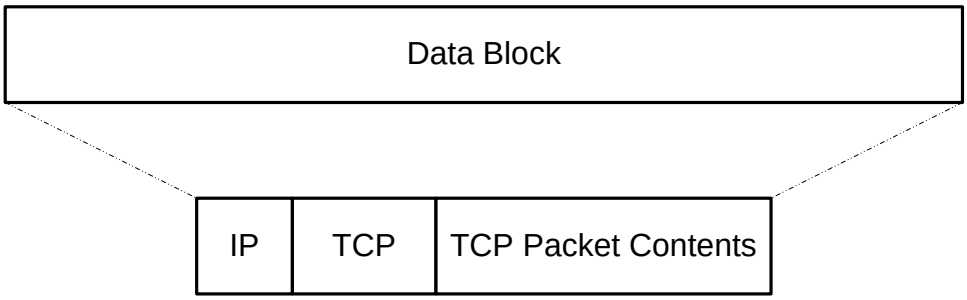


Figure H.2.1 TCP Packet to Data Block

H.2.1.1.2 (NU) UDP Packets

Because some UDP packets are also RTP packets, the Hub may want to assign a higher priority to the transmission of these packets. The exact mechanism by which the Hub prioritizes among its waiting-to-be-sent data is outside the scope of this document.

Otherwise, after this point, the Hub treats the entire UDP packet as a Data Block, thus indistinguishable from other IP packet types, as per Figure H.2.2.

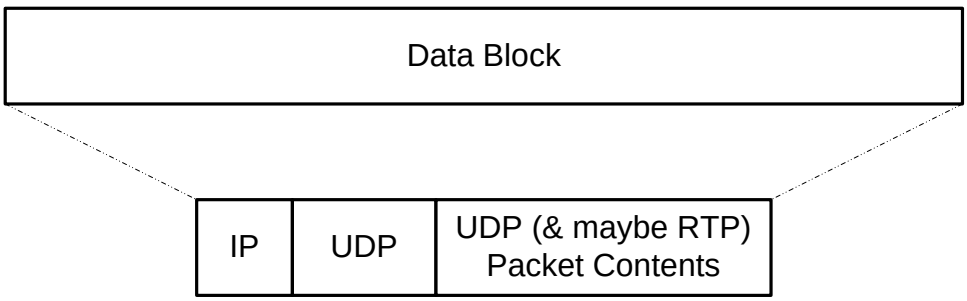


Figure H.2.2 UDP Packet to Data Block

H.2.1.1.3 (NU) Other IP Packets

If the IP packet is neither TCP nor UDP, then no special processing is possible. After this point, the Hub treats the entire IP packet as a Data Block, thus indistinguishable from other IP packet types, as per Figure H.2.3.

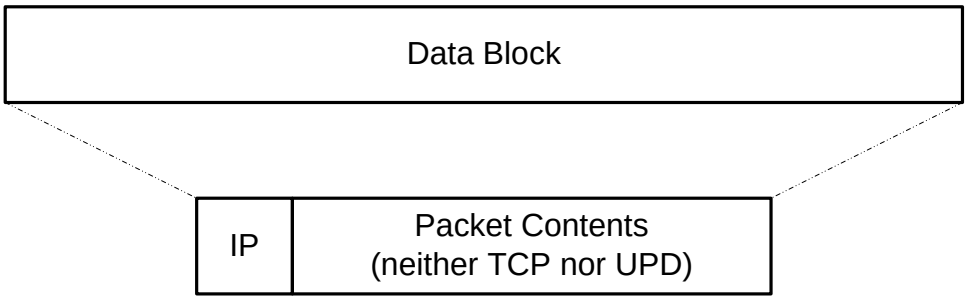


Figure H.2.3 IP Packet to Data Block

H.2.1.2 (NU) Internal Data Blocks

These are Blocks used to hold data which the Hub needs to send to its Remote. Examples include Timeplans and Minor Adjustments (see Annex F for the entire list).

It is recommended that internally generated Data Blocks, associated with maintaining the link between Hub and Remote, be given the highest priority for transmission. The exact mechanism by which the Hub prioritizes among its waiting-to-be-sent data is outside the scope of this document.

These internally generated Data Blocks are not provided with IP headers, since their destination is not outside of, beyond, the Remotes on the other side of the satellite hop. Because of this, they are treated in parallel with externally generated Data Blocks, but are kept separate for the next several steps. After the appropriate Link Level Header is appended to each internally generated Data Block (see section H.2.6), these Data Blocks will be indistinguishable from externally generated Data Blocks.

H.2.1.3 (NU) VLAN (optional)

Virtual Local Area Networks (VLAN) are a method of logically tying together nodes (hosts, terminals, printers, etc.) into a logical network, even when there is no physically continuous network connecting them. To bridge the physical gap between one Virtual LAN segment and

another, the VLAN is assigned a VLAN ID (VID), which is appended to all data blocks being transmitted between VLAN segments.

Whether or not a given Remote-Hub link will support VLAN tagging is determined at the time the Remote is registered at the Hub, and is thus outside the scope of this document. When a Remote-Hub link is defined as supporting VLAN tagging, every data block will have a VLAN Header (VH) appended to it, prior to all other processing.

VLAN tagging follows the IEEE 802.1d and 802.1q standards.

Figure H.2.4 shows the transformation of a Data Block by the prepending of a VLAN Header. Because there is no change to the data in the data block during this process; and because all subsequent steps will treat the VLAN Header as just another pair of Data Block bytes – up until the VLAN processing step at the Remote – the “result” of prepending the VLAN Header is also called a “Data Block”. Table H.2.1 depicts the contents of the VLAN Header in detail.

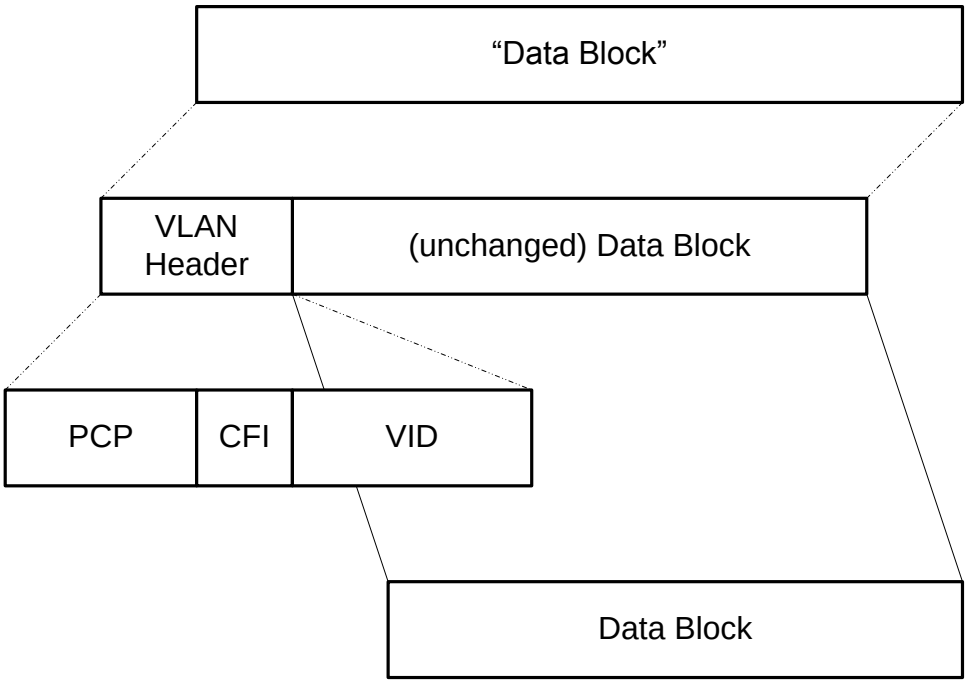


Figure H.2.4 VLAN Tagging

INDEX	NAME	SIZE	USE

1	PCP	3 bits	Priority Code Point
2	CFI	1 bit	Canonical Format Indicator
3	VID	12 bits	VLAN Identifier

Table H.2.1 VLAN Header Fields

1. PCP. Priority Code Point. Assignment of a priority to this data block, as per IEEE 802.1d. The determination of what priority to assign a given data block is outside the scope of this document. The implementation of methodologies to take advantage of those priorities is outside the scope of this document.

2. CFI. Canonical Format Indicator. As per IEEE 802.1q: if “0”, the Media Access Control (MAC) address is in canonical format (e.g., as for an Ethernet network); if “1”, the MAC is in non-canonical format (e.g., as for a Token Ring network). Determining which value to set this bit is outside the scope of this document.

3. VID. VLAN Identifier. Assignment of an identifier to allow physically separate parts of the same logical (virtual) network to be associated with each other by the intervening communications devices and protocols, as per IEEE 802.1q. The assignment and tracking of VIDs is outside the scope of this document.

H.2.2 (NU) Encryption at Packet Level (optional)

The Hub may perform an encryption on the entire incoming packet. Whether the Hub applies this step to all incoming packets, to none of the incoming packets, or only to selected ones, and in the latter case how the Hub selects which packets to encrypt, are controlled by methodologies which are outside the scope of this document.

Note that this is a separate and independent step from the TRANSEC encryption (which is not optional).

Packet level encryption shall use the AES 256 CFB algorithm.

Details of packet level encryption are presented in section H.6.4.

H.2.3 (NU) Data Segmentation (optional)

This is an optional step which is recommended due to the potential improvement in overall data transfer during periods of congestion. By breaking each Data Block down into smaller,

fixed size Segments, it is possible to apply a finer grain of control over the intermingling of large and small Data Blocks, and correspondingly improve the delivery for all data.

Downstream Data Segmentation is turned on or off as part of the Remote Configuration process, and will apply to all, or none, of the packets being transmitted by the Hub, on a Remote-by-Remote basis. The decision of which Remotes will, and which will not, be have Data Segmentation turned on, as well as the methodology by which the Hub determines which packets will have Data Segmentation applied to them, are outside the scope of this document.

Figure H.2.5 shows the transformation of a Data Block (or of an Packet Level Encrypted Data Block) into a series of Data Segments plus Segment Headers. Each Data Block is split into a series of smaller segments, and each segment is given a header for recombining, on the Remote side, the segments into the Data Block they came from. Table H.2.2 depicts the contents of the Segment Header in detail.

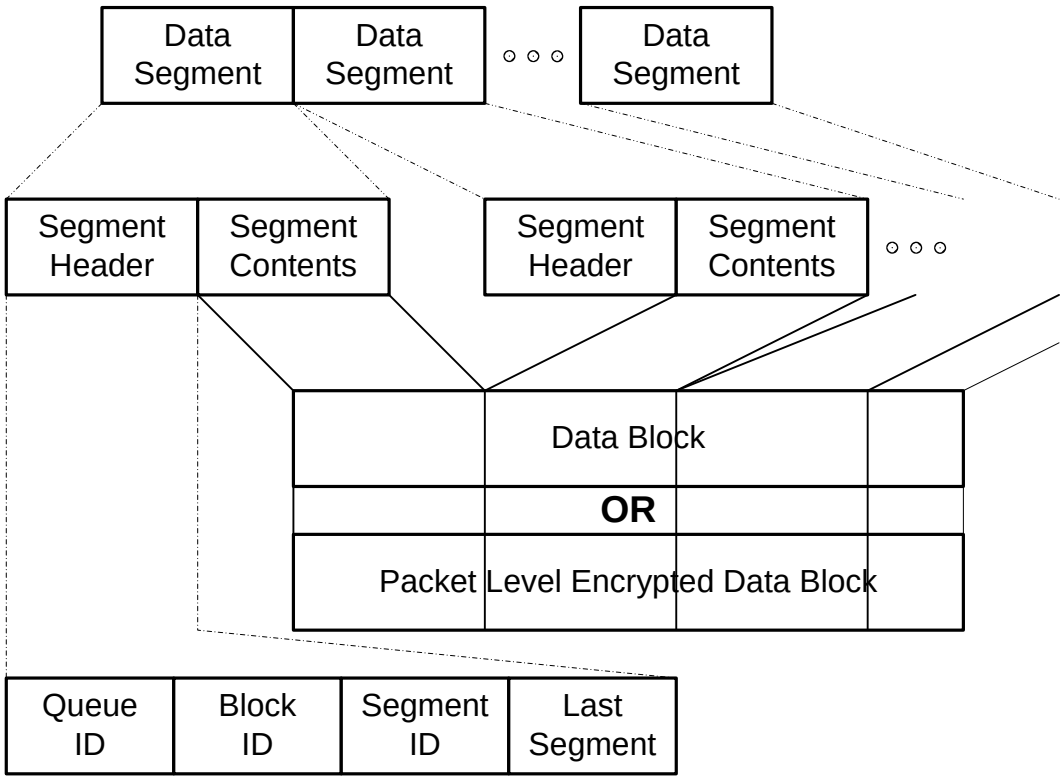


Figure H.2.5 Data Segmentation

INDEX	NAME	SIZE	USE
1	Queue ID	6 bits	identifies transmission queue for Block

2	Block ID	3 bits	common for all Segments of a Block
3	Segment ID	6 bits	sequential for Segments in a Block
4	Last Segment	1 bit	set to 1 if last Segment of this Block

Table H.2.2 Segment Header Fields

1 Queue ID. All of the Data Segments which make up a single IP packet will have been placed, in order on the same transmission queue. This field uniquely identifies that common queue. The Queue ID itself is meaningless on the reception side, but the uniqueness of this field's value assists in the reassembly of the entire packet.

2. Block ID. Rotating values of 0 through 7. All Segments which are part of the same Block will be set to the same Block ID. At the start of the next Block, this value will be incremented, rolling over from 7 to 0.

3. Segment ID. Starts at 0 for the first Segment of each Block. Each subsequent Segment in the same Block is assigned the next sequential number as its Segment ID.

4. Last Segment. Set to 0 if there are more Segments which make up this Block. Set to 1 if this is the last Segment making up this Block.

Data Segments are of a constant size (thus there is no need for a length indicator in the Segment Header). This size varies depending on the MODCOM in use; and in particular with the FEC Rate that is being used. Table H.2.3 shows the relationship between the FEC Rate and the segment size.

FEC RATES	0.431	0.533	0.495	0.793	0.879
BLOCK SIZE bytes	128	128	512	512	2048
FEC FIELD SIZE	75	62	261	108	248
TRANSEC HEADER	20	20	20	20	20
PAYLOAD SIZE	33	46	231	384	1780

Table H.2.3 Segment Size per FEC Rate

H.2.4 (NU) Data Fragmentation

The use of TRANSEC (see Annex I) requires that all of the inputs to the TRANSEC encryption algorithm be exactly the same size. This fragmentation step ensures that this condition is met.

Figure H.2.6 shows the transformation of a Data Block (or of an Packet Level Encrypted Data Block or a Data Segment) into a series of Data Fragments: Fragment Headers plus Fragment Contents. Each Data Block is split into a series of smaller fragments, and each fragment is given a header for recombining, on the Remote side, the fragments into the Data Block they came from. If the final fragment is too short, it is padded with a meaningless bit pattern to make it conform to the required size. (The composition of this “meaningless bit pattern” is implementation dependent, and thus outside the scope of this document: since these bits are discarded by the Remote, they truly are meaningless.) Table H.2.4 depicts the contents of the Fragment Header in detail.

Note that while the individual Fragment Contents may be of varying lengths, the Fixed Length Content must be of a constant size. Accordingly, the Hub must keep track of the growing size of the Fixed Length Content, as additional Data Fragments are constructed and added to it. The Fragment Contents of the final Data Fragment in each Fixed Length Content must be given a length to exactly finish off the length of the Fixed Length Content.

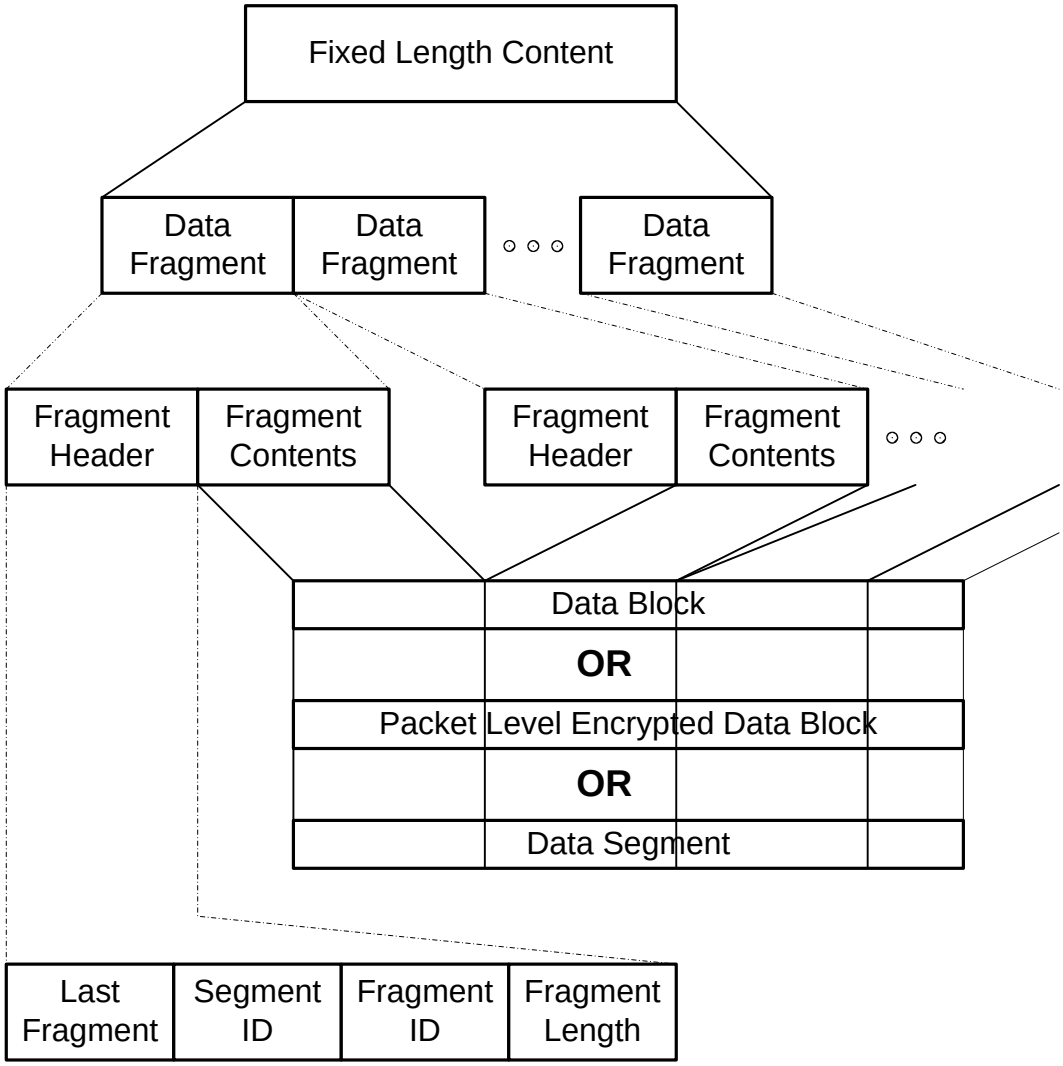


Figure H.2.6 Downstream Data Fragmentation

INDEX	NAME	SIZE	USE
1	Last Fragment	1 bit	set to 1 if last fragment of this segment
2	Segment ID	3 bits	common for all fragments of a segment
3	Fragment ID	3 or 5 bits	sequential for fragments in a segment
4	Fragment Length	9 or 7 bits	size of this fragment in bytes

Table H.2.4 Fragment Header Fields

1. Last Fragment. Set to 0 if there are more Fragments which make up this Segment. Set to 1 if this is the last Fragment making up this Segment.

2. Segment ID. Rotating values of 0 through 7. All Fragments which are part of the same Segment will be set to the same Segment ID. At the start of the next Segment, this value will be incremented (rolling over from 7 to 0 when appropriate).

3. Fragment ID. If Small Frames are being used, then this data field is five bits in length, and has values from zero up to a possible 31. If Large Frames are being used, then this data field is three bytes in length, and has values from zero up to a possible 7. Starts at 0 for the first Fragment of each Segment. Each subsequent Fragment in the same Segment is assigned the next sequential number as its Fragment ID.

4. Fragment Length. If Small Frames are being used, then this data field is seven bits in length, and has values up to a possible 127. If Large Frames are being used, then this data field is nine bits in length, and has values up to a possible 511. This measures the length, in bytes, of the Fragment Contents to which this Fragment Header is appended.

Note that the selection between Small Frames and Large Frames was made as part of the original System Configuration – and is consistent for all Remotes in the same InRoute Group – and as such is outside the scope of this document.

H.2.5 (NU) HDLC Encoding

This step builds the Link Level, peer to peer handshaking between Hub and Remote, onto the data being transmitted. This handshaking allows the Hub and Remote to guarantee

successful and error free delivery of the data, by allowing for a retransmission of it, should a problem occur during the initial transmission.

H.2.5.1 (NU) Bit Stuffing

Binary network communications use a recognizable Mark or Flag as a means by which a transmitter can tell a receiver that some condition has changed. Such Flags must be sent when needed, without waiting for a gap in the network customers' data. The standard value for this Flag is $H7E, or six "1" bits in a row. To protect against customer data coincidentally having this pattern imbedded within it, the transmission side device (the Hub in this case) must scan the bit stream. Every time it see six "1" bits in a row, it will insert a "0" as the next bit. [The receiving side will remove this "0".]

H.2.5.2 (NU) HDLC Header

This header is appended on the Hub side and stripped off on the Remote side. It is used first to tell all of the Remotes which Remote should be looking at the data block. All other Remotes will receive but ignore this HDLC Block. The second purpose is to exchange Link Level handshakes between the Hub and the Remote.

Figure H.2.7 shows a Fixed Length Content Block being given a Link Level Header, and then a CRC being appended to the block. Since the Link Level Header and the CRC fields are always the same size, the entire HDLC Block will also always be the appropriate size for the TRANSEC encryption in the next step.

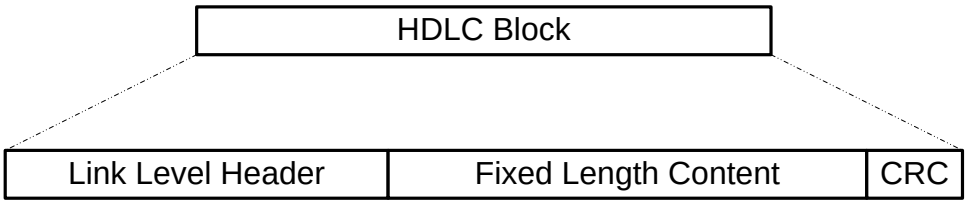


Figure H.2.7 HDLC Block

Please refer to section H.6.1, "Link Level", which shows a breakdown of the four Frame Types within the Link Level Control Field. The Tables in that section depict the fields and values within the Link Level Header.

At this step, when the Link Level addressing is appended to the Fixed Length Content Block, there is no longer any need to keep separate track of internally sourced Data Blocks versus incoming IP Packet sourced Data Blocks. From this step onwards, there is no requirement that externally generated and internally generated blocks be tracked separately; although there may be other reasons (e.g., delivery priorities) for the Hub to keep an internal tracking to differentiate them. Such an implementation detail is outside the scope of this document.

H.2.5.3 (NU) Appending CRC

After the appropriate Link Level Header has been prepended to the Fixed Length Content Block, the resulting bit stream is run through the standard CRC algorithm, and the resulting CRC field is appended to the end of the Block.

The CRC polynomial used shall be the CRC-16-CCITT standard: $x^{16} + x^{12} + x^5 + 1$

H.2.6 (NU) TRANSEC

Please refer to Annex I, section I.2, “Downstream TRANSEC Encryption” for the information on this step.

Some Blocks, all internally generated, will not have been passed through the TRANSEC encryption process. These Clear Text Blocks are destined for Remotes that are still being acquired by the Network, and have not yet had an opportunity to exchange Certificates with the Hub. These Blocks are discussed in detail in Annexes E and F.

H.2.7 (NU) Bit Scrambling

Ideally, the bits, or more accurately the symbols which are their analog counterparts, would be random enough to allow the spectrum energy to be evenly spread across the entire bandwidth. In practice, byte and word boundaries result in peaks. To artificially even these peaks out, the bitstream generated by the HDLC step (section H.2.6) is systematically scrambled [and descrambled on the Remote].

Scrambling is to comply with the Intelsat Intermediate Data Rate (IDR) standard mode, as per IESS-309, Appendix F, section F.6.

The polynomial to be used is: $1 + x^{-14} + x^{-15}$

The Scrambler seed is: 001001001001001

H.2.8 (NU) Forward Error Correction

Forward Error Correcting (FEC) decreases the number of packets which have to be resent due to a transmission error. Over a satellite hop, this is a great savings, and justifies the large percentage of the bandwidth which the FEC occupies.

Figure H.2.8 shows how a series of HDLC Blocks are encoded into a stream of FEC Frames. The “UW” field is a Unique Word, as discussed in Section H.2.8.3.

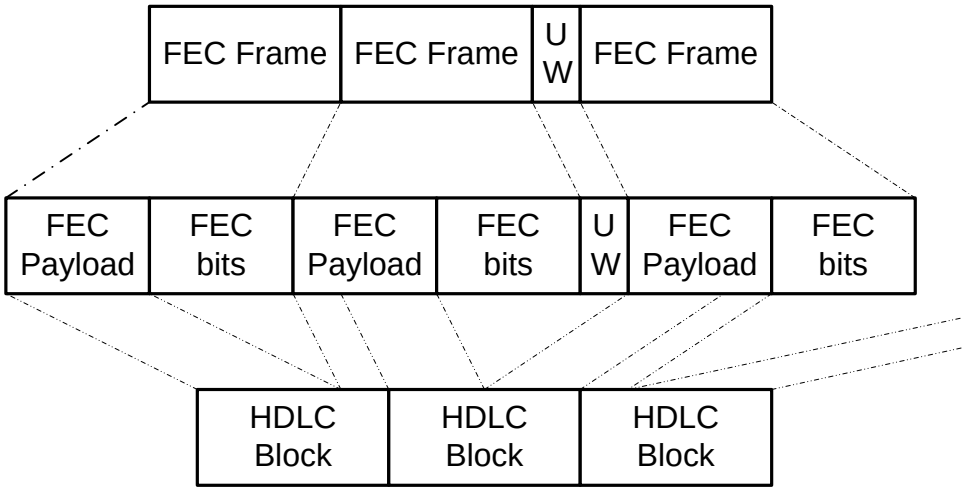


Figure H.2.8 FEC Stream Creation

When one FEC Payload field is filled before all of the current HDLC Block has been added, the remainder of the HDLC Block will start filling up the next FEC Payload field. If an HDLC Block has been completely added, and there is still room left in the current FEC Payload field, then the next HDLC Block will start loading immediately.

H.2.8.1 (NU) FEC Rates and Payload Sizes

Table H.2.5 shows the payload characteristics for each of five standard FEC rates. The Hub must be able to encode and the Remote decode FEC Blocks at any of these five standard rates.

The selection of which encoding is in use was made during System Configuration, and is the same for all Remotes in the same InRoute Group. How the Hub makes, or is assigned, this choice is outside the scope of this document.

FEC RATES	0.431	0.533	0.495	0.793	0.879
BLOCK SIZE bytes	128	128	512	512	2048
FEC FIELD SIZE	75	62	261	108	248
TRANSEC HEADER	20	20	20	20	20
PAYLOAD SIZE	33	46	231	384	1780

Table H.2.5 Payload Size per FEC Rate

H.2.8.2 (NU) FEC Encoding

FEC Encoding (and decoding) shall be based on the IEEE 802.16 standard, and the methodology for FEC stated therein.

H.2.8.3 (NU) Unique Word

As the Hub builds the FEC stream, it must keep track of the number of FEC Frames it has created. After every “N” FEC Frames, it will insert a Unique Word between that FEC Frame and the successive FEC Frame. The selection of the size of “N” is made when the Hub is established, and is outside the scope of this document.

The Unique Word is eight bytes long, and is used as a synchronization flag by the Remote. Its known symbol sequence can be used to aid in demodulation of the data stream.

The digital representation of the Unique Word is a 32 symbol sequence:

[1 1 1 -1 1 -1 -1 -1 1 -1 1 1 1 -1 1 1 1 1 -1 -1 -1 1 -1 -1 1 -1 1 -1 -1 1 -1 -1]

H.2.9 (NU) Encoding as amplitudes and phases

In preparation for the creation of an analog symbol stream, out of the FEC Frame bit stream, the bit stream is first converted into a series of two-component vectors, one vector for each symbol, represented at this point as digital values for each component of the vector. The number of bits which are used to define one symbol, and the exact encoding of the vector for

that symbol, are based on which bit mapping schema is in use. Each of the three bit mapping schemas is described separately, in section H.6.2.

The selection of which bit mapping schema will be used is determined during network installation, and is outside the scope of this document.

H.2.10 (NU) Symbol Mapping

The stream of two-component vectors (digital representations of symbols) is split into two separate but lockstepped streams: one of the I components, the other of the Q components.

H.2.11 (NU) Square Root Raised Cosine Pulse Shaping

The SRRC filter shall comply with the IESS 308/309 spectral mask, and use a 1.2 rolloff.

See Section H.6.3, Signal Spectrum, for the Signal Spectrum.Template and the Modulator Filter Group Delay Template.

H.2.12 (NU) Intermediate Frequency (IF) Signals

The baseband signal is converted to an L-Band IF frequency. IF frequencies are proprietary, local to either the Hub or the Remote, and thus are outside the scope of this document. The Carrier frequency may be anywhere between 950 MHz and 1,700 MHz.

H.3 (NU) Downstream: Hub to Remote, Remote Side

This section describes the necessary actions that the Remote must take at each step, from when it first receives an Intermediate Frequency stream of analog symbols, from the hardware connected to its antenna, until it hands off a digital data packet to the outside world.

H.3.1 (NU) Intermediate Frequency (IF) Signals

The RF signal from the antenna is converted to an L-Band IF frequency. IF frequencies are proprietary, local to either the Hub or the Remote, and thus are outside the scope of this document. The Carrier frequency may be anywhere between 950 MHz and 1,700 MHz. L-Band signals are then downconverted to baseband.

H.3.2 (NU) Demodulation and Frame Synchronization

The digital demodulation locks to and tracks the carrier phase, symbol timing, and amplitude of the incoming baseband signal: such that the I/Q constellation may be sampled and demapped into bits.

When the Remote detects the arrival of a Unique Word, it will strip those symbols out of the (downstream) input stream, after using them to position the symbol stream so that the symbols may be separated into FEC blocks for input into the FEC decoder.

The digital representation of the Unique Word is a 32 symbol sequence:

[1 1 1 -1 1 -1 -1 -1 1 -1 1 1 1 -1 1 1 1 1 -1 -1 -1 1 -1 -1 1 -1 1 -1 -1 1 -1 -1]

H.3.3 (NU) FEC decoding

Forward Error Correcting (FEC) decreases the number of packets which have to be resent due to a transmission error. Over a satellite hop, this is a great savings, and justifies the large percentage of the bandwidth which the FEC occupies.

FEC Decoding shall be based on the Turbo Product Coding (TPC) as defined in the IEEE 802.16 standard (section 8.3.3.2.2), and the methodology for FEC stated therein. This is an iterative, recursive process which deals with the probabilities of multiple check sums and parity checks successfully detecting and identifying, and correcting, simultaneous transmission errors within one FEC Frame.

Figure H.3.1 depicts a stream of FEC frames (bits) being separated into individual frames; then each FEC Frame being run through the TPC algorithm to correct any transmission errors and restore the original data, and finally the building of a series of HDLC Blocks out of that original data.

As a final check, each HDLC Block is run through a CRC check to validate that the Block's contents are intact.

The CRC polynomial used shall be the CRC-16-CCITT standard:

$x^{16} + x^{12} + x^5 + 1$

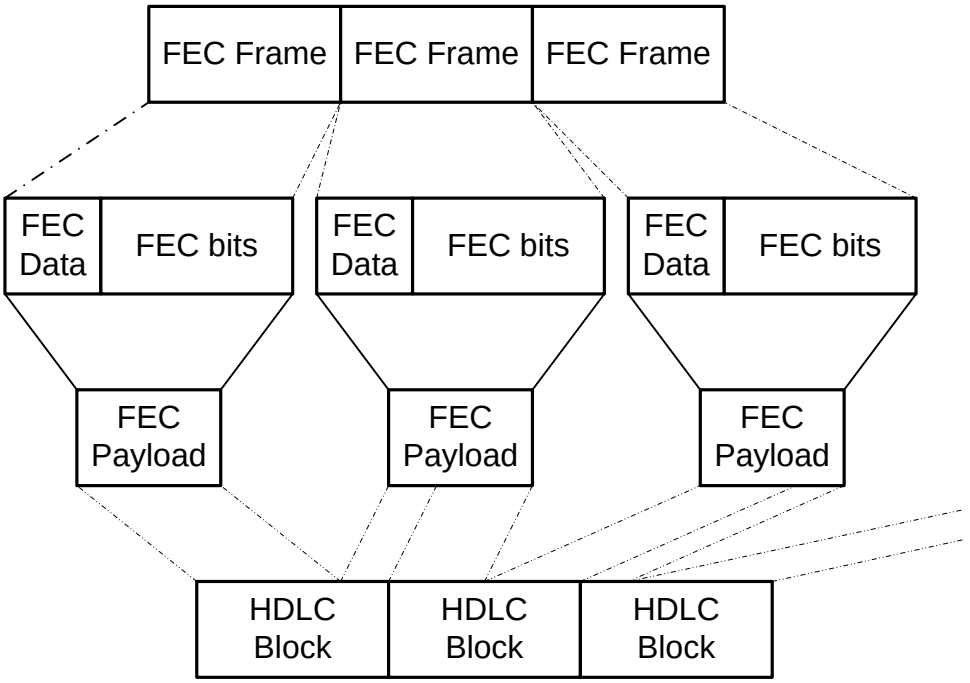


Figure H.3.1 FEC Stream converting to HLDC Stream

H.3.4 (NU) Bit Descrambling

Ideally, the bits, or more accurately the symbols which are their analog counterparts, would be random enough to allow the spectrum energy to be evenly spread across the entire bandwidth. In practice, byte and word boundaries result in peaks. To artificially even these

peaks out, the bitstream generated by the Hub's HDLC step (section H.2.6) must now be systematically descrambled

Descrambling is to comply with the Intelsat Intermediate Data Rate (IDR) standard mode, as per IESS-309, Appendix F, section F.6.

The polynomial to be used is: $1 + x^{-14} + x^{-15}$

The Scrambler seed is: 001001001001001

H.3.5 (NU) TRANSEC decryption

Please refer to Annex I, section I.3, "Downstream TRANSEC Decryption" for the information on this step.

H.3.6 (NU) HDLC Decoding

As shown in Figure H.3.2, this HDLC Block consists of a Link Level Header, a Text Block (the actual payload), and a CRC.

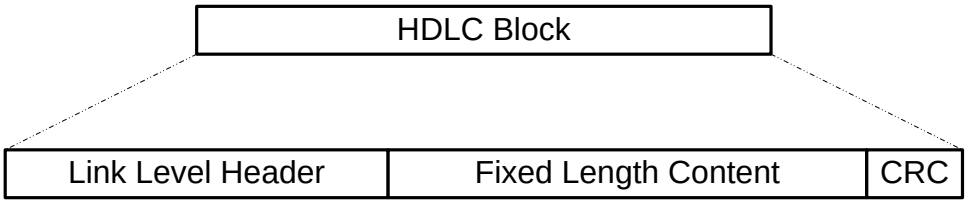


Figure H.3.2 HDLC Block

As a final check, once the FEC algorithm has extracted the best reproduction of the source FEC payload, the Remote will perform a CRC check on the reassembled HDLC Block, to validate that the Block was indeed transmitted and received accurately. If a Block's CRC check indicates that the Block was not transmitted successfully, then the Remote will send a Frame Reject back to the Hub, telling the Hub at which Block to begin resending. For more details on this, see section 6.1.2, "Supervisory Frames".

After the Block has been validated, its Link Level header is then interpreted, as described in detail in section H.6.1, "Link Level".

The data block is then run through the inverse of Bit Stuffing. Binary network communications use a recognizable Mark or Flag as a means by which a transmitter can tell a receiver that some condition has changed. Such Flags must be sent when needed, without waiting for a gap in the network customers' data. The standard value for this Flag is $H7E, or six "1" bits in a row. To protect against customer data coincidentally having this pattern imbedded within it, the transmission side device (the Hub in this case) will have scanned the bit stream. Every time it saw six "1" bits in a row, it will have inserted a "0" as the next bit. The Remote must now also scan for six "1" bits in a row, and every time it finds that sequence it must remove the following seventh bit (which will always be a "0").

H.3.7 (NU) Partitioning Burst

By the very nature of satellite communications, the Hub sends out its transmissions as a burst to all Remotes, rather than targeting each Remote with only its individual destination data. As such, it is necessary for each Remote to extract, from the burst, exactly those data blocks which have been addressed to it.

H.3.7.1 (NU) Pull Off Timeplan

As described in Annex F, "Link Maintenance", the Hub sends out information about each InRoute, including both a Timeplan for that InRoute's frequency, and occasionally specific commands to an individual Remote for making adjustments to its transmission aspects. The Remote must look at all of the InRoutes in the InRoute Group to which it belongs. It must check to see if there are any commands to it, to adjust its transmission aspects, and if so it must make them. And it must scan the Timeplan for that InRoute, looking to see which (if any) Time Division slots have been assigned to it, for it to transmit into.

H.3.7.2 (NU) Discard Packets for Other Remotes

The Link Level Information Frames which are received (and decrypted, etc., as described above) all have an Address Field which identifies which Remote is the intended recipient of this Frame. The Remote will recognize and keep only those Frames which have been addressed to it, discarding all others.

H.3.8 (NU) Data Defragmentation

Because of the necessity of having a fixed length of content as input to the Hub's TRANSEC encryption, data was broken down into Fragments, which could be adjusted in size so as to exactly fill the fixed length. Accordingly, the Remote must take these Fragments and

recombine their contents. Figure H.3.3 shows the relationship between the TRANSEC's fixed length Block, the Data Fragments which make up that Block, and the Data Segment(s) which are rebuilt from the Fragments.

Note that if the optional Data Segmentation step was not used by the Hub, then the "Data Segment" described as the result of this step will instead be a "Packet Level Encrypted Data Block"; or, if that option also was not used, then the result of the defragmentation will be a "Data Block".

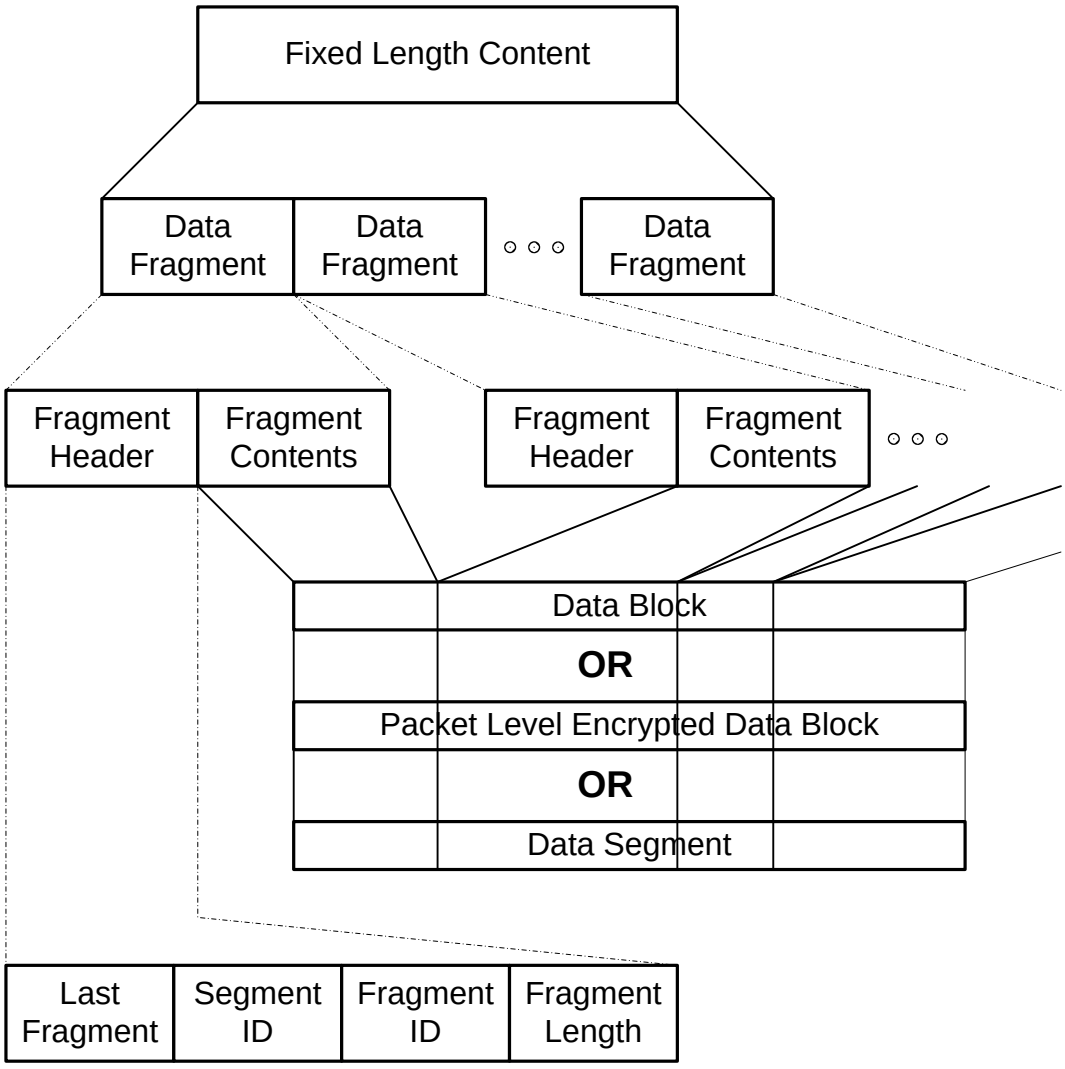


Figure H.3.3 Downstream Data Defragmentation

Each Fragment's Header contains information which is used by the Remote to rebuild a Data Segment from the (one or more) Data Fragments into which it was split. Table H.3.1 describes the contents of the Fragment Header.

INDEX	NAME	SIZE	USE
1	Last Fragment	1 bit	set to 1 if last fragment of this segment
2	Segment ID	3 bits	common for all fragments of a segment
3	Fragment ID	3 or 5 bits	sequential for fragments in a segment
4	Fragment Length	9 or 7 bits	size of this fragment in bytes

Table H.3.1 Fragment Header Fields

1. Last Fragment. Set to 0 if there are more Fragments which make up this Segment. Set to 1 if this is the last Fragment making up this Segment. When the Remote sees this flag, it knows it has completed the reconstruction of the current Data Segment.

2. Segment ID. Rotating values of 0 through 7. All Fragments which are part of the same Segment will be set to the same Segment ID. At the start of the next Segment, this value will be incremented (rolling over from 7 to 0 when appropriate).

3. Fragment ID. If Small Frames are being used, then this data field is five bits in length, and has values from zero up to a possible 31. If Large Frames are being used, then this data field is three bytes in length, and has values from zero up to a possible 7. Starts at 0 for the first Fragment of each Segment. Each subsequent Fragment in the same Segment is assigned the next sequential number as its Fragment ID.

4. Fragment Length. If Small Frames are being used, then this data field is six bits in length, and has values up to a possible 127. If Large Frames are being used, then this data field is nine bits in length, and has values up to a possible 511. This measures the length, in bytes, of the Fragment Contents to which this Fragment Header is appended.

Note that the selection between Small Frames and Large Frames was made as part of the original System Configuration – and is consistent for all Remotes in the same InRoute Group – and as such is outside the scope of this document.

H.3.9 (NU) Data Segmentation Reconstructions (optional)

If the Hub applied the optional step of breaking the Data Block (or the Packet Level Encrypted Data Block) into Segments, then the Remote must now recombine those Segments to reconstruct the original Block. Figure H.3.4 shows the relationship between the Data Segments and the Data Block(s) which are rebuilt from the Segments.

Note that if the optional Packet Level Encryption step was not used by the Hub, then the "Packet Level Encrypted Data Block" described as the result of this step will instead be a "Data Block".

Downstream Data Segmentation is turned on or off as part of the Remote Configuration process, and will apply to all, or none, of the packets being transmitted by the Hub, on a Remote-by-Remote basis. The decision of which Remotes will, and which will not, be have Data Segmentation turned on, as well as the methodology by which the Hub determines which packets will have Data Segmentation applied to them, are outside the scope of this document.

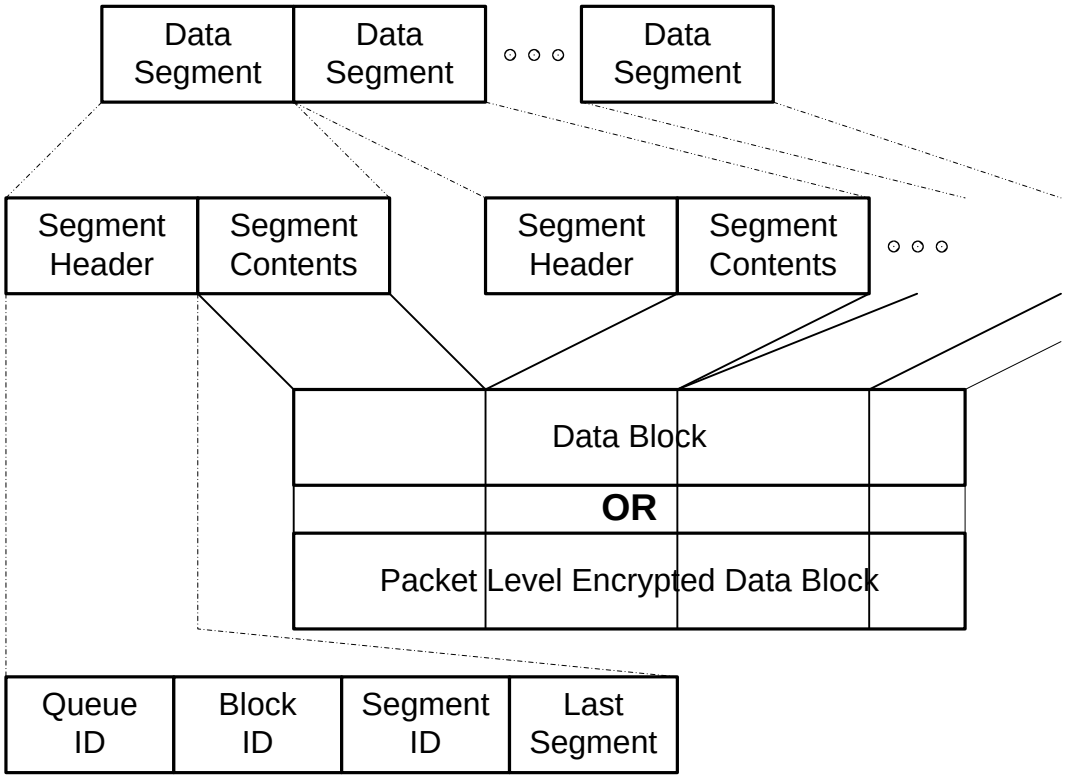


Figure H.3.4 Data Segmentation

Each Segment's Header contains information which is used by the Remote to rebuild a Data Block from the (one or more) Data Segments into which it was split. Table H.3.2 describes the contents of the Segment Header.

INDEX	NAME	SIZE	USE
1	Queue ID	6 bits	identifies QOS Service Level for Block
2	Block ID	3 bits	common for all Segments of a Block
3	Segment ID	6 bits	sequential for Segments in a Block
4	Last Segment	1 bit	set to 1 if last Segment of this Block

Table H.3.2 Segment Header Fields

1 Queue ID. All of the Data Segments which make up a single IP packet will have been placed, in order on the same transmission queue. This field uniquely identifies that common queue. The Queue ID itself is meaningless on the reception side, but the uniqueness of this field's value assists in the reassembly of the entire packet.

2. Block ID. Rotating values of 0 through 7. All Segments which are part of the same Block will be set to the same Block ID. At the start of the next Block, this value will be incremented, rolling over from 7 to 0.

3. Segment ID. Starts at 0 for the first Segment of each Block. Each subsequent Segment in the same Block is assigned the next sequential number as its Segment ID.

4. Last Segment. Set to 0 if there are more Segments which make up this Block. Set to 1 if this is the last Segment making up this Block.

H.3.10 (NU) Decryption at Packet Level (optional)

The Hub may have performed an encryption on an entire incoming packet. Whether the Hub applied this step to all incoming packets, to none of the incoming packets, or only to selected ones, and in the latter case how the Hub selects which packets to encrypt, are controlled by methodologies which are outside the scope of this document.

Note that this is a separate and independent step from the TRANSEC encryption (which is not optional).

When the Hub has applied Packet Level Encryption, it is necessary for the Remote to apply decryption, to allow the packet to be read at its ultimate destination.

Packet Level Decryption shall use the AES 256 CFB algorithm.

Details of packet level encryption are presented in section H.6.4.

H.3.11 (NU) VLAN (optional)

Virtual Local Area Networks (VLAN) are a method of logically tying together nodes (hosts, terminals, printers, etc.) into a logical network, even when there is no physically continuous network connecting them. To bridge the physical gap between one Virtual LAN segment and another, the VLAN is assigned a VLAN ID (VID), which is appended to all data blocks being transmitted between VLAN segments.

Whether or not a given Remote-Hub link will support VLAN tagging is determined at the time the Remote is registered at the Hub, and is thus outside the scope of this document. When a Remote-Hub link is defined as supporting VLAN tagging, every data block will have a VLAN Header (VH) appended to it, prior to all other processing.

VLAN tagging follows the IEEE 802.1d and 802.1q standards.

Figure H.3.5 shows the transformation of a “Data Block” into its components of a VLAN Header and a Data Block. Because there is no change to the data in the data block during this process; and because all previous steps treated the VLAN Header as just another pair of Data Block bytes both sides of the transformation are called a “Data Block”. Table H.3.3 depicts the contents of the VLAN Header in detail.

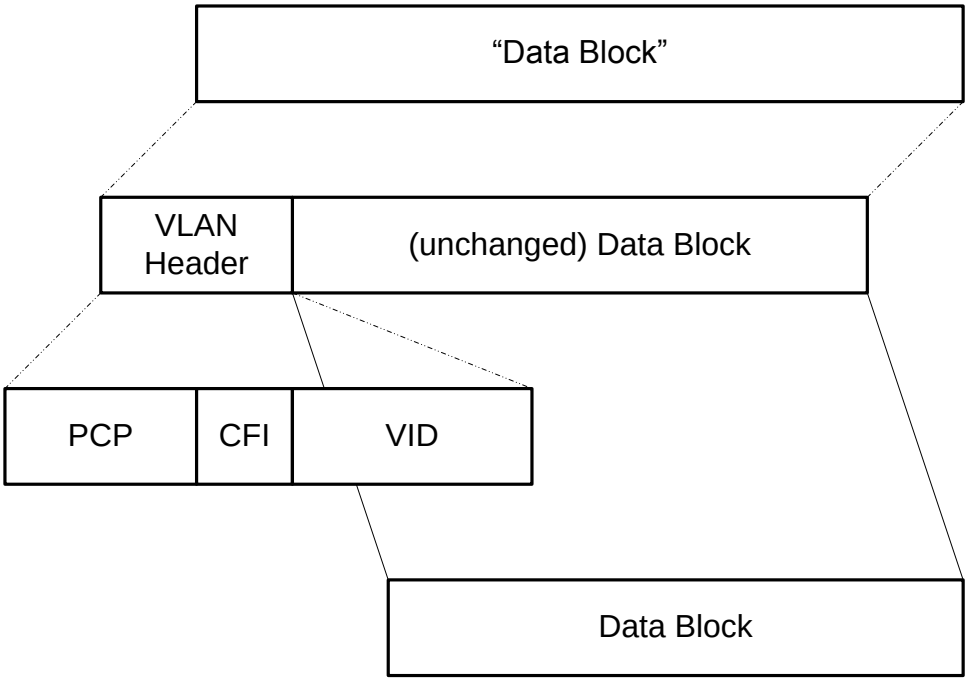


Figure H.3.5 VLAN Tagging

INDEX	NAME	SIZE	USE
1	PCP	3 bits	Priority Code Point
2	CFI	1 bit	Canonical Format Indicator
3	VID	12 bits	VLAN Identifier

Table H.3.3 VLAN Header Fields

1. PCP. Priority Code Point. Assignment of a priority to this data block, as per IEEE 802.1d. The determination of what priority to assign a given data block is outside the scope of this document. The implementation of methodologies to take advantage of those priorities is outside the scope of this document.

2. CFI. Canonical Format Indicator. As per IEEE 802.1q: if "0", the Media Access Control (MAC) address is in canonical format (e.g., as for an Ethernet network); if "1", the MAC is in non-canonical format (e.g., as for a Token Ring network). Determining which value to set this bit is outside the scope of this document.

3. VID. VLAN Identifier. Assignment of an identifier to allow physically separate parts of the same logical (virtual) network to be associated with each other by the intervening communications devices and protocols, as per IEEE 802.1q. The assignment and tracking of VIDs is outside the scope of this document. The reception of this VID will be used by the Remote as part of its Level 3 processing and routing, which is outside the scope of this document.

H.3.12 (NU) IP Packet Delivery

Once the IP Packet has been reconstructed, it is handed off to Level 3 processing for routing it towards its final destination.

H.4 (NU) Upstream: Remote to Hub, Remote side

This section describes the necessary actions that the Remote must take at each step, from when it first receives a digital data packet, until it hands off an Intermediate Frequency stream of analog symbols to the hardware connected to the antenna.

H.4.1 (NU) External Packets (IP only)

All packets arriving from the outside world must be IPv4 packets complying with IETF RFC 791. Any non-IP data which arrives at the Hub will be discarded.

Different packets may be assigned differing levels of priority for transmission. How packets are identified and marked for each different priority level, and what the Remote does to honor those priority requests, are both outside the scope of this document.

IP packets come in a variety of flavors. For purposes of this STANAG, these flavors may be collapsed into only three categories: TCP packets (as per IETF RFC 793); UDP packets (as per IETF RFC 768); and all other IP packets.

H.4.1.1 (NU) TCP packets

After this point, the Remote treats the entire TCP packet as a Data Block, thus indistinguishable from other IP packet types, as per Figure H.4.1.

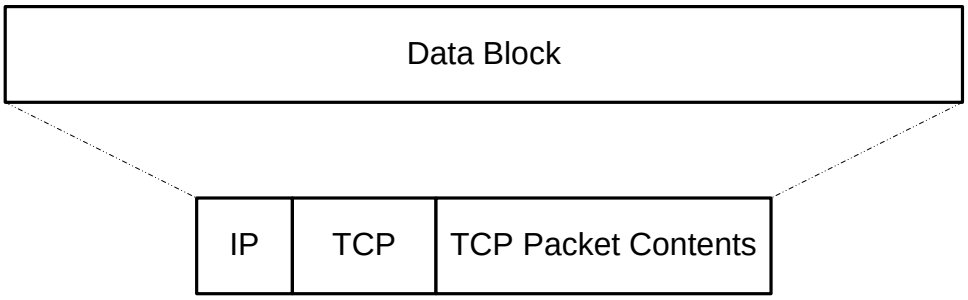


Figure H.4.1 TCP Packet to Data Block

H.4.1.2 (NU) UDP packets

Because some UDP packets are also RTP packets, the Remote may want to assign a higher priority to the transmission of these packets. The exact mechanism by which the Remote prioritizes among its waiting-to-be-sent data is outside the scope of this document.

Otherwise, after this point, the Remote treats the entire UDP packet as a Data Block, thus indistinguishable from other IP packet types, as per Figure H.4.2.

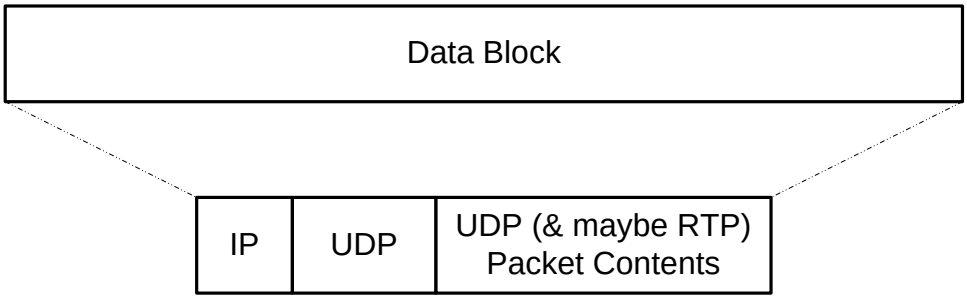


Figure H.4.2 UDP Packet to Data Block

H.4.1.3 (NU) Other IP packets

If the IP packet is neither TCP nor UDP, then no special processing is possible. After this point, the Remote treats the entire IP packet as a Data Block, thus indistinguishable from other IP packet types, as per Figure H.4.3.

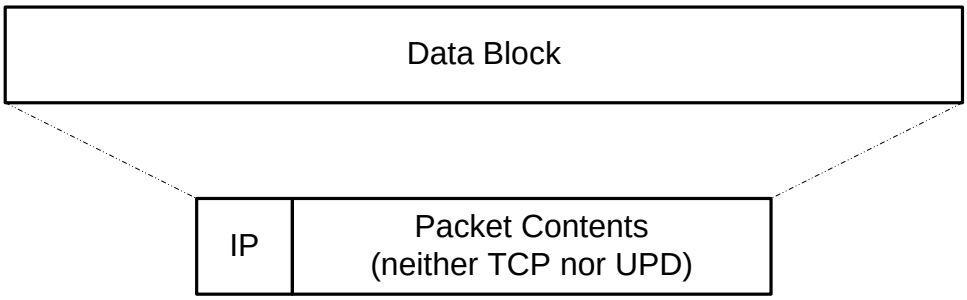


Figure H.4.3 IP Packet to Data Block

H.4.1.4 (NU) VLAN (optional)

Virtual Local Area Networks (VLAN) are a method of logically tying together nodes (hosts, terminals, printers, etc.) into a logical network, even when there is no physically continuous network connecting them. To bridge the physical gap between one Virtual LAN segment and another, the VLAN is assigned a VLAN ID (VID), which is appended to all data blocks being transmitted between VLAN segments.

Whether or not a given Remote-Hub link will support VLAN tagging is determined at the time the Remote is registered at the Hub, and is thus outside the scope of this document. When a Remote-Hub link is defined as supporting VLAN tagging, every data block will have a VLAN Header (VH) appended to it, prior to all other processing.

VLAN tagging follows the IEEE 802.1d and 802.1q standards.

Figure H.4.4 shows the transformation of a Data Block by the prepending of a VLAN Header. Because there is no change to the data in the data block during this process; and because all subsequent steps will treat the VLAN Header as just another pair of Data Block bytes – up until the VLAN processing step at the Remote – the “result” of prepending the VLAN Header is also called a “Data Block”. Table H.4.1 depicts the contents of the VLAN Header in detail.

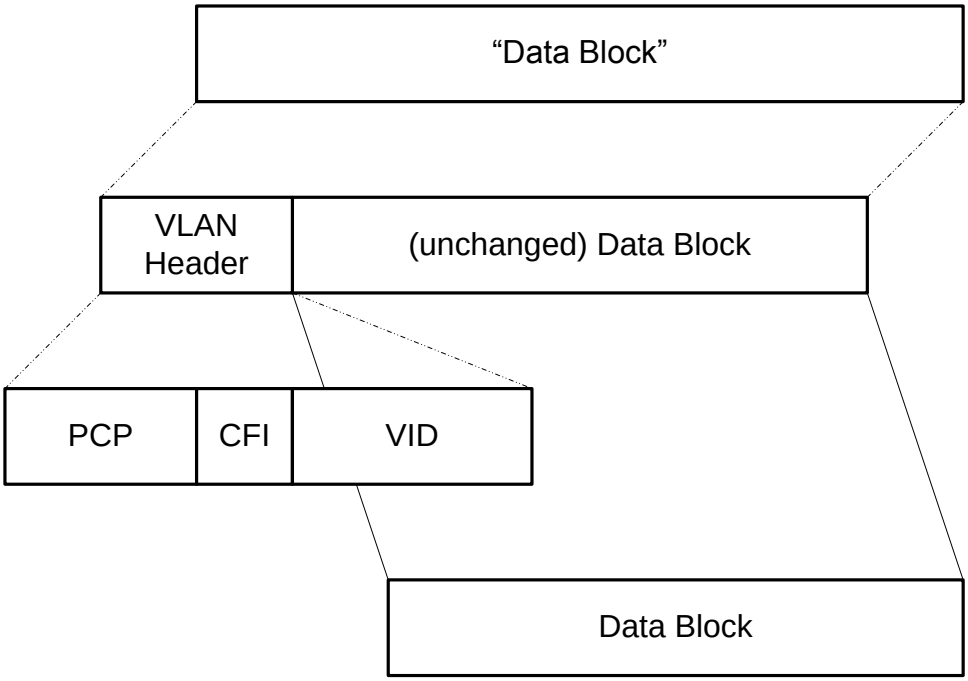


Figure H.4.4 VLAN Tagging

INDEX	NAME	SIZE	USE
1	PCP	3 bits	Priority Code Point
2	CFI	1 bit	Canonical Format Indicator
3	VID	12 bits	VLAN Identifier

Table H.4.1 VLAN Header Fields

1. PCP. Priority Code Point. Assignment of a priority to this data block, as per IEEE 802.1d. The determination of what priority to assign a given data block is outside the scope of this document. The implementation of methodologies to take advantage of those priorities is outside the scope of this document.

2. CFI. Canonical Format Indicator. As per IEEE 802.1q: if "0", the Media Access Control (MAC) address is in canonical format (e.g., as for an Ethernet network); if "1", the MAC is in non-canonical format (e.g., as for a Token Ring network). Determining which value to set this bit is outside the scope of this document.

3. VID. VLAN Identifier. Assignment of an identifier to allow physically separate parts of the same logical (virtual) network to be associated with each other by the intervening communications devices and protocols, as per IEEE 802.1q. The assignment and tracking of VIDs is outside the scope of this document.

H.4.2 (NU) Encryption at Packet Level (optional)

The Remote may perform an encryption on the entire incoming packet. Whether the Remote applies this step to all incoming packets, to none of the incoming packets, or only to selected ones, and in the latter case how the Remote selects which packets to encrypt, are controlled by methodologies which are outside the scope of this document.

Note that this is a separate and independent step from the TRANSEC encryption (which is not optional).

Packet level encryption shall use the AES 256 CFB algorithm.

Details of packet level encryption are presented in section H.6.4.

H.4.3 (NU) Data Segmentation

This is an optional step which is recommended due to the potential improvement in overall data transfer during periods of congestion. By breaking each Data Block down into smaller, fixed size Segments, it is possible to apply a finer grain of control over the intermingling of large and small Data Blocks, and correspondingly improve the delivery for all data.

Whether the Remote applies this step to all incoming packets, to none of the incoming packets, or only to selected ones, and in the latter case how the Remote selects which packets to segment, are controlled by methodologies which are outside the scope of this document.

Figure H.4.5 shows the transformation of a Data Block (or of an Packet Level Encrypted Data Block) into a series of Data Segments plus Segment Headers. Each Data Block is split into a series of smaller segments, and each segment is given a header for recombining, on the Hub side, the segments into the Data Block they came from. Table H.4.2 depicts the contents of the Segment Header in detail.

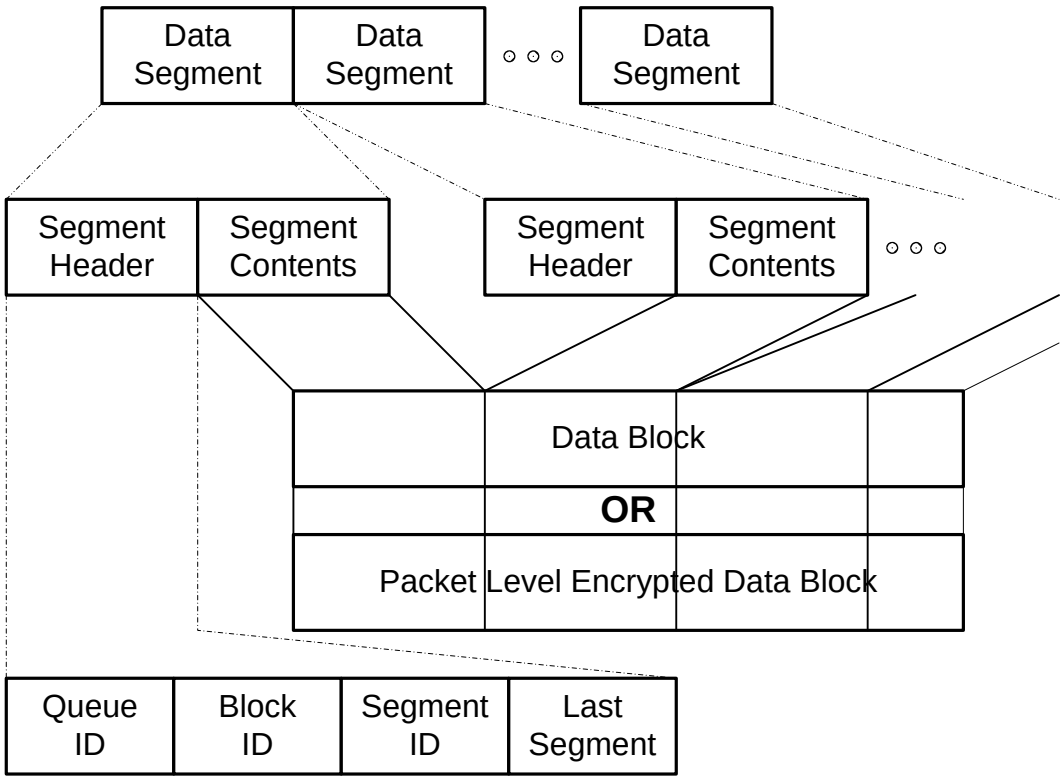


Figure H.4.5 Data Segmentation

INDEX	NAME	SIZE	USE
1	Queue ID	6 bits	identifies QOS Service Level for Block
2	Block ID	3 bits	common for all Segments of a Block
3	Segment ID	6 bits	sequential for Segments in a Block
4	Last Segment	1 bit	set to 1 if last Segment of this Block

Table H.4.2 Segment Header Fields

1 Queue ID. All of the Data Segments which make up a single IP packet will have been placed, in order on the same transmission queue. This field uniquely identifies that common queue. The Queue ID itself is meaningless on the reception side, but the uniqueness of this field's value assists in the reassembly of the entire packet.

2. Block ID. Rotating values of 0 through 7. All Segments which are part of the same Block will be set to the same Block ID. At the start of the next Block, this value will be incremented, rolling over from 7 to 0.

3. Segment ID. Starts at 0 for the first Segment of each Block. Each subsequent Segment in the same Block is assigned the next sequential number as its Segment ID.

4. Last Segment. Set to 0 if there are more Segments which make up this Block. Set to 1 if this is the last Segment making up this Block.

Data Segments are of a constant size (thus there is no need for a length indicator in the Segment Header). This size varies depending on the MODCOM in use; and in particular with the FEC Rate that is being used. Table H.4.3 shows the relationship between the FEC Rate and the segment size.

FEC RATES	**0.431**	**0.533**	**0.660**	**0.793**
BLOCK SIZE bytes	128	128	128	512
FEC FIELD SIZE	75	62	46	108
TRANSEC HEADER	4	4	4	4
TDMA OVERHEAD	10	10	10	10
DATA SEGMENTATION	2	2	2	2
PAYLOAD SIZE (no options)	37	50	66	388
if optional VLAN turned on	-2	-2	-2	-2
if optional Packet Encryption	-2	-2	-2	-2

Table 4.3 Segment Size per FEC Rate

H.4.4 (NU) Data Fragmentation

The use of TRANSEC (see Annex I) requires that all of the inputs to the TRANSEC encryption algorithm be exactly the same size. This fragmentation step is a necessary precursor to ensure that this condition is met.

Figure H.4.5 shows the transformation of a Data Block (or of an Packet Level Encrypted Data Block or a Data Segment) into a series of Data Fragments: Fragment Headers plus Fragment Contents. Each Data Block is split into a series of smaller fragments, and each fragment is given a header for recombining, on the Hub side, the fragments into the Data Block they came from. If the final fragment is too short, it is padded with a meaningless bit pattern to

make it conform to the required size. (The composition of this “meaningless bit pattern” is implementation dependent, and thus outside the scope of this document: since these bits are discarded by the Hub, they truly are meaningless.) Table H.4.4 depicts the contents of the Fragment Header in detail.

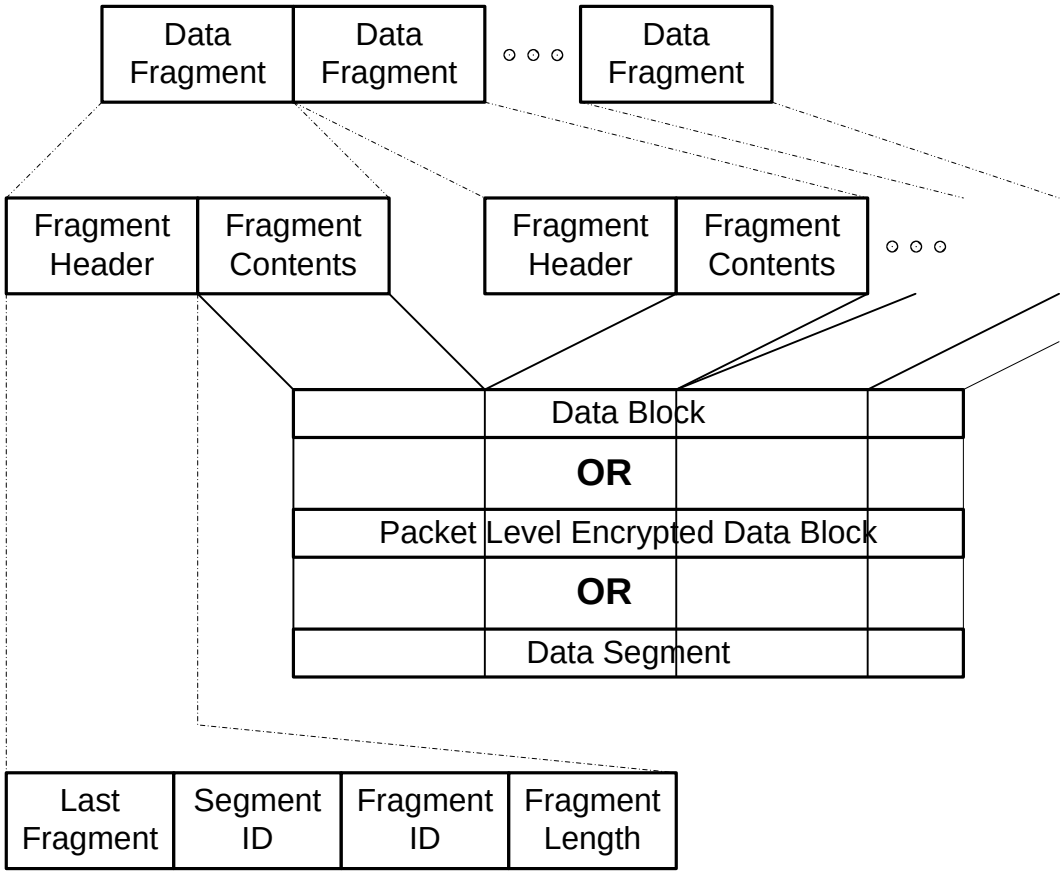


Figure H.4.6 Upstream Data Fragmentation

INDEX	NAME	SIZE	USE
1	Last Fragment	1 bit	set to 1 if last fragment of this segment
2	Segment ID	3 bits	common for all fragments of a segment
3	Fragment ID	3 or 5 bits	sequential for fragments in a segment
4	Fragment Length	9 or 7 bits	size of this fragment in bytes

Table H.4.4 Fragment Header Fields

1. Last Fragment. Set to 0 if there are more Fragments which make up this Segment. Set to 1 if this is the last Fragment making up this Segment.

2. Segment ID. Rotating values of 0 through 7. All Fragments which are part of the same Segment will be set to the same Segment ID. At the start of the next Segment, this value will be incremented (rolling over from 7 to 0 when appropriate).

3. Fragment ID. If Small Frames are being used, then this data field is five bits in length, and has values from zero up to a possible 31. If Large Frames are being used, then this data field is three bytes in length, and has values from zero up to a possible 7. Starts at 0 for the first Fragment of each Segment. Each subsequent Fragment in the same Segment is assigned the next sequential number as its Fragment ID.

4. Fragment Length. If Small Frames are being used, then this data field is seven bits in length, and has values up to a possible 127. If Large Frames are being used, then this data field is nine bits in length, and has values up to a possible 511. This measures the length, in bytes, of the Fragment Contents to which this Fragment Header is appended.

Note that the selection between Small Frames and Large Frames was made as part of the original System Configuration – and is consistent for all Remotes in the same InRoute Group – and as such is outside the scope of this document.

Note, however, that the Upstream TDMA Block, as described in section H.4.5, must be a constant size to meet the requirements for TRANSEC encryption. This constraint is met by having all Upstream Fragments be the same length. That, plus the constant size of a Fragment Header, and then additionally the constant size of the other components of the TDMA Block, guarantees that the input to the TRANSEC encryption process will always be the appropriate size.

H.4.5 (NU) TDMA

At this step, the Remote first builds and appends a TDMA Link Level Header to each Fragment, then prepends in front of that a Demand Header, and finally calculates and attaches a CRC, as shown in Figure H.4.7. Note that the constant sizes of these three fields – 2 bytes, 6 bytes, and 2 bytes, respectfully – ensures that the TDMA Block is of a constant size, as required by the TRANSEC algorithm.

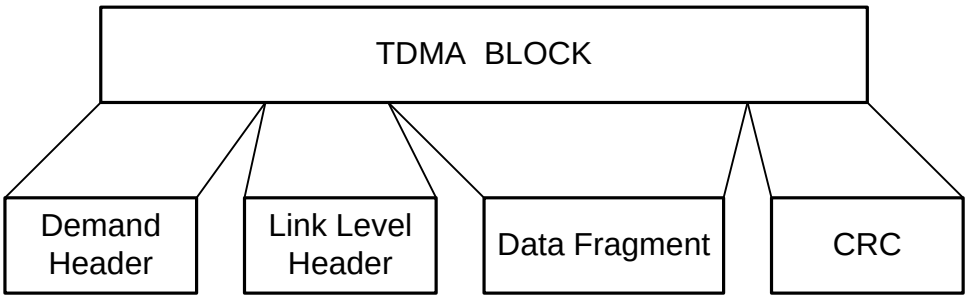


Figure H.4.7 TDMA Block

The Link Level Header is appended on the Remote side and stripped off on the Hub side. It is used first to tell the Hub which Remote this TDMA Block is from. The second purpose is to exchange Link Level handshakes between the Hub and the Remote.

Please refer to section H.6.1, “Link Level”, which shows a breakdown of the four Frame Types within the Link Level Control Field. The Tables in that section depict the fields and values within the Link Level Header.

The Demand Header is the means by which the Remote keeps the Hub informed concerning the amount of bandwidth the Remote would like to have, to fulfill its transmission requirements.

Please refer to Annex F, section F.6, “Bandwidth Request from Remote” for the details on this header.

After the appropriate Link Level Header and Demand Header have been prepended to the TDMA Block, the resulting bit stream is run through the standard CRC algorithm, and the resulting CRC field is appended to the end of the Block.

The CRC polynomial used shall be the CRC-16-CCITT standard:

$x^{16} + x^{12} + x^5 + 1$

H.4.6 (NU) TRANSEC

Please refer to Annex I, section I.2, “Downstream TRANSEC Encryption” for the information on this step.

H.4.7 (NU) Bit Scrambling

Ideally, the bits, or more accurately the symbols which are their analog counterparts, would be random enough to allow the spectrum energy to be evenly spread across the entire bandwidth. In practice, byte and word boundaries result in peaks. To artificially even these peaks out, the bitstream generated by the TRANSEC step (section H.4.7) is systematically scrambled [and descrambled on the Remote].

Scrambling is to comply with the Intelsat Intermediate Data Rate (IDR) standard mode, as per IESS-309, Appendix F, section F.6.

The polynomial to be used is: $1 + x^{-14} + x^{-15}$

The Scrambler seed is: 001001001001001

Note that for TDMA the scrambler is synchronized with the seed at the start of each burst, rather than the free running methodology used for SCPC.

H.4.8 (NU) FEC Encoding

Forward Error Correcting (FEC) decreases the number of packets which have to be resent due to a transmission error. Over a satellite hop, this is a great savings, and justifies the large percentage of the bandwidth which the FEC occupies.

Figure H.4.8 shows how a series of TRANSEC Blocks are encoded into a matching stream of FEC Frames. The “UW” field is a Unique Word, as discussed in Section H.4.8.3.

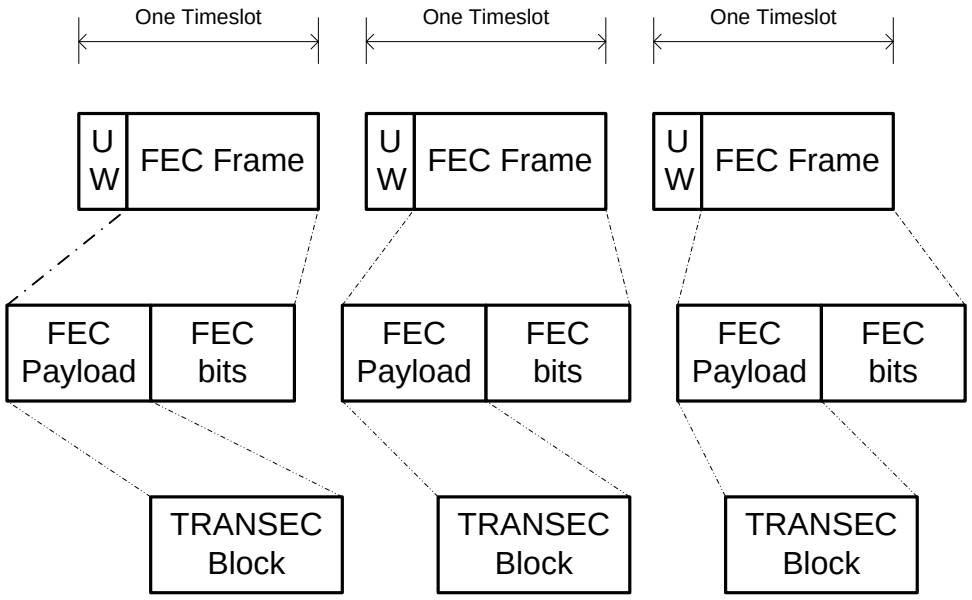


Figure H.4.8 FEC Stream Creation

Each TRANSEC Block becomes the Payload for one FEC Block. The fixed size of the TRANSEC Block, plus the fixed size of the FEC check bits, plus the fixed size of the Unique Word, ensures that, at a later step, the symbol stream transmitted into a single Time Slot (as per the Timeplan received by this Remote) will exactly fit that Time Slot.

H.4.8.1 (NU) FEC Rates and Payload Sizes

Table H.4.5 shows the payload characteristics for each of four standard upstream FEC rates. The Remote must be able to encode and the Hub decode FEC Blocks at any of these four standard rates.

The selection of which encoding is in use was made when the Hub established its InRoute Group(s), because all InRoutes in the same InRoute Group must use the same FEC rate. How the Hub makes, or is assigned, this choice is outside the scope of this document.

FEC RATES	0.431	0.533	0.660	0.793
BLOCK SIZE bytes	128	128	128	512
FEC FIELD SIZE	75	62	46	108
TRANSEC HEADER	4	4	4	4
TDMA OVERHEAD	10	10	10	10
PAYLOAD SIZE (no options)	39	52	68	390
if optional Data Segmentation	-2	-2	-2	-2
if optional VLAN turned on	-2	-2	-2	-2
if optional Packet Encryption	-2	-2	-2	-2

Table H.4.5 Payload Size per FEC Rate

H.4.8.2 (NU) FEC Encoding

FEC Encoding (and decoding) shall be based on the IEEE 802.16 standard, and the methodology for FEC stated therein.

H.4.8.3 (NU) Unique Word

As the Remote builds the FEC stream, it will insert a Unique Word in front of each successive FEC Frame. The Unique Word is eight bytes long, and is used as a synchronization flag by the Hub. Its known symbol sequence can be used to aid in demodulation of the data stream.

The digital representation of the Unique Word is varies with the bit mapping schema being used.

For QPSK and 8PSK, the digital representation of the Unique Word is a 32 symbol sequence:

[1 1 1 -1 1 -1 -1 -1 1 -1 1 1 1 -1 1 1 1 1 -1 -1 -1 1 -1 -1 1 -1 1 -1 -1 1 -1 -1]

For BPSK, the digital representation of the Unique Word is a 52 bit symbol sequence:

[0 0 1 0 1 1 0 1 1 0 0 0 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0]

H.4.9 (NU) Encoding as Amplitudes and Phases

In preparation for the creation of an analog symbol stream, out of the FEC Frame bit stream, the bit stream is first converted into a series of two-component vectors, one vector for each symbol, represented at this point as digital values for each component of the vector. The number of bits which are used to define one symbol, and the exact encoding of the vector for that symbol, are based on which bit mapping schema is in use. Each of the three bit mapping schemas is described separately, in section H.6.2.

The selection of which bit mapping schema will be used is determined during network installation, and is outside the scope of this document.

H.4.10 (NU) Symbol Mapping

The stream of two-component vectors (digital representations of symbols) is split into two separate but lockstepped streams: one of the I components, the other of the Q components.

H.4.11 (NU) Square Root Raised Cosine Pulse Shaping

The SRRC filter shall comply with the IESS 308/309 spectral mask, and use a 1.2 rolloff.

See Section H.6.3, Signal Spectrum, for the Signal Spectrum.Template and the Modulator Filter Group Delay Template.

H.4.12 (NU) Intermediate Frequency (IF) Signals

The baseband signal is converted to an L-Band IF frequency. IF frequencies are proprietary, local to either the Hub or the Remote, and thus are outside the scope of this document. The Carrier frequency may be anywhere between 950 MHz and 1,700 MHz.

H.4.13 (NU) Guard Times

Guard Times are not transmitted. Rather, they are margins between the transmissions of bursts into adjacent Time Slots. These margins protect against two different Remotes – which have been assigned by the Hub to burst into adjacent Time Slots in the same InRoute – each being slightly off in their time frame calibrations, such that without this margin the two bursts would overlap each other. [The length of the Guard Time is determined as part of the installation of the Remote, and thus outside the scope of this document.]

H.5 (NU) Upstream: Remote to Hub, Hub Side

This section describes the necessary actions that the Hub must take at each step, from when it first receives an Intermediate Frequency stream of analog symbols, from the hardware connected to its antenna, until it hands off a digital data packet to the outside world.

H.5.1 (NU) Intermediate Frequency (IF) Signals

The RF signal from the antenna is converted to an L-Band IF frequency. IF frequencies are proprietary, local to either the Hub or the Remote, and thus are outside the scope of this document. The Carrier frequency may be anywhere between 950 MHz and 1,700 MHz. L-Band signals are then downconverted to baseband.

H.5.2 (NU) Demodulation and Frame Synchronization

The digital demodulation detects the presence of the burst by correlating against the unique word sequence, and then estimates and corrects carrier phase, symbol timing, and amplitude of the baseband signal such that the I/Q constellation may be sampled and demapped to bits.

Note that one TDMA burst contains exactly one FEC block. Thus once the block is demodulated, the precise sample location of symbols of the FEC Block are known and may be input to the decoder.

When the Hub detects the arrival of a Unique Word, it will strip those symbols out of the (downstream) input stream, after using them to position the symbol stream so that the FEC block in this TDMA slot may be input into the FEC decoder.

The digital representation of the Unique Word is varies with the bit mapping schema being used.

For QPSK and 8PSK, the digital representation of the Unique Word is a 32 symbol sequence:

[1 1 1 -1 1 -1 -1 -1 1 -1 1 1 1 -1 1 1 1 1 -1 -1 -1 1 -1 -1 1 -1 1 -1 -1 1 -1 -1]

For BPSK, the digital representation of the Unique Word is a 52 bit symbol sequence:

[0 0 1 0 1 1 0 1 1 0 0 0 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0]

H.5.3 (NU) FEC Decoding

Forward Error Correcting (FEC) decreases the number of packets which have to be resent due to a transmission error. Over a satellite hop, this is a great savings, and justifies the large percentage of the bandwidth which the FEC occupies.

FEC Decoding shall be based on the Turbo Product Coding (TPC) as defined in the IEEE 802.16 standard (section 8.3.3.2.2), and the methodology for FEC stated therein. This is an iterative, recursive process which deals with the probabilities of multiple check sums and parity checks successfully detecting and identifying, and correcting, simultaneous transmission errors within one FEC Frame.

Figure H.5.1 depicts a stream of FEC frames (bits) being separated into individual frames; then each FEC Frame being run through the TPC algorithm to correct any transmission errors and restore the original data, and finally the TRANSEC Blocks which result: one TRANSEC Block per FEC Payload..

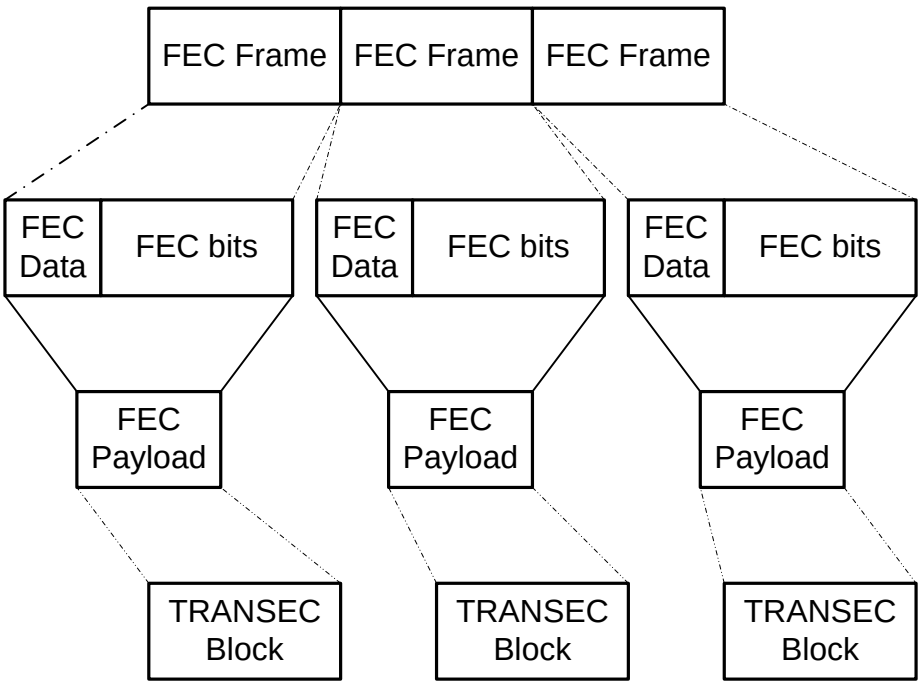


Figure H.5.1 FEC Stream converting to TRANSEC Stream

H.5.4 (NU) Bit Descrambling

Ideally, the bits, or more accurately the symbols which are their analog counterparts, would be random enough to allow the spectrum energy to be evenly spread across the entire bandwidth. In practice, byte and word boundaries result in peaks. To artificially even these peaks out, the bitstream generated by the Remote's TRANSEC step (section H.4.6) must now be systematically descrambled

Descrambling is to comply with the Intelsat Intermediate Data Rate (IDR) standard mode, as per IESS-309, Appendix F, section F.6.

The polynomial to be used is: $1 + x^{-14} + x^{-15}$

The Scrambler seed is: 001001001001001

Note that for TDMA the scrambler is synchronized with the seed at the start of each burst, rather than the free running methodology used for SCPC.

H.5.5 (NU) TRANSEC Decryption

Please refer to Annex I, section I.3, "Downstream TRANSEC Decryption" for the information on this step.

H.5.6 (NU) TDMA Header Interpretation

The first step is to perform a CRC Check on the now-decrypted content of the TRANSEC block.

The CRC polynomial used shall be the CRC-16-CCITT standard:

$x^{16} + x^{12} + x^5 + 1$

If the CRC Check is not successful, then a Link Level Frame Reject (SREJ) must be constructed and returned to the originating Remote. See Section H.6.1, "Link Level" for details on this step.

If the check is successful, there are two remaining parts to this step in the Hub's processing of the TDMA Block. It first interprets the Demand Header, and then handles the Link Level protocols.

The Demand Header is the means by which the Remote keeps the Hub informed concerning the amount of bandwidth the Remote would like to have, to fulfill its transmission requirements.

Please refer to Annex F, section F.6, “Bandwidth Request from Remote” for the details on this header.

The Link Level Header is appended on the Remote side and stripped off on the Hub side. It is used first to tell the Hub which Remote this TDMA Block is from. The second purpose is to exchange Link Level handshakes between the Hub and the Remote.

Please refer to section H.6.1, “Link Level”, which shows a breakdown of the four Frame Types within the Link Level Control Field. The Tables in that section depict the fields and values within the Link Level Header.

H.5.7 (NU) Data Defragmentation

Because of the necessity of having a fixed length of content as input to the Remote’s TRANSEC encryption, data was broken down into Fragments, which could be adjusted in size so as to exactly fill the fixed length. Accordingly, the Hub must take these Fragments and recombine their contents. Figure H.5.3 shows the relationship between the TRANSEC’s fixed length Block, the Data Fragments which make up that Block, and the Data Segment(s) which are rebuilt from the Fragments.

Note that if the optional Data Segmentation step was not used by the Remote, then the “Data Segment” described as the result of this step will instead be a “Packet Level Encrypted Data Block”; or, if that option also was not used, then the result of the defragmentation will be a “Data Block”.

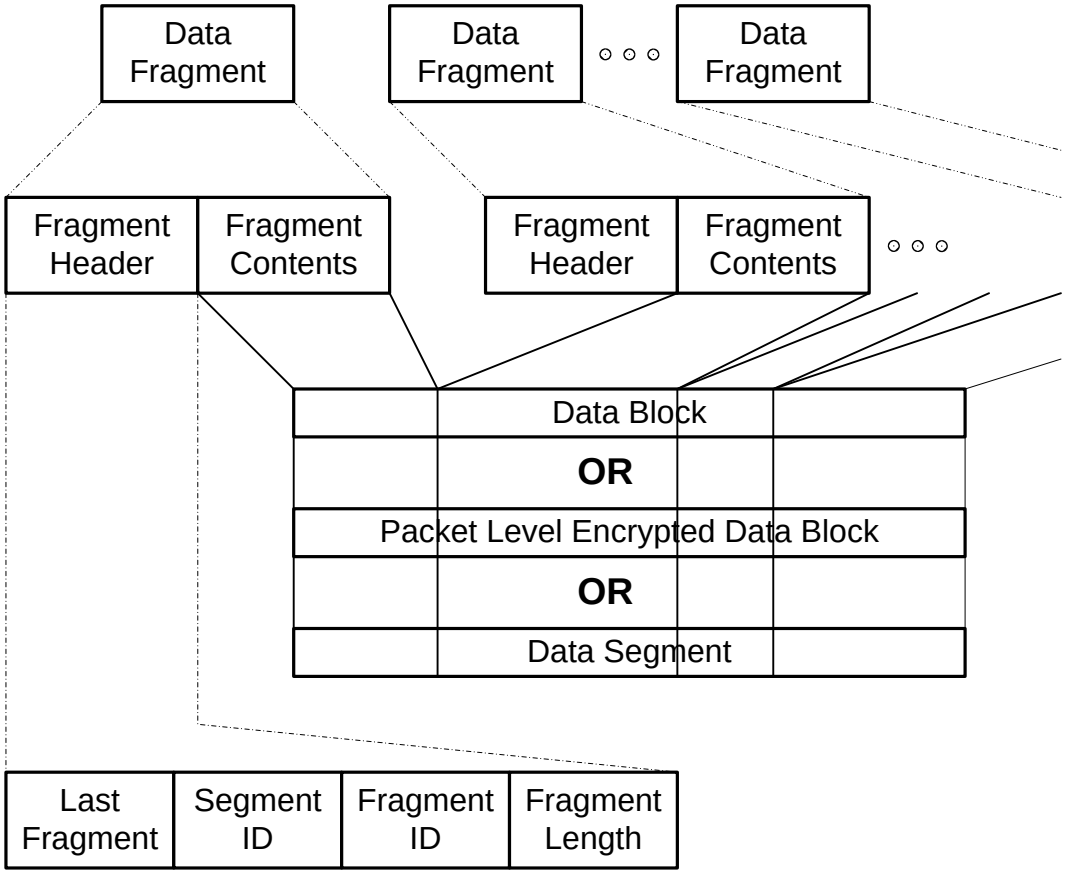


Figure H.5.1 Upstream Data Defragmentation

Each Fragment's Header contains information which is used by the Hub to rebuild a Data Segment from the (one or more) Data Fragments into which it was split. Table H.5.1 describes the contents of the Fragment Header.

INDEX	NAME	SIZE	USE
1	Last Fragment	1 bit	set to 1 if last fragment of this segment
2	Segment ID	3 bits	common for all fragments of a segment
3	Fragment ID	3 or 5 bits	sequential for fragments in a segment
4	Fragment Length	9 or 7 bits	size of this fragment in bytes

Table H.5.1 Fragment Header Fields

1. Last Fragment. Set to 0 if there are more Fragments which make up this Segment. Set to 1 if this is the last Fragment making up this Segment. When the Remote sees this flag, it knows it has completed the reconstruction of the current Data Segement.

2. Segment ID. Rotating values of 0 through 7. All Fragments which are part of the same Segment will be set to the same Segment ID. At the start of the next Segment, this value will be incremented (rolling over from 7 to 0 when appropriate).

3. Fragment ID. If Small Frames are being used, then this data field is five bits in length, and has values from zero up to a possible 31. If Large Frames are being used, then this data field is three bytes in length, and has values from zero up to a possible 7. Starts at 0 for the first Fragment of each Segment. Each subsequent Fragment in the same Segment is assigned the next sequential number as its Fragment ID.

4. Fragment Length. If Small Frames are being used, then this data field is seven bits in length, and has values up to a possible 127. If Large Frames are being used, then this data field is nine bits in length, and has values up to a possible 511. This measures the length, in bytes, of the Fragment Contents to which this Fragment Header is appended.

Note that the selection between Small Frames and Large Frames was made as part of the original System Configuration – and is consistent for all Remotes in the same InRoute Group – and as such is outside the scope of this document.

H.5.8 (NU) Data Segmentation Reconstructions

If the Remote applied the optional step of breaking the Data Block (or the Packet Level Encrypted Data Block) into Segments, then the Hub must now recombine those Segments to reconstruct the original Block. Figure H.5.4 shows the relationship between the Data Segments and the Data Block(s) which are rebuilt from the Segments.

Note that if the optional Packet Level Encryption step was not used by the Hub, then the “Packet Level Encrypted Data Block” described as the result of this step will instead be a “Data Block”.

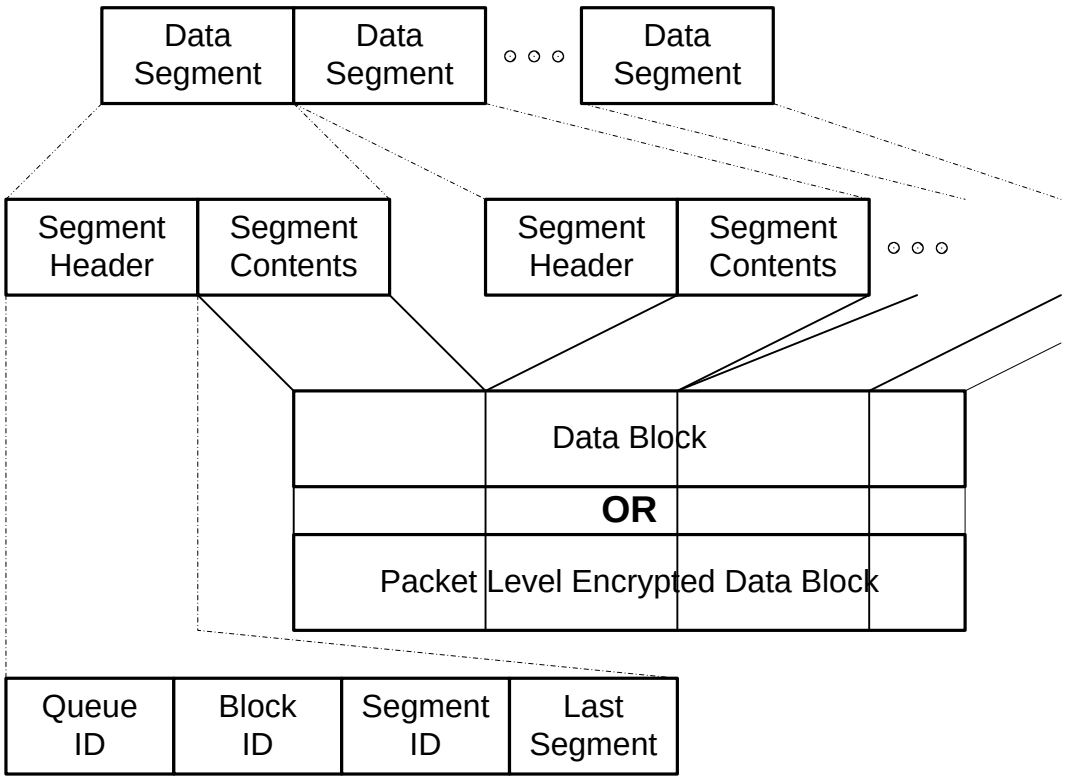


Figure H.5.2 Data Segmentation

Each Segment's Header contains information which is used by the Remote to rebuild a Data Block from the (one or more) Data Segments into which it was split. Table H.5.2 describes the contents of the Segment Header.

INDEX	NAME	SIZE	USE
1	Queue ID	6 bits	identifies QOS Service Level for Block
2	Block ID	3 bits	common for all Segments of a Block
3	Segment ID	6 bits	sequential for Segments in a Block
4	Last Segment	1 bit	set to 1 if last Segment of this Block

Table H.5.2 Segment Header Fields

1 Queue ID. All of the Data Segments which make up a single IP packet will have been placed, in order on the same transmission queue. This field uniquely identifies that common queue. The Queue ID itself is meaningless on the reception side, but the uniqueness of this field's value assists in the reassembly of the entire packet.

2. Block ID. Rotating values of 0 through 7. All Segments which are part of the same Block will be set to the same Block ID. At the start of the next Block, this value will be incremented, rolling over from 7 to 0.

3. Segment ID. Starts at 0 for the first Segment of each Block. Each subsequent Segment in the same Block is assigned the next sequential number as its Segment ID.

4. Last Segment. Set to 0 if there are more Segments which make up this Block. Set to 1 if this is the last Segment making up this Block.

H.5.9 (NU) Decryption at Packet Level (optional)

The Remote may have performed an encryption on an entire incoming packet. Whether the Remote applied this step to all incoming packets, to none of the incoming packets, or only to selected ones, and in the latter case how the Remote selected which packets to encrypt, are controlled by methodologies which are outside the scope of this document.

Note that this is a separate and independent step from the TRANSEC encryption (which is not optional).

When the Remote has applied Packet Level Encryption, it is necessary for the Hub to apply decryption, to allow the packet to be read at its ultimate destination.

Packet Level Decryption shall use the 3-DES CFB algorithm.

Details of packet level encryption are presented in section H.6.4.

H.5.10 (NU) VLAN (optional)

Virtual Local Area Networks (VLAN) are a method of logically tying together nodes (hosts, terminals, printers, etc.) into a logical network, even when there is no physically continuous network connecting them. To bridge the physical gap between one Virtual LAN segment and another, the VLAN is assigned a VLAN ID (VID), which is appended to all data blocks being transmitted between VLAN segments.

Whether or not a given Remote-Hub link will support VLAN tagging is determined at the time the Remote is registered at the Hub, and is thus outside the scope of this document. When a Remote-Hub link is defined as supporting VLAN tagging, every data block will have a VLAN Header (VH) appended to it, prior to all other processing.

VLAN tagging follows the IEEE 802.1d and 802.1q standards.

Figure H.5.3 shows the transformation of a “Data Block” into its components of a VLAN Header and a Data Block. Because there is no change to the data in the data block during this process; and because all previous steps treated the VLAN Header as just another pair of Data Block bytes both sides of the transformation are called a “Data Block”. Table H.5.3 depicts the contents of the VLAN Header in detail.

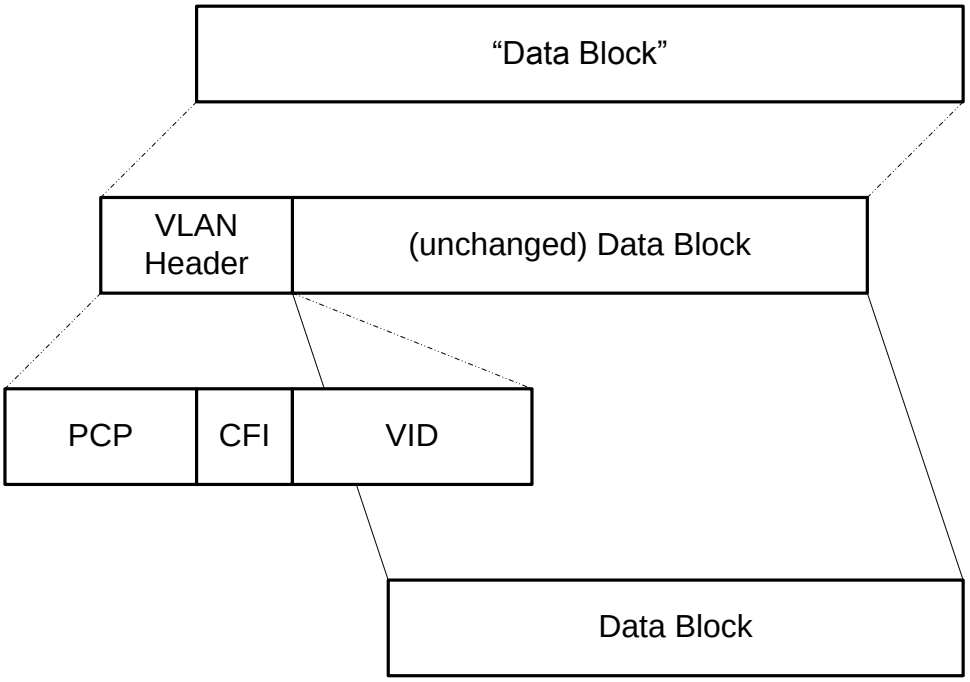


Figure H.5.3 VLAN Tagging

INDEX	NAME	SIZE	USE
1	PCP	3 bits	Priority Code Point
2	CFI	1 bit	Canonical Format Indicator
3	VID	12 bits	VLAN Identifier

Table H.5.3 VLAN Header Fields

1. PCP. Priority Code Point. Assignment of a priority to this data block, as per IEEE 802.1d. The determination of what priority to assign a given data block is outside the scope of this document. The implementation of methodologies to take advantage of those priorities is outside the scope of this document.

2. CFI. Canonical Format Indicator. As per IEEE 802.1q: if “0”, the Media Access Control (MAC) address is in canonical format (e.g., as for an Ethernet network); if “1”, the MAC is in non-canonical format (e.g., as for a Token Ring network). Determining which value to set this bit is outside the scope of this document.

3. VID. VLAN Identifier. Assignment of an identifier to allow physically separate parts of the same logical (virtual) network to be associated with each other by the intervening communications devices and protocols, as per IEEE 802.1q. The assignment and tracking of VIDs is outside the scope of this document. The reception of this VID will be used by the Remote as part of its Level 3 processing and routing, which is outside the scope of this document.

H.5.11 (NU) IP Packet Delivery

Once the IP Packet has been reconstructed, it is handed off to Level 3 processing for routing it towards its final destination.

H.6 (NU) Common Details

This section contains information that is the same for the parallel steps in all of the four Data Paths, and which have a sufficiently large amount of text that repeating it within each of the four steps would be more of a hindrance than a help, in conveying the actions necessary for that step.

H.6.1 (NU) Link Level

Figure H.6.1 shows an expansion of a Link Level Header. Tables H.6.1 through H.6.5 depict, in detail, the contents of each of the four Frame Types

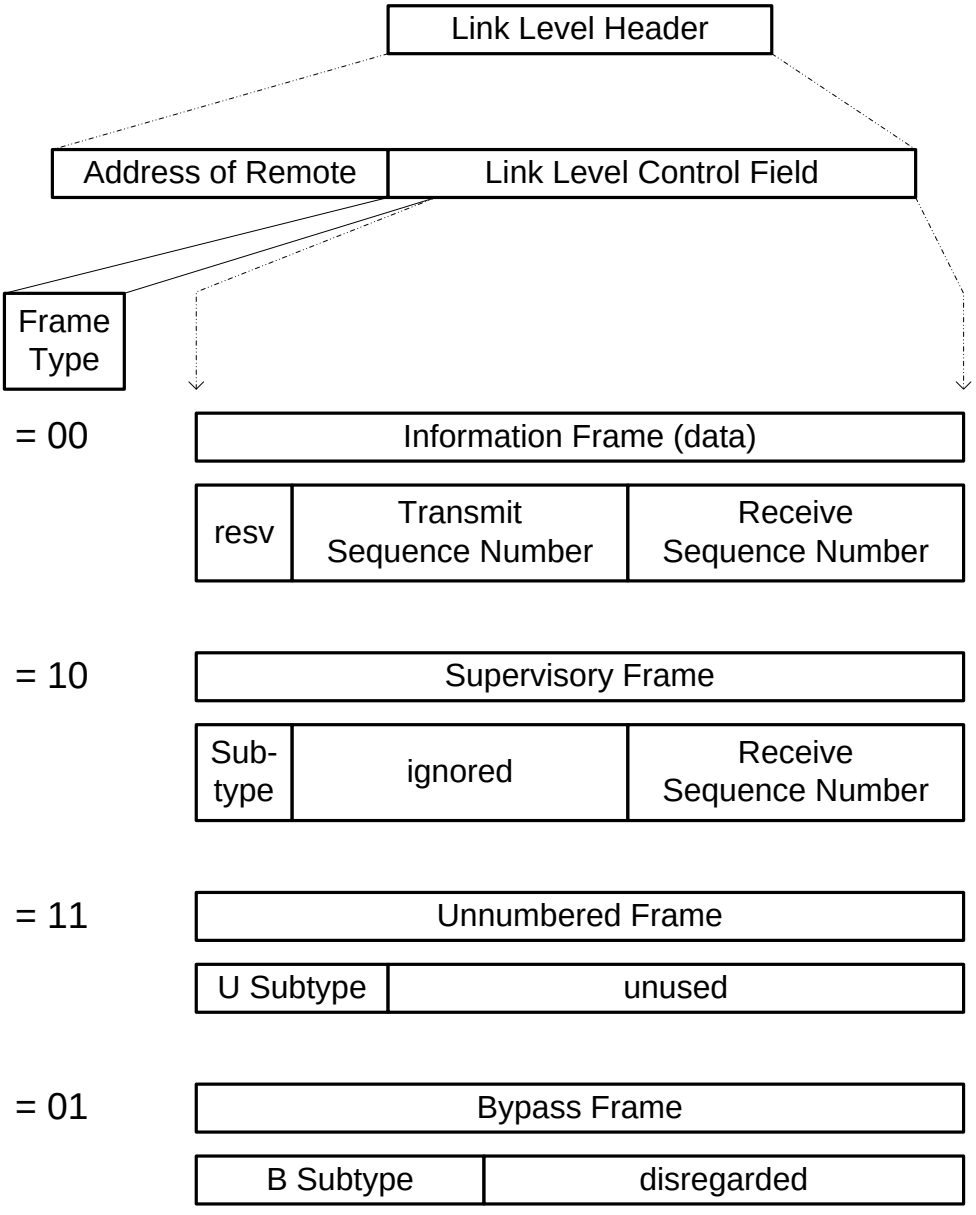


Figure H.6.1 Link Layer Header

INDEX	NAME	SIZE	USE
1	Address of Remote	2 bytes	Hub-assigned ID for a particular Remote
2	Frame Type	2 bits	one of four Frame Types
3	resv	2 bits	not used in Information Frames
4	Tx Seq. Number	14 bits	incremented for each new Info Frame
5	Rx Seq. Number	14 bits	recent Tx Num. received from other side
6	S Subtype	2 bits	for Supervisory Frame; see Table H.6.2
7	ignored	14 bits	not used in Supervisory Frames
8	Rx Seq. Number	14 bits	recent Tx Num. received from other side
9	U Subtype	6 bits	for Unnumbered Frame; see Table H.6.3
10	unused	24 bits	not used in Unnumbered Frames
11	B Subtype	12 bits	for Bypass Frame; see Table H.6.4
12	disregarded	18 bits	not used in Bypass Frames

Table H.6.1 Link Level Header

1. Address of Remote. This is the Remote's ID as assigned to it by the Hub at the time of Remote Acquisition (see Annex E for more details). Upon receipt, each Remote will examine this field and discard any frames which are not addressed to it. The Hub will use this field to help recombine data from each single Remote which arrive in different slots.

A special value, $HFFFF, is defined as being a broadcast, or addressed to everyone, flag. Every Remote will accept this address as being that of a frame it must keep and take action on.

2. Frame Type. There are four frame types: Information Frames, Supervisory Frames, Unnumbered Frames, and Bypass Frames. The remainder of the Link Level Header varies in its definition based on which Frame Type this frame belongs to.

Each different Frame Type is discussed separately in the following subsections.

H.6.1.1 (NU) Information Frames

For an Information Frame, the remainder of the Link Level Header is defined by Indexes 3, 4, and 5 in Table H.6.1.

3. resv. These bits of the Header are not used in Information Frames.

4. Tx Sequence Number. This is a monotonically increasing number assigned by the other side (peer) of this link. The number following the Transmit Sequence Number will be placed into the Receive Sequence Number field of the next frame being sent back to that peer.

5. Rx Sequence Number. This is used by the link's peer to inform this side of the next (as of the time of the transmission of this frame) frame that the peer expects to receive. Because Sequence Numbers are monotonically increasing, this also acts as an acknowledgement for all frames with Sequence numbers lower than this value (frames that were transmitted earlier).

Once the receiver has stored off the Tx and Rx Sequence Numbers for its internal use, the remainder of the HDLC Block, the "information" in this Information Frame, is passed on to the next step in its processing.

Likewise, once the transmitter has built the Link Level Header, and inserted the appropriate Tx and Rx Sequence numbers, it is prepended to the Information Frame and passed on to the next step in its processing.

H.6.1.2 (NU) Supervisory Frames

For a Supervisory Frame, the remainder of the Link Level Header is defined by Indexes 6, 7, and 8 in Table H.6.1

6. S Subtype. Table H.6.2 lists the defined subtypes for Supervisory Frames.

7. ignored. These bits of the Header are not used in Supervisory Frames.

8. Rx Sequence Number. This is used by the link's peer to inform this side of the next (as of the time of the transmission of this frame) frame that the peer expects to receive. Because Sequence Numbers are monotonically increasing, this also acts as an acknowledgement for all frames with Sequence numbers lower than this value (frames that were transmitted earlier).

SUBTYPE (includes Type)	HEX	MEANING
1000	$H80	Receiver Ready (RR)
1010	$HA0	Receiver Not Ready (RNR)
1011	$HB0	Selective Reject (SREJ)
1001	$H90	SREJ End Mark

Table H.6.2 Supervisory Frame Subtypes

Receive Ready. Tells the recipient that the sender is able to receive more frames. In particular, this negates the reception of an earlier RNR, and allows data transmission to resume.

Receive Not Ready.
Tells the recipient that there is a (non-fatal) problem on the peer, and that data transmission to that peer should be suspended until such time as the peer announces (via an RR) that it is again able to receive data.

Selective Reject.
Tells the recipient that there was a problem with the delivery of a (one or more) frame(s). The format for reporting these missed frames is depicted in Figure H.6.2.

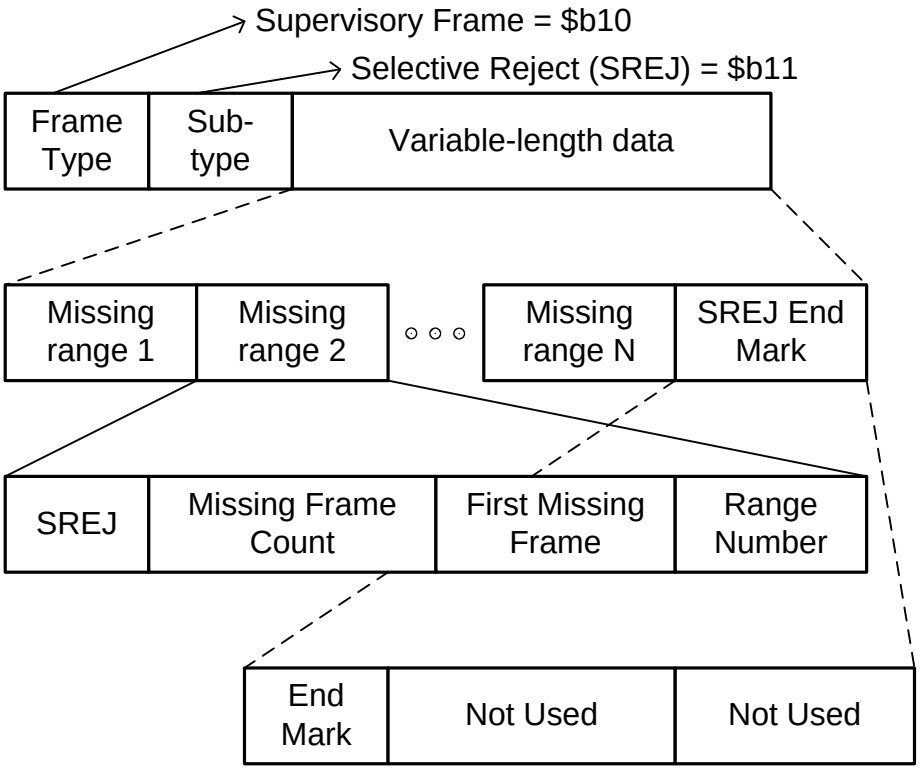


Figure H.6.2 Selective Reject

SREJ. Selective Reject, field length is two bits, value is $b11. Used semi-recursively to indicate the start of the next range of missing frames.

Missing Frame Count. Field length is fourteen bits. If there are N missing frames in this range, the value placed here is N-1.

First Missing Frame. Field length is fourteen bits. This is the Transmit Sequence Number of the first frame, in this range, that was not successfully received from the other side of the link.

For example, if the receiving side did not properly receive frames 1006 through 1010, a total of five missing, then the Missing Frame Count would be 4 – five minus one – and the First Missing Frame would be 1006. If only a single frame were missing, the Missing Frame Count would be zero.

Range Number. Field length is one byte. The cardinality of this range of missing frames.

End Mark. Field length is two bits, value is $b01. Used to flag that all of the ranges of missing fields have been defined.

Not Used. Both of these fields have lengths of fourteen bits. The contents of these fields are never read.

SREJ End Mark.
This subtype is recognized but ignored: its only use is within a Selective Reject sub-frame, so it should never be encountered as a stand-alone sub-frame. No action is taken upon its reception.

H.6.1.3 Unnumbered Frames

For an Unnumbered Frame, the remainder of the Link Level Header is defined by Indexes 9 and 10 in Table H.6.1.

9. U Subtype. Table H.6.3 defines subtypes for Unnumbered Frames.

10. unused. These bits of the Header are not used in Unnumbered Frames.

SUBTYPE (includes Type)	**HEX**	**MEANING**
1111 0110	$HF6	SABME
1100 0110	$HC6	Unnumbered Ack (UA)
1100 0000	$HC0	Unnumbered I (UI) *
1100 0001	$HC1	Unnumbered Datagram (UD)
1110 0001	$HE1	Frame Reject (FRMR)
1100 0010	$HC2	Disconnect (DISC)
1100 0100	$HC4	Disconnected Mode (DM)

Table H.6.3 Unnumbered Frame Subtypes

* Please note that the UI subtype, alone among all the Unnumbered Frames, does define and contain an Rx Sequence Number. This is used by the link's peer to inform this side of the next (as of the time of the transmission of this frame) frame that the peer expects to receive. Because Sequence Numbers are monotonically increasing, this also acts as an acknowledgement for all frames with Sequence numbers lower than this value (frames that were transmitted earlier).

The recipient's responses to these Unnumbered Frame Subtypes can best be presented by describing the recipient as a Finite State Table, and listing the action, and new state, that each subtype (trigger) invokes when it arrives. This Finite State Table is shown in Table H.6.4.

The four states that the recipient can be in are:

1. Disconnected (DISC). The recipient is not actively involved in data transfer, nor attempting to gain access to the network, but is listening to incoming traffic.
2. Opening (OPEN). The recipient is engaged in acquisition: requesting access to the network.
3. Data Transfer (DATA). The recipient is ready to actively send and receive data.
4. Closing (CLOSE). The recipient is engaged in shutting down and disconnecting from the network.

TRIGGER		**DISC**	**OPEN**	**DATA**	**CLOSE**		**ACTION STATE**
SABME		none	send ua	send disc	none		action
		DISC	OPEN	CLOSE	CLOSE		new state
UA		none	none	none	send disc		action
		DISC	DATA	DATA	CLOSE		new state
UI		none	none	none	none		action
		DISC	OPEN	DATA	CLOSE		new state
UD		none	none	none	none		action
		DISC	OPEN	DATA	CLOSE		new state
FRMR		none	none	send ua	none		action
		DISC	OPEN	CLOSE	CLOSE		new state
DISC		none	none	send ua	none		action
		DISC	OPEN	CLOSE	CLOSE		new state
DM		none	none	send disc	none		action
		DISC	OPEN	CLOSE	CLOSE		new state

Table H.6.4 Finite State Table for Unnumbered Frame Subtypes

H.6.1.4 (NU) Bypass Frames

Bypass Frames are so named because the Link Layer does not, after identifying them as such, do any further processing of them. Bypass Frames should be detected and processed by appropriate routines below (prior to) Link Level processing. Any Bypass Frames which do reach the Link Layer are treated as unknown message types. (Exact handling of unrecognized messages is implementation dependent and outside the scope of this document.)

For a Bypass Frame, the remainder of the Link Level Header is defined by Indexes 11 and 12 in Table H.6.1.

11. B Subtype. Table H.6.5 defines subtypes for Bypass Frames.

12. disregarded. These bits of the Header are not used in Bypass Frames.

SUBTYPE (includes Type)	**HEX**	**MEANING**
0111 1111 1111 01	$H7FF4	Acquisition
0111 1111 1111 10	$H7FF8	Keep-Alive
0111 1111 1111 11	$H7FFC	Pseudo-Broadcast
0111 1111 1110 11	$H7FEC	Encryption OOB Message
0111 1111 1110 00	$H7FE0	OOB Chunked Message
0111 1111 1110 01	$H7FE4	Remote OOB Message

Table H.6.5 Bypass Frame Subtypes

Acquisition: Frames carrying messages for bringing a new Remote onto (or back onto) the network will use this subtype.

Keep-Alive: Within normal networks, this frame subtype is sent whenever either side has no actual data, or other messages, that need to be transmitted, so that the receiving side will know the transmitting side is still alive. However, a network under TRANSEC conditions will always be sending data in both directions – if there is no “real” data to send, then the TRANSEC procedures will generate “fake” data and send that. Therefore there should never an occasion to use this frame subtye.

Pseudo-Broadcast: Frames carrying messages destined for all Remotes, for example timeplans, will use this subtype. This frame subtype should only appear in downstream traffic.

Encryption OOB Message: Frames carrying messages related to encryption will use this subtype. Examples of this include: certificates, keyroll requests, acknowledgements. This frame subtype can appear in both upstream and downstream traffic; but see also “OOB Chunked Message”.

OOB Chunked Message: If a message (e.g., an Encryption Certificate) is too large to fit into a single TDMA “slot”, it can be broken up into “chunks” and sent inside multiple slots. In that case, the frames involved all have this subtype. This frame subtype should only appear in upstream traffic.

Remote OOB Message: Frames carrying messages containing status information about itself from a Remote to the Hub will use this subtype. This frame subtype should only appear in upstream traffic.

H.6.2 (NU) Amplitude and Phase Details

This section contains the descriptions of the three bit mapping schemas: BPSK, QPSK, and 8PSK.

H.6.2.1 (NU) BPSK

The Hub shall employ conventional Gray-coded BPSK modulation with absolute mapping (no differential coding). Bit mapping into the BPSK constellation shall follow Figure H.6.3. The normalized energy per symbol shall be equal to $\rho^2 = 1$.

One bit from the FEC Frame bitstream is mapped to each BPSK symbol. The symbol mapping is explicitly given in Table H.6.6.

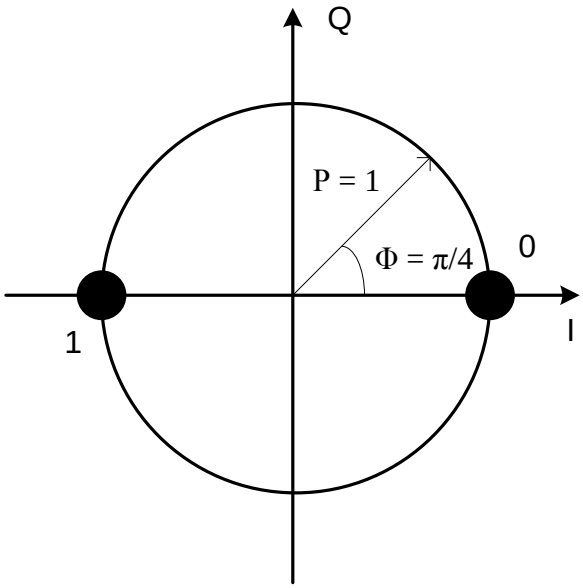


Figure H.6.3 BPSK Constellation

din	I	Q
{xx0}	+1	0
{xx1}	-1	0

Table 6.6 BPSK Symbol Mapping

H.6.2.2 (NU) QPSK

The Hub shall employ conventional Gray-coded QPSK modulation with absolute mapping (no differential coding). Bit mapping into the QPSK constellation shall follow Figure H.6.4. The normalized energy per symbol shall be equal to $\rho^2 = 1$.

Two consecutive bits from the FEC Frame bitstream are mapped to each QPSK symbol. The symbol mapping is explicitly given in Table H.6.7.

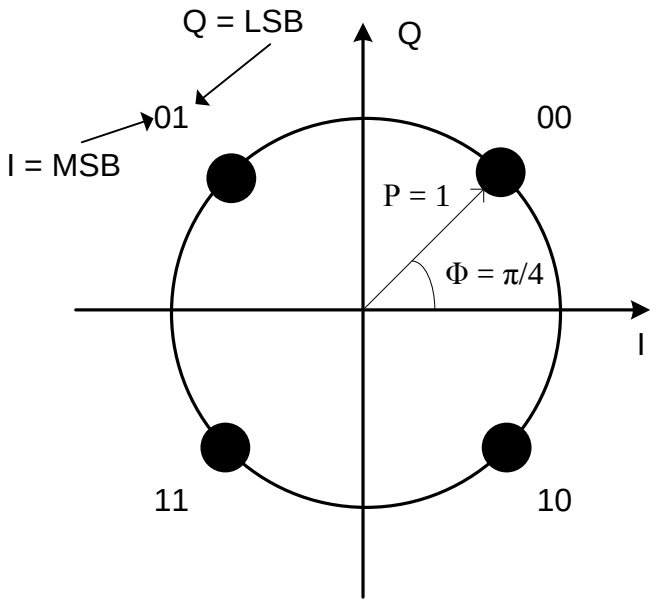


Figure H.6.4 QPSK Constellation

din	I	Q
00	+1	+1
01	-1	+1
11	-1	-1
10	+1	-1

Table 6.7 QPSK Symbol Mapping

H.6.2.3 (NU) 8PSK

The Hub shall employ conventional Gray-coded 8PSK modulation with absolute mapping (no differential coding). Bit mapping into the QPSK constellation shall follow Figure H.6.5. The normalized energy per symbol shall be equal to $\rho^2 = 1$.

Three consecutive bits from the FEC Frame bitstream are mapped to each 8PSK symbol. The symbol mapping is explicitly given in Table H.6.8.

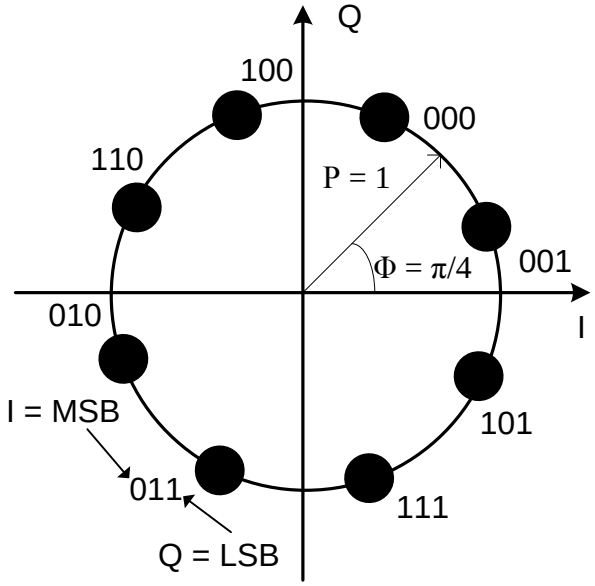


Figure H.6.5 8PSK Constellation

din	I/Q angle
000	67.5
100	112.5
110	157.5
010	202.5
011	247.5
111	292.5
101	337.5
001	22.5

Table 6.8 8PSK Symbol Mapping

H.6.3 (NU) Signal Spectrum

For a roll-off factor α = 0.20, the signal spectrum at the modulator output shall be in accordance with EN 300 421.

Figure H.6.6 gives a template for the signal spectrum at the modulator output.

Figure H.6.7 gives a mask for the group delay modulator filter.

The points A to S, as shown in both Figure H.6.6 and Figure H.6.7, are defined in table H.6.9. Note that there are no points labeled “I” or “O”.

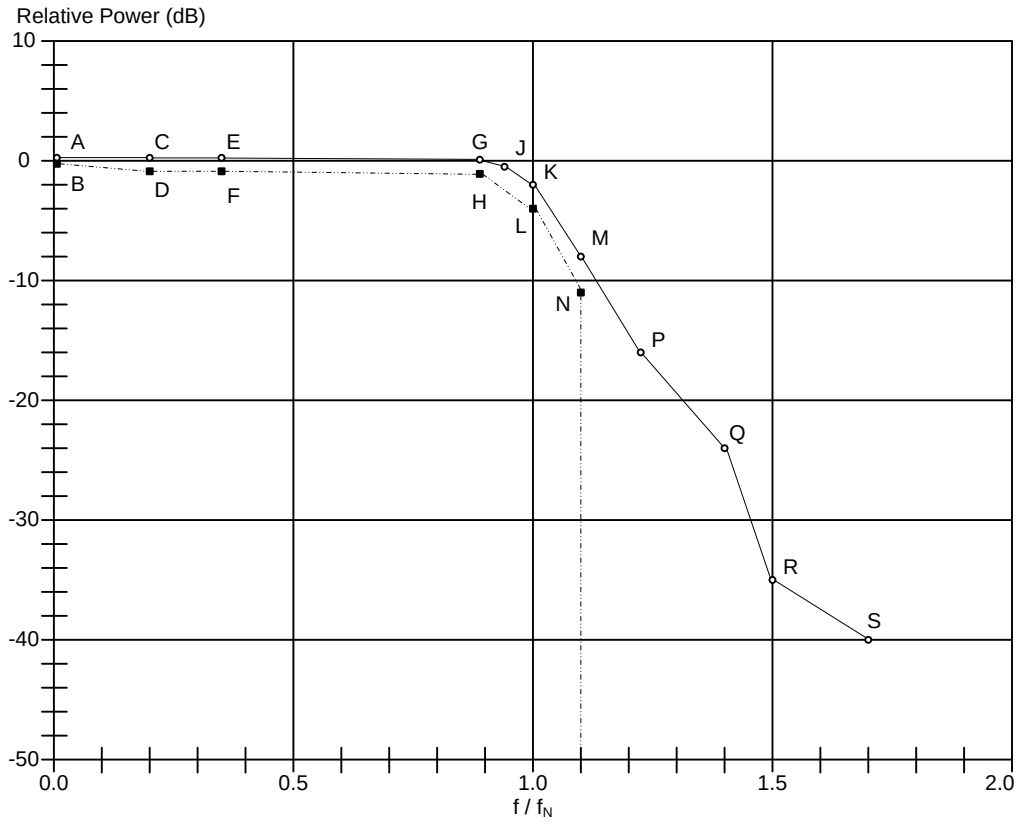


Figure H.6.6 Signal Spectrum Template

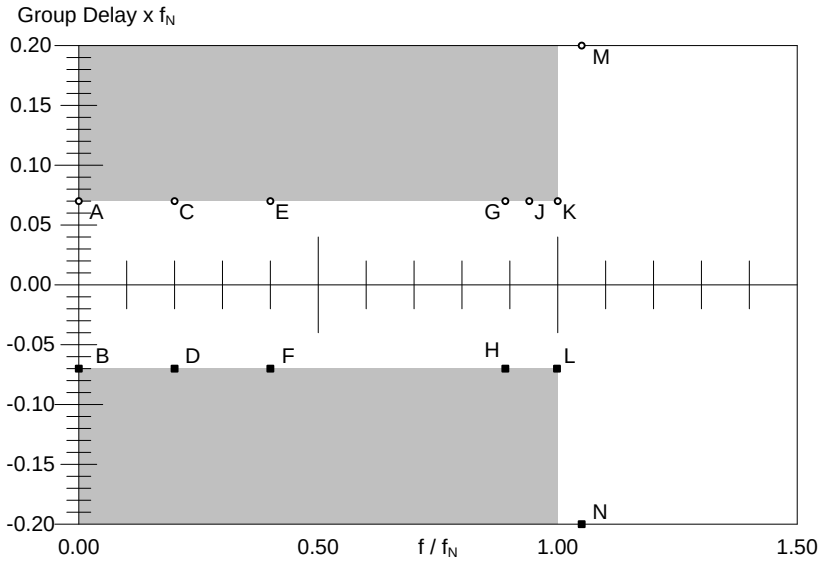


Figure H.6.7 Modulator Filter Group Delay Template

Point	Frequency for α = 0.20	Relative Power (dB)	Group delay
A	0.0 f_N	+0.25	+0.07 / f_N
B	0.0 f_N	-0.25	-0.07 / f_N
C	0.2 f_N	+0.25	+0.07 / f_N
D	0.2 f_N	-0.40	-0.07 / f_N
E	0.4 f_N	+0.25	+0.07 / f_N
F	0.4 f_N	-0.40	-0.07 / f_N
G	0.89 f_N	+0.15	+0.07 / f_N
H	0.89 f_N	-1.10	-0.07 / f_N
J	0.94 f_N	-0.50	+0.07 / f_N
K	1.0 f_N	-2.00	+0.07 / f_N
L	1.0 f_N	-4.00	-0.07 / f_N
M	1.11 f_N	-8.00	-
N	1.11 f_N	-11.0	-
P	1.23 f_N	-16.00	-
Q	1.4 f_N	-24.00	-
R	1.5 f_N	-35.00	-
S	1.7 f_N	-40.00	-

Table H.6.9 Definition of Points

H.6.4 (NU) Packet Level Encryption

These details are common to Packet Level Encryption (and Decryption) along all four of the data paths.

H.6.4.1 (NU) Encryption of Data Packets

Figure H.6.7 shows the transformation between the clear text Data Block and the Encrypted Data plus an Encryption Header. Table H.6.10 depicts the contents of the Encryption Header in detail. The entirety of this construct – Header plus Data – is referred to as a single entity Packet Level Encrypted Data Block.

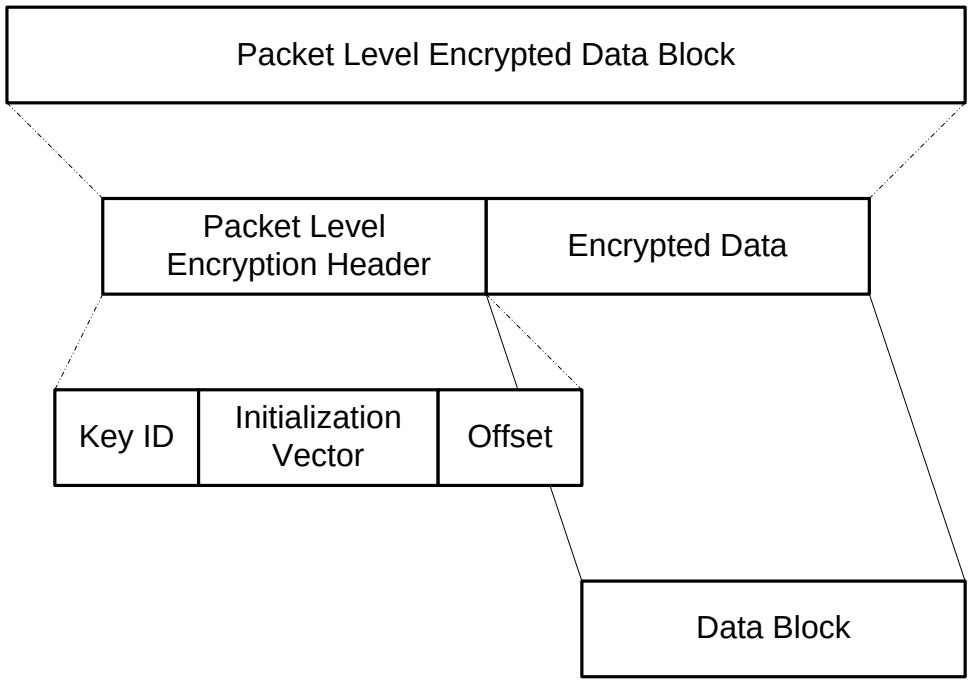


Figure H.6.7 Packet Level Encryption and Decryption

INDEX	NAME	SIZE	USE
1	Key ID	2 bits	index into current key table
2	Initialization Vector	10 bits	for AES algorithm
3	Encoding Offset	4 bits	amount of padding included

Table H.6.10 Packet Encryption Header Fields

1. Key ID. The values of this field, 0 through 3, are used to select an entry from the Key Ring as shown in Tables I.3 through I.6 (see section H.6.4.2, "Updating of Encryption Keys", below).

2. Initialization Vector. These ten bits are used by the sender as input into the AES algorithm for encrypting, and then passed (in clear in this header) to the receiver for use by the AES algorithm in decrypting. They must be different for each packet.

3. Encoding Offset. This field is set to a value between 0 and 15, specifying how many bytes of Padding the AES algorithm requires for a prefix to this Data Block's encrypted form. For the AES CFB algorithm, this value is always zero.

H.6.4.2 (NU) Exchange of Encryption Certificates

This step is required prior to the use of Packet Level Encryption. After all other acquisition steps have been taken (see Annex E, Link Establishment), and the new Remote is no longer using an Acquisition slot for its responses to the Hub, and after TRANSEC has been established for this link, the Hub and this Remote must exchange Certificates for validating their subsequent Key Exchanges and Encryption usage.

Figure H.6.8 depicts this exchange.

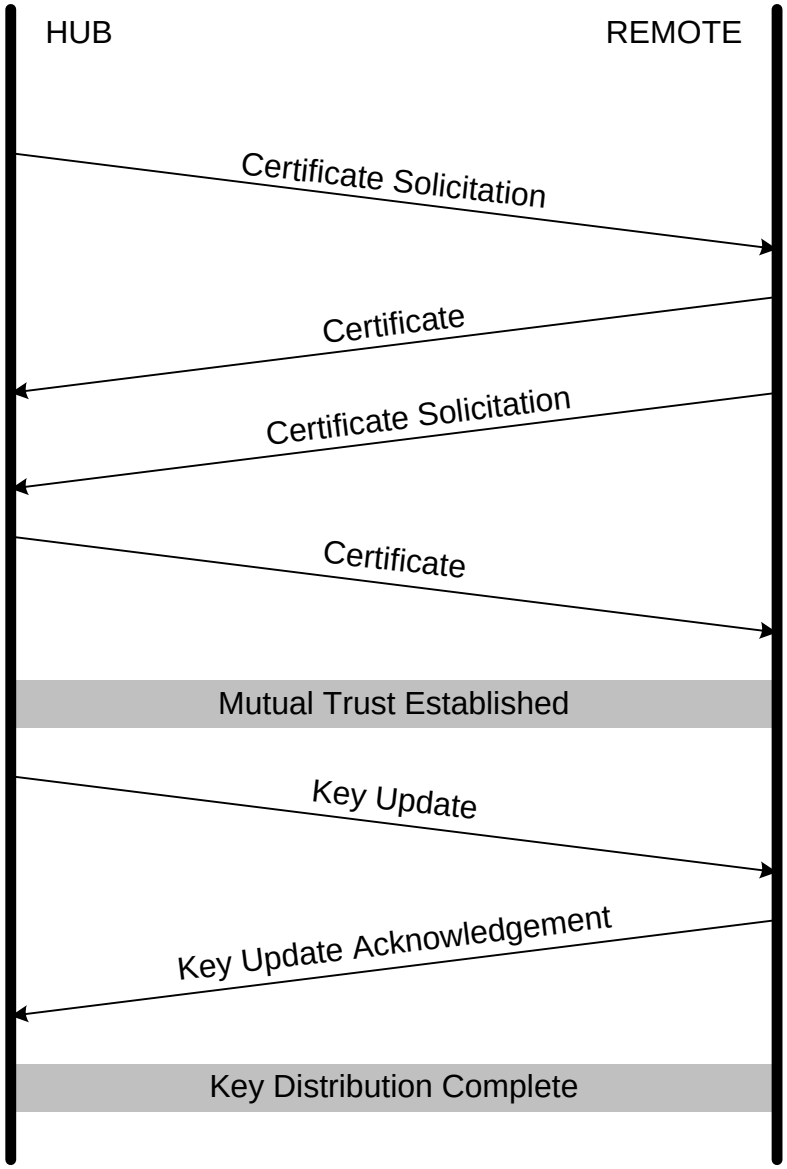


Figure H.6.8 Certificate Exchange

The Hub's next step is to send a Certificate Solicitation message to the new Remote. The Remote responds with its Certificate, and then sends its own Certificate Solicitation message to the Hub.

The Hub must be able to validate and establish a chain of trust, upon the receipt of a certificate from a new Remote, based on the contents of that certificate. X.509 certificates and validation methodologies are to be used for this step. If it does validate the remote, then it will respond to the Remote's Certificate Solicitation message by sending its own Certificate. If however it does not validate the Remote, then the Remote will not be allowed to enter the network.

The Remote, upon receiving the Hub's Certificate, must likewise validate that Certificate (this step prevents a Remote from being "hijacked" by a hostile Hub).

Once both peers – the Hub and the new Remote – are confident about each other, the Hub needs to provide the Remote with the Encryption Key for their link through a Key Update message. . The exchange of X.509 certificates allows the Hub to encrypt the Encryption Key according to the Remote's public key information. The Remote receives the Encryption Key, stores it for use, and returns a Key Update Ack message.

H.6.4.2.1 (NU) Certificate Solicitation Format

The format for a Certificate Solicitation message is shown in Table H.6.11.

IX	NAME	SIZE bytes	XDR	REPEAT	USE
1	Modem ID	4	int	once	type of machine code is running on
2	Network ID	4	int	once	which network
3	Rx Session State	4	int	once	status of peer node's sending
4	Rx Session ID	4	uns long	once	node's ID for incoming exchanges
5	Peer Tx Session ID	4	uns long	once	copied from latest peer exchange
6	Tx Session State	4	int	once	status of this node's sending
7	Tx Session ID	4	uns long	once	node's ID for outgoing exchanges
8	Peer Rx Session ID	4	uns long	once	copied from latest peer exchange
9	Rx Certificate State	4	int	once	status of peer's certificate
10	Rx Certificate ID	4	uns long	once	abbreviation of peer's certificate
11	Tx Certificate ID	4	uns long	once	abbreviation of node's certificate
12	Algorithms Length	4	uns long	once	number of Algorithms following
13	Algorithms	4*N	uns long	N	recognized encryption algorithms
14	Transport set count	4	int	once	number of Transport sets following
15	Transport ID sets	2	IIOB	twice	start and stop values for set
16	Message Length	4	uns long	once	length of Message following
17	Message	N	byte	N	optional text string

Table H.6.11 Certificate Solicitation

1. Modem ID: A combination of a numeric representation of the Modem Model and a Serial Number. Modem Models are provided as a vendor-specific number, which must be 14 bits or less in size (between 0 and 16,383). This number is left-shifted 18 places and ORed with the 18 least significant bits of the Serial Number of the particular modem at this node. Note that there may be some overlap of Modem Model numbers between different vendors.

2. Network ID: assigned during the installation of the Hub; provided to each Remote as part of its installation.

3. Rx Session State: There are three possible states that a session can be in:
    0 ::= Session Up
    1 ::= Session Down
    2 ::= Session Opening

4. Rx Session ID: The initial value is set to a random number when the node comes up. Thereafter the value is incremented once every constant time period (e.g., every fifteen minutes). The time period is configured during system installation.

5. Peer Tx Session ID: If there have been no exchanges from the peer, then this field has a value of zero. Otherwise the received Certificate Solicitation from the peer will contain that node's TX Session ID, and it will be copied into this field.

6. Tx Session State: There are three possible states that a session can be in:
    0 ::= Session Up
    1 ::= Session Down
    2 ::= Session Opening

7. Tx Session ID: The initial value is set to a random number when the node comes up. Thereafter the value is incremented once for each Key Update. Note that if the link goes down and the Remote needs to be re-acquired, there will be a Key Update as part of that Link Establishment activity (see Annex E), and thus this value will also be incremented.

8. Peer Rx Session ID: If there have been no exchanges from the peer, then this field has a value of zero. Otherwise the received Certificate Solicitation from the peer will contain that node's RX Session ID, and it will be copied into this field.

9. Rx Certificate State: There are two possible states that a Certificate can be in:
    0 ::= Have Not Received Certificate
    1 ::= Have Received Certificate

10. Rx Certificate ID: If there have been no exchanges from the peer, then this field has a value of zero. Otherwise the peer's certificate is run through ASN1_digest() – an Open SSL Library standard hash routine – to produce a twenty byte long abbreviation. The first four bytes of this are then placed into an unsigned long, with byte 0 being placed in the least significant eight bits, through byte 3 being placed in the most significant eight bits.

11. Tx Certificate ID: The node's certificate is run through ASN1_digest() – an Open SSL Library standard hash routine – to produce a twenty byte long abbreviation. The first four bytes of this are then placed into an unsigned long, with byte 0 being placed in the least significant eight bits, through byte 3 being placed in the most significant eight bits.

12. Algorithms Length: This is a count of how many Algorithms (see 13., below) will be listed.

13. Algorithms: This is a list of the encryption algorithms which this node is capable of using. The list places each algorithm's identifying number (see Table H.6.12) into a separate unsigned long field.

ENCRYPTION ALGORITHM	ID NUMBER
RSA	500
3-DES CBC	500
AES 256 CBC	507
AES 256 CFB	508
AES 256 CTR	509

Table H.6.12 Encodings for Encryption Algorithms

14. Transport set count: This is a count of how many Transport Sets (see 15., below) will be listed. For Packet Level Encryption, only one Transport Set is allowed, so the value will always be "1".

15. Transport ID sets: Each set is a pair of numbers, each encoded as Integer In One Byte (IIOB), containing the first number in a range and then the last number in that range. There can be multiple ranges included. However, under TRANSEC, the only Transport IDs actually available are listed in TableH.6.13. Furthermore, of these, only the Packet Encryption Unicast is usable for Packet Level Encryption certificates. Therefore, for Packet Level Encryption, the single range will have a start value of "0" and a stop value of "0".

TRANSPORT ID	ID NUMBER
Packet Encryption Unicast	0
TRANSEC Master TX Netkey	2
TRANSEC Master RX Netkey	3

Table H.6.13 Transport IDs

16. Message Length: This is a count of how many bytes are in the Message field (see 17., below). As the appending of a text message is an optional action, this value may be zero.

17. Message: An optional text message whose purpose might be for something along the lines of human-level status notification, condition logging, etc. How this message is created, what is done with it on the receiving side, and even whether it is included, are all outside the scope of this document.

IH.6.4.2.2 (NU) Certificate Format

The Certificate is formatted in compliance with the X.509 Standard.

H.6.4.2.3 (NU) Key Update Format

The format for a Key Update message is shown in Table H.6.14. Note that Key Update messages are only sent from the Hub to the Remote.

IX	NAME	SIZE bytes	XDR	REPEAT	USE
1	Message Type	4	int	once	identifies Key Update message
2	Payload Length	4	uns long	once	length of Payload following
3	Payload	N	byte	N	encrypted new key
4	Algorithm Length	4	uns int	once	length of Algorithm following
5	Algorithm	N^	byte	N^	name of algorithm
6	Signature Length	4	uns int	once	length of Signature following
7	Signature	N	byte	N	validation of new key
8	Network ID	4	uns long	once	which network

Table H.6.14 Key Update

1. Message Type: Identifies this message as a Key Update. Always set to “1012”

2. Payload Length: This is a count of how many bytes are in the Payload field (see 3., below).

3. Payload: This field contains the encrypted contents of the Key Update Command. They have been encrypted first with the Remote's public RSA key, and then secondly with the Host's private RSA key.

Inside the encryptions, the Key Update Command has the format shown in Table H.6.15.

NAME	SIZE bytes	XDR	REPEAT	USE
Tx Session ID	4	uns long	once	node's ID for outgoing exchanges
Rx Session ID	4	uns long	once	node's ID for incoming exchanges
Command Count	4	uns long	once	number of "commands" following
Commands	varies		CmdCnt	Command Count of these

Table H.6.15 Key Update Command

Tx Session ID: The initial value is set to a random number when the node comes up. Thereafter the value is incremented once for each Key Update. Note that if the link goes down and the Remote needs to be re-acquired, there will be a Key Update as part of that Link Establishment activity (see Annex E), and thus this value will also be incremented.

Rx Session ID: The initial value is set to a random number when the node comes up. Thereafter the value is incremented once every constant time period (e.g., every fifteen minutes). The time period is configured during system installation.

Command Count: The number of commands contained in the rest of this block

Command: Each command has the format shown in Table H.6.16.

NAME	SIZE bytes	XDR	REPEAT	USE
Transport set count	4	int	once	number of Transport sets following
Transport ID sets	2	IIOB	twice	start and stop values of set
Active Key Index	1	UIIOB	once	currently active key ring index
New Key Count	4	uns long	once	number of new keys following
New Key	varies		NK Cnt	New Key Count of these

Table H.6.16 Command Format

Transport set count: Transport set count: This is a count of how many Transport Sets (see next field) will be listed. For Packet Level Encryption, only one Transport Set is allowed, so the value will always be “1”.

Transport ID sets: Each set is a pair of numbers, each encoded as Integer In One Byte (IIOB), containing the first number in a range and then the last number in that range. There can be multiple ranges included. However, under TRANSEC, the only Transport IDs actually available are listed in TableH.6.13. Furthermore, of these, only the Packet Encryption Unicast is usable for Packet Level Encryption certificates. Therefore, for Packet Level Encryption, the single range will have a start value of “0” and a stop value of “0”.

Active Key Index: In the Key Ring, which entry is currently active. Has a valid value of 0 through 3.

New Key Count: The number of new keys included in this command.

New Key: Each New Key has the format shown in Table H.6.17.

NAME	SIZE bytes	XDR	REPEAT	USE
Key Ring Index	1	UIIOB	once	slot in Key Ring for this new key
Valid	1	BIOB	once	is this a valid new key
Key Length	4	uns int	once	length of actual key
Key	N	bytes	N	the new key

Table H.6.17 New Key Format

Key Ring Index: The Remote's Key Ring has four slots, 0 through 3. This field tells the Remote which slot to store this new key into. Stored as an Unsigned Int In One Byte (UIIOB).

Valid: This should always be TRUE. Stored as a Boolean In One Byte (BIOB).

Key Length: The length in bytes of the actual key.

Key: The actual new key.

4. Algorithm Length: This is a count of how many bytes are in the Algorithm field (see 5., below).

5. Algorithm: The name of the encryption algorithm. Valid names are listed in Table H.6.18.

^ Note that this field is buffered out to an exact four byte boundary, by appending meaningless bytes until the next boundary is reached. The value in field 4., "Algorithm Length" depicts the exact length of the contents of this field. To calculate the number of buffer bytes (if any) following the contents, use the formula:

Buffer byte count = (4 – (Algorithm Length MOD 4)) MOD 4

ALGORITHM NAMES
"ALG_3DES_CBC"
"ALG_AES_CBC"
"ALG_AES_CFB"
"ALG_AES_CTR"

Table H.6.18 Algorithm Names

6. Signature Length: This is a count of how many bytes are in the Signature field (see 7., below).

7. Signature: The first twenty bytes of the new key are taken and encrypted through first the destination Remote's Public RSA key, and then through the Hub's Private RSA key.

8. Network ID: assigned during the installation of the Hub; provided to each Remote as part of its installation.

H.6.4.2.4 (NU) Key Update Acknowledgement

Note that this message type is only sent from the Remote to the Hub.

This message has the exact same format as the Certificate Solicitation message. The Acknowledgement is explicitly contained in the Transport ID Sets field (field 15), by having the Transport ID Set contain both of the valid values for TRANSEC: “2” and “3”.

If this message is being used as a Certificate Solicitation message, then either the Peer Tx Session ID (field 5), or the Peer Rx Session ID, or both, will be zero: indicating that a Certificate has not been received and successfully processed. If both of these fields do have values, and if they match what the Hub expects to see from this Remote, then this message is a Key Update Acknowledgement.

H.6.4.3 (NU) Updating of Encryption Keys

For security reasons, the Encryption key held in common between the Hub and an individual Remote should be changed out periodically. The period selected, and the means of defining it at the Hub, are outside the scope of this document. The Hub and each Remote will keep multiple Encryption keys defined, thus allowing a mechanism for switching to a “next” key, without having to immediately precede a key “roll” with a key transfer.

The Key Ring is a table with four rows, each with three entries. Table H.6.19 defines the contents of each single row of the Key Ring. Tables H.6.20, H.6.21, and H.6.22 show how the contents of the table are modified to support a Key update.

Each of the four rows of the Key Ring has a symmetric key – each of which is different. At the time that the Table H.6.20 version of the Key Ring is in use, the Encryption Header Key ID fields will all have a value of “10”, indicating that the symmetric key stored at row INDEX=2 is the one that was used for encryption, and is to be used for decryption. The symmetric key stored at row INDEX=3 will be the next one used, after the current one is finished. The symmetric key stored at row INDEX=0 is a far future key. And the symmetric key stored at row INDEX=1 is the most recently exhausted – no longer in use – key.

When it is time to update the key, a new key is generated by the Hub and stored in its Key Ring table, overwriting the Fallow-2 entry (which held the most recently exhausted key). This is shown at INDEX= 2 in Table H.6.21.

Then, the Hub transmits this new symmetric key to the Remote. This transfer is RSA encrypted: using that Remote’s public key and with a signature which is created by using the Hub’s private key.

Next, but not necessarily immediately thereafter, the Key Ring Value for each entry is incremented (with rollover from 11 to 00), resulting in the Key Ring depicted in Table H.6.22. This is the actual rollover into using a new (the next) key. There is no explicit notification, from the Hub to the Remote, that the rollover has been initiated; rather, the Hub now uses the "new" current Key ID ("11" now that the Table H.6.22 version of the Key Ring is in use) to pick the symmetric key for use in the encryption, and includes that value in the Encryption Block Header. The Remote reads this value from the Encryption Block Header, uses it to index into the Key Ring and retrieve the proper symmetric key for use in decrypting the Block, and begins using that index to select the symmetric key for use when encrypting its own upstream Encryption Blocks.

INDEX	NAME	USE
1	Activity	Current, Next, Fallow-1, or Fallow-2
2	Active Key Value	00, 01, 10, or 11
3	Key	Symmetric Key: generated and stored

Table H.6.19 Encryption Key Ring Entries

INDEX	Activity	Value	Key
1	Fallow-1	00	future symmetric key
2	Fallow-2	01	symmetric key no longer in use
3	Current	10	symmetric key currently in use
4	Next	11	symmetric key to shift to next

Table H.6.20 Encryption Key Ring, pre-Rolling

INDEX	Activity	Value	Key
1	Fallow-1	00	future symmetric key
2	Fallow-2	01	newly generated key
3	Current	10	symmetric key currently in use
4	Next	11	symmetric key to shift to next

Table H.6.21 Encryption Key Ring, new Key inserted

INDEX	Activity	Value	Key

1	Fallow-1	01	newly generated key
2	Fallow-2	10	symmetric key rolled away from
3	Current	11	symmetric key rolled into (now current)
4	Next	00	future symmetric key (now next)

Table H.6.22 Encryption Key Ring, post-Rolling

[This page intentionally left blank.]

ANNEX I
TRANSEC

CONTENTS

I.1 (NU) Overview

Transmission Security (TRANSEC) provides a layer of security against information gathering by an adversary. In addition to packet level encryption, TRANSEC protects against an adversary obtaining statistical information about network traffic patterns. For example when TRANSEC is enabled, details regarding traffic volume to and from a particular Remote cannot be determined.

Note that during Remote Acquisition [For details on Remote Acquisition, see Annex E] the initial steps are conducted in the clear because the remote has not yet completed authentication and does not possess the network key. Once authentication is complete, all communications between the Remote and the Hub are subject to the TRANSEC methods described below.

I.2 (NU) Downstream TRANSEC Encryption

Figure I.1 shows the transformation of a Fixed Length Content – made up of variable length Data Fragments – into TRANSEC Encrypted Data plus a TRANSEC Encryption Header. Table I.1 depicts the contents of the TRANSEC Encryption Header in detail. After this step, the entirety of this construct – Header plus Data – is referred to as a single entity TRANSEC Block.

The actual encryption of each Fixed Length Content adheres to the strictures of the X.509 standard.

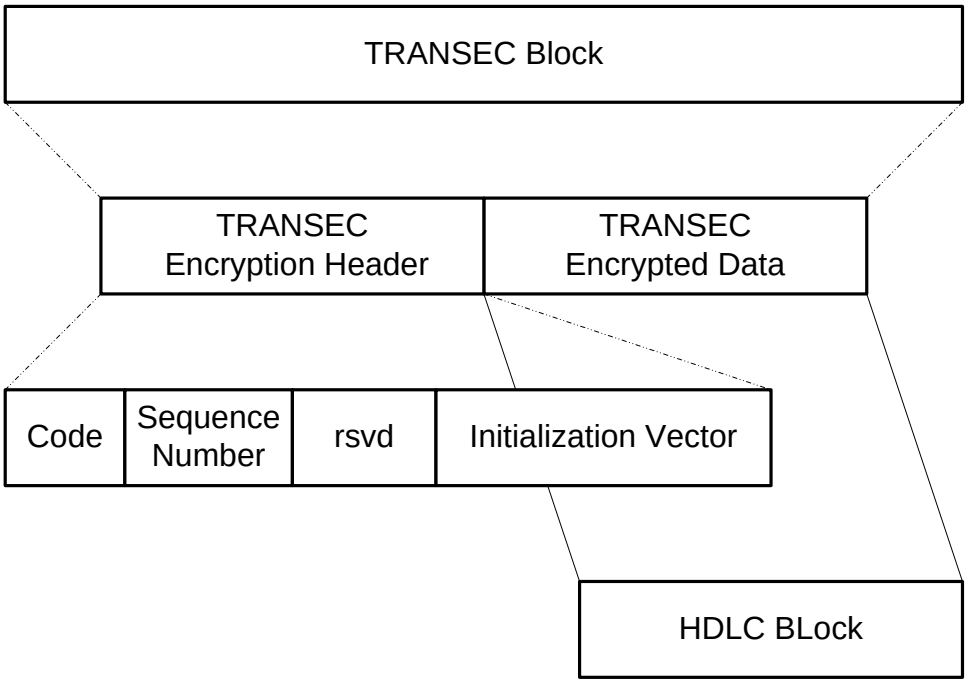


Figure I.1 Downstream TRANSEC Encryption

INDEX	NAME	SIZE	USE
1	Code	1 byte	which Key Ring entry is in use
2	Sequence Number	2 bytes	incremented with each TRANSEC Block
3	rsvd	1 byte	reserved
4	Initialization Vector	16 bytes	unsigned random 128 bits

Table I.1 TRANSEC Downstream Encryption Header Fields

1. Code. This field indicates if the frame is encrypted or not, and if encrypted indicates the entry within the Key Ring to be utilized for this frame. If the ‘encryption’ bit is on (True) then this frame is encrypted; if it is off (False) then the frame is in clear. The encryption bit is found by using a $H01 mask on the Code byte.

The ‘key ring’ bits provide the values and meanings shown in Tables I.3 through I.6 (see section I.7, “Updating of TRANSEC Keys”, below). The key ring value is found by using a $H06 mask on the Code byte, and then performing a single right-shift to obtain values in the range of 0 through 3 (bit values 00, 01, 10, and 11).

2. Sequence Number. This field is incremented by the Hub for each TRANSEC block.

3. Reserved for future use.

4. Initialization Vector. This is utilized by the AES encryption algorithm. A new IV is generated for each segment. The first IV is generated from the cipher text of the initial Known Answer Test (KAT) conducted at system boot time. Subsequent IVs are taken from the last 128 bits of the cipher text of the previously encrypted segment. IVs are continuously updated regardless of key rotations and they are independent of the key rotation process. They are also continuously updated regardless of the presence of user traffic since the filler segments are encrypted.

In the event that the Hub does not currently have any data to broadcast downstream, it must create an artificial and meaningless TRANSEC Block, and transmit that, repeating this step as necessary until it once again does have true data to broadcast. This action is required to prevent hostile eavesdroppers from deducing meta-information about the transmissions, based on a pattern of transmission peaks and lulls.

The contents of an artificial TRANSEC block must be random and different for each such artificial block. This action is required to prevent hostile eavesdroppers from detecting such artificial blocks by noticing a repeating pattern (which a fixed artificial block would generate). The contents, while random and meaningless, are nonetheless run through the encryption algorithm.

The Header for an artificial TRANSEC block must be valid. That entails: selecting a valid Key Ring entry to use for the encryption – and recording that in the Code field; incrementing the Sequence Number; and setting the Initialization Vector to the previous TRANSEC block’s final 128 bits of cipher text.

Once the Block has been encrypted, processing continues with HDLC Encoding, as described in Annex H, section H.2.7, “Bit Scrambling”.

I.3 (NU) Downstream TRANSEC Decryption

The remote for which this TRANSEC Block is designated will use the TRANSEC Block Header (see Figure I.1) to provide the information needed by the AES encryption algorithm for decrypting this Block.

The actual decryption of each TRANSEC Block adheres to the strictures of the X.509 standard.

The Sequence Number is used to verify that each TRANSEC Block is being handled (decrypted) in order. If TRANSEC processing detects that a Block is missing (a Sequence Number has been skipped), it will not react to this condition (other than perhaps logging an event). Rather, it will depend on other protocol levels to detect and recover the missing block.

The Code is used for two purposes. First, if the 'encryption' bit is False, then this Block is not encrypted, and the decryption step is skipped. Otherwise, if this bit is True, meaning that the Block has been encrypted, the 'key ring' bits are used to determine which Key Ring entry was used to encrypt this block. See Tables I.3 through I.6 (in section I.7, below) for the interpretation of the 0 through 3 values of these two bits.

The encryption bit is found by using a $H01 mask on the Code byte.

The key ring value is found by using a $H06 mask on the Code byte, and then performing a single right-shift to obtain values in the range of 0 through 3 (bit values 00, 01, 10, and 11).

The Initialization Vector is a necessary component for starting the AES algorithm on a new Block.

Once the Block has been decrypted, processing continues with the partitioning of the Burst, as described in Annex H, section H.3.6, "HDLC Decoding".

I.4 (NU) Upstream TRANSEC Encryption

Figure I.2 shows the transformation of a Fixed Length Content – made up of one fixed length Data Fragments – into TRANSEC Encrypted Data plus a TRANSEC Encryption Header. Table I.2 depicts the contents of the TRANSEC Encryption Header in detail. After this step, the entirety of this construct – Header plus Data – is referred to as a single entity TRANSEC Block.

The actual encryption of each Data Fragment adheres to the strictures of the X.509 standard.

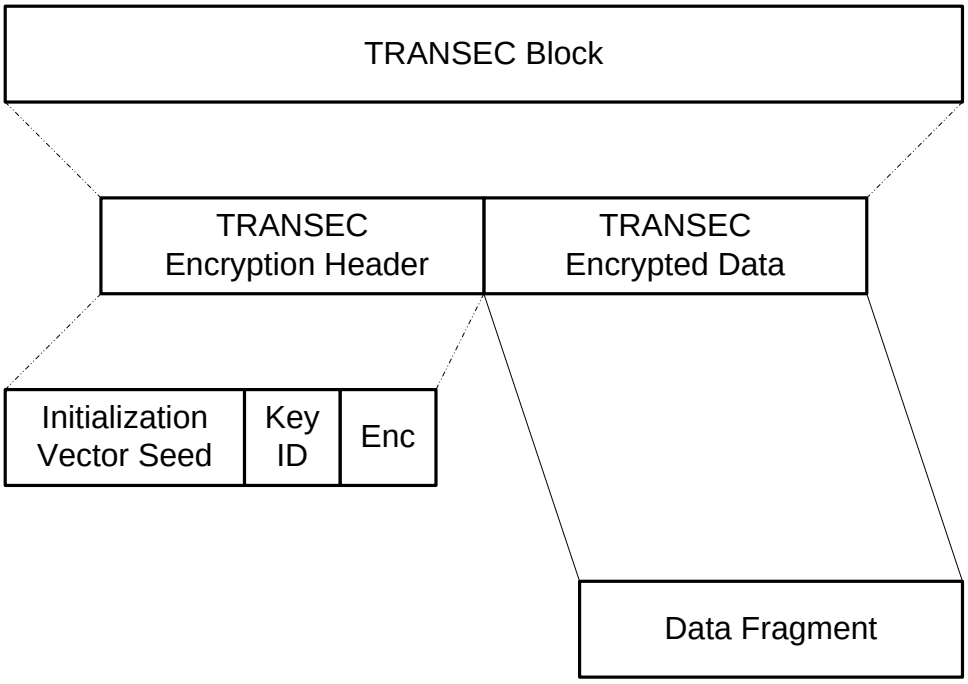


Figure I.2 Upstream TRANSEC Encryption

INDEX	NAME	SIZE	USE
1	Init Vector Seed	29 bits	generate an Initialization Vector
2	Key ID	2 bits	which Key Ring entry is in use
3	Enc	1 bit	whether or not block is encrypted

Table I.2 TRANSEC Upstream Encryption Header Fields

1. Init Vector Seed. This field starts at 0, when the Remote first comes on-line and has begun using TRANSEC (see section I.6, below). It is incremented with each TRANSEC Block.

2. Key ID. The values of this field, 0 through 3, are used to select an entry from the Key Ring as shown in Tables I.3 through I.6 (see section I.7, “Updating of TRANSEC Keys”, below).

3. Enc. This bit is set to 1 if the Block has been encrypted. If the Block is in clear text, this bit is set to 0.

In the event that a Remote does not currently have enough data to fill the current time and frequency slot (as assigned to it by the Hub's Timeplan) it must create an artificial and meaningless TRANSEC Block, and transmit that, repeating this step as necessary until it once again does have true data to broadcast. This action is required to prevent hostile eavesdroppers from deducing meta-information about the transmissions, based on a pattern of transmission peaks and lulls.

The contents of an artificial TRANSEC block must be random and different for each such artificial block. This action is required to prevent hostile eavesdroppers from detecting such artificial blocks by noticing a repeating pattern (which a fixed artificial block would generate). The contents, while random and meaningless, are nonetheless run through the encryption algorithm.

The Header for an artificial TRANSEC block must be valid. That entails: selecting a valid Key Ring entry to use for the encryption – and recording that in the Key ID field; incrementing the Init Vector Seed value; and setting the Enc field to 1.

Once the Block has been encrypted, processing continues with Bit Scrambling, as described in Annex H, section H.4.9, "Bit Scrambling".

I.5 (NU) Upstream TRANSEC Decryption

The Hub will use the information in the TRANSEC Block Header to properly initialize the AES algorithm for decrypting this Block.

The actual decryption of each TRANSEC Block adheres to the strictures of the X.509 standard.

If the Enc bit is set to 0, then no decryption is required and this step is skipped. A value of 1 for this bit means that the Block was encrypted, and thus decryption is necessary.

The Key ID is used to determine which Key Ring entry was used to encode this block. See Tables I.3 through I.6 (in section I.7, below) for the interpretation of the 0 through 3 values of these two bits.

The Init Vector Seed field's 29 bit contents are expanded to a full Initialization Vector field, 128 bits in length, by encrypting the Seed with the current AES key for the InRoute assigned by the Hub for the Data Slot that this TRANSEC Block was transmitted into by the Remote.

Once the Block has been decrypted, processing continues with the TDMA Header Interpretation, as described in Annex H, section H.5.6, "TDMA Header Interpretation".

I.6 (NU) Initial TRANSEC Exchange

This step is required to complete the acquisition of a newly online Remote by the Hub. After all other acquisition steps have been taken (see Annex E, Link Establishment), and the new Remote is no longer using an Acquisition slot for its responses to the Hub, it must remain in clear data transfer mode until the Initial TRANSEC Exchange has been completed.

Figure I.3 depicts this exchange.

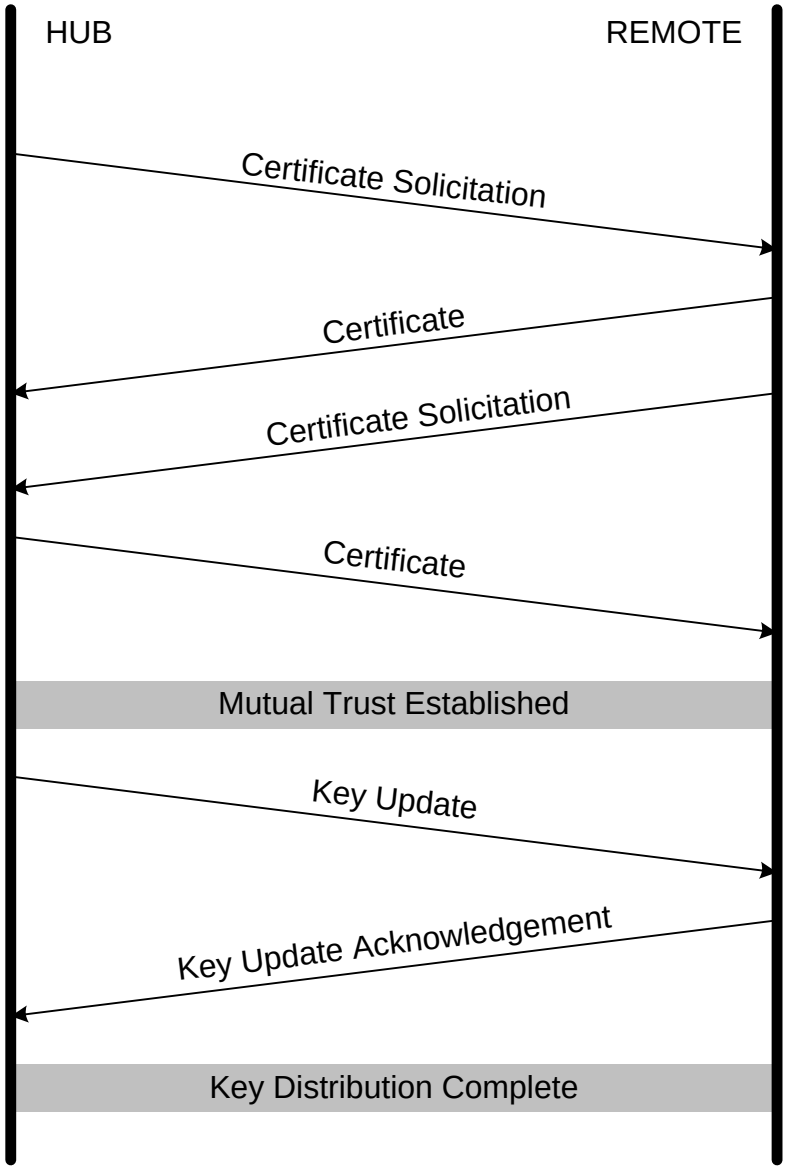


Figure I.3 Certificate Exchange

The Hub's next step is to send a Certificate Solicitation message to the new Remote. The Remote responds with its Certificate, and then sends its own Certificate Solicitation message to the Hub.

The Hub must be able to validate and establish a chain of trust, upon the receipt of a certificate from a new Remote, based on the contents of that certificate. X.509 certificates

and validation methodologies are to be used for this step. If it does validate the remote, then it will respond to the Remote's Certificate Solicitation message by sending its own Certificate. If however it does not validate the Remote, then the Remote will not be allowed to enter the network.

The Remote, upon receiving the Hub's Certificate, must likewise validate that Certificate (this step prevents a Remote from being "hijacked" by a hostile Hub).

All transmissions of Certificate Solicitations and Certificates must be performed in clear.

Once both peers – the Hub and the new Remote – are confident about each other, the Hub needs to provide the Remote with the TRANSEC Key for their link through a Key Update message. This Key must also be sent as clear text, so far as TRANSEC encryption is concerned. However, the exchange of X.509 certificates allows the Hub to encrypt the TRANSEC Key according to the Remote's public key information. The Remote receives the TRANSEC Key, stores it for use, and returns a Key Update Ack message.

At this point the Remote enters the network as a full and trusted member, and further interactions with the Hub are carried out in the TRANSEC encrypted partition of the timeplan (from the Hub), and in TRANSEC encrypted data blocks in normal upstream slots (from the Remote).

I.6.1 (NU) Certificate Solicitation Format

The format for a Certificate Solicitation message is shown in Table I.3.

IX	NAME	SIZE bytes	XDR	REPEAT	USE
1	Modem ID	4	int	once	type of machine code is running on
2	Network ID	4	int	once	which network
3	Rx Session State	4	int	once	status of peer node's sending
4	Rx Session ID	4	uns long	once	node's ID for incoming exchanges
5	Peer Tx Session ID	4	uns long	once	copied from latest peer exchange
6	Tx Session State	4	int	once	status of this node's sending
7	Tx Session ID	4	uns long	once	node's ID for outgoing exchanges
8	Peer Rx Session ID	4	uns long	once	copied from latest peer exchange
9	Rx Certificate State	4	int	once	status of peer's certificate
10	Rx Certificate ID	4	uns long	once	abbreviation of peer's certificate
11	Tx Certificate ID	4	uns long	once	abbreviation of node's certificate
12	Algorithms Length	4	uns long	once	number of Algorithms following
13	Algorithms	4*N	uns long	N	recognized encryption algorithms
14	Transport set count	4	int	once	number of Transport sets following
15	Transport ID sets	2	IIOB	twice	start and stop values for set
16	Message Length	4	uns long	once	length of Message following
17	Message	N	byte	N	optional text string

Table I.3 TRANSEC Certificate Solicitation

1. Modem ID: A combination of a numeric representation of the Modem Model and a Serial Number. Modem Models are provided as a vendor-specific number, which must be 14 bits or less in size (between 0 and 16,383). This number is left-shifted 18 places and ORed with the 18 least significant bits of the Serial Number of the particular modem at this node. Note that there may be some overlap of Modem Model numbers between different vendors.

2. Network ID: assigned during the installation of the Hub; provided to each Remote as part of its installation.

3. Rx Session State: There are three possible states that a session can be in:
    0 ::= Session Up
    1 ::= Session Down
    2 ::= Session Opening

4. Rx Session ID: The initial value is set to a random number when the node comes up. Thereafter the value is incremented once every constant time period (e.g., every fifteen minutes). The time period is configured during system installation.

5. Peer Tx Session ID: If there have been no exchanges from the peer, then this field has a value of zero. Otherwise the received Certificate Solicitation from the peer will contain that node's TX Session ID, and it will be copied into this field.

6. Tx Session State: There are three possible states that a session can be in:
    0 ::= Session Up
    1 ::= Session Down
    2 ::= Session Opening

7. Tx Session ID: The initial value is set to a random number when the node comes up. Thereafter the value is incremented once for each Key Update. Note that if the link goes down and the Remote needs to be re-acquired, there will be a Key Update as part of that Link Establishment activity (see Annex E), and thus this value will also be incremented.

8. Peer Rx Session ID: If there have been no exchanges from the peer, then this field has a value of zero. Otherwise the received Certificate Solicitation from the peer will contain that node's RX Session ID, and it will be copied into this field.

9. Rx Certificate State: There are two possible states that a Certificate can be in:
    0 ::= Have Not Received Certificate
    1 ::= Have Received Certificate

10. Rx Certificate ID: If there have been no exchanges from the peer, then this field has a value of zero. Otherwise the peer's certificate is run through ASN1_digest() – an Open SSL Library standard hash routine – to produce a twenty byte long abbreviation. The first four bytes of this are then placed into an unsigned long, with byte 0 being placed in the least significant eight bits, through byte 3 being placed in the most significant eight bits.

11. Tx Certificate ID: The node's certificate is run through ASN1_digest() – an Open SSL Library standard hash routine – to produce a twenty byte long abbreviation. The first four bytes of this are then placed into an unsigned long, with byte 0 being placed in the least significant eight bits, through byte 3 being placed in the most significant eight bits.

12. Algorithms Length: This is a count of how many Algorithms (see 13., below) will be listed. For TRANSEC, only the AES 256 CBC algorithm is allowed, so the value will always be "1".

13. Algorithms: This is a list of the encryption algorithms which this node is capable of using. The list places each algorithm's identifying number (see Table I.4) into a separate unsigned long field. For TRANSEC, only the AES 256 CBC algorithm is allowed, so "N" will always be "1", and the single algorithm list entry will be "507".

ENCRYPTION ALGORITHM	ID NUMBER
RSA	500
3-DES CBC	500
AES 256 CBC	507
AES 256 CFB	508
AES 256 CTR	509

Table I.4 TRANSEC Encodings for Encryption Algorithms

14. Transport set count: This is a count of how many Transport Sets (see 15., below) will be listed. For TRANSEC, only one Transport Set is allowed, so the value will always be "1".

15. Transport ID sets: Each set is a pair of numbers, each encoded as Integer In One Byte (IIOB), containing the first number in a range and then the last number in that range. There can be multiple ranges included. However, under TRANSEC, the only Transport IDs actually available are listed in Table I.5. Furthermore, of these, only the TRANSEC Master TX Netkey and the TRANSEC Master RX Netkey are usable for TRANSEC certificates. Therefore, for TRANSEC, the single range will have a start value of "2" and a stop value of "3".

TRANSPORT ID	ID NUMBER
Packet Encryption Unicast	0
TRANSEC Master TX Netkey	2
TRANSEC Master RX Netkey	3

Table I.5 TRANSEC Transport IDs

16. Message Length: This is a count of how many bytes are in the Message field (see 17., below). As the appending of a text message is an optional action, this value may be zero.

17. Message: An optional text message whose purpose might be for something along the lines of human-level status notification, condition logging, etc. How this message is created, what is done with it on the receiving side, and even whether it is included, are all outside the scope of this document.

I.6.2 (NU) Certificate Format

The Certificate is formatted in compliance with the X.509 Standard.

I.6.3 (NU) Key Update Format

The format for a Key Update message is shown in Table I.6. Note that Key Update messages are only sent from the Hub to the Remote.

IX	NAME	SIZE bytes	XDR	REPEAT	USE
1	Message Type	4	int	once	identifies Key Update message
2	Payload Length	4	uns long	once	length of Payload following
3	Payload	N	byte	N	encrypted new key
4	Algorithm Length	4	uns int	once	length of Algorithm following
5	Algorithm	N^	byte	N^	name of algorithm
6	Signature Length	4	uns int	once	length of Signature following
7	Signature	N	byte	N	validation of new key
8	Network ID	4	uns long	once	which network

Table I.6 TRANSEC Key Update

1. Message Type: Identifies this message as a Key Update. Always set to “1012”

2. Payload Length: This is a count of how many bytes are in the Payload field (see 3., below).

3. Payload: This field contains the encrypted contents of the Key Update Command. They have been encrypted first with the Remote’s public RSA key, and then secondly with the Host’s private RSA key.

Inside the encryptions, the Key Update Command has the format shown in Table I.7.

NAME	**SIZE bytes**	**XDR**	**REPEAT**	**USE**
Tx Session ID	4	uns long	once	node’s ID for outgoing exchanges
Rx Session ID	4	uns long	once	node’s ID for incoming exchanges
Command Count	4	uns long	once	number of “commands” following
Commands	varies		CmdCnt	Command Count of these

Table I.7 TRANSEC Key Update Command

Tx Session ID: The initial value is set to a random number when the node comes up. Thereafter the value is incremented once for each Key Update. Note that if the link goes down and the Remote needs to be re-acquired, there will be a Key Update as part of that Link Establishment activity (see Annex E), and thus this value will also be incremented.

Rx Session ID: The initial value is set to a random number when the node comes up. Thereafter the value is incremented once every constant time period (e.g., every fifteen minutes). The time period is configured during system installation.

Command Count: The number of commands contained in the rest of this block

Command: Each command has the format shown in Table I.8.

NAME	SIZE bytes	XDR	REPEAT	USE
Transport ID	1	int	once	purpose of this command
Active Key Index	1	UIIOB	once	currently active key ring index
New Key Count	4	uns long	once	number of new keys following
New Key	varies		NK Cnt	New Key Count of these

Table I.8 TRANSEC Command Format

Transport ID: Encoded as Integer In One Byte (IIOB). Under TRANSEC, the only Transport IDs actually available are listed in Table I.5. Furthermore, of these, only the TRANSEC Master TX Netkey and the TRANSEC Master RX Netkey are usable for TRANSEC certificates. Therefore, for TRANSEC, the Transport ID value of "2" or "3".

Active Key Index: In the Key Ring, which entry is currently active. Has a valid value of 0 through 3.

New Key Count: The number of new keys included in this command.

New Key: Each New Key has the format shown in Table I.9.

NAME	SIZE bytes	XDR	REPEAT	USE
Key Ring Index	1	UIIOB	once	slot in Key Ring for this new key
Valid	1	BIOB	once	is this a valid new key
Key Length	4	uns int	once	length of actual key
Key	N	bytes	N	the new key

Table I.9 TRANSEC New Key Format

Key Ring Index: The Remote's Key Ring has four slots, 0 through 3. This field tells the Remote which slot to store this new key into. Stored as an Unsigned Int In One Byte (UIIOB).

Valid: This should always be TRUE. Stored as a Boolean In One Byte (BIOB).

Key Length: The length in bytes of the actual key.

Key: The actual new key.

4. Algorithm Length: This is a count of how many bytes are in the Algorithm field (see 5., below).

5. Algorithm: The name of the encryption algorithm. Valid names are listed in Table I.10. For TRANSEC, the only valid Algorithm Name which is in use is “ALG_AES_CBC”.

^ Note that this field is buffered out to an exact four byte boundary, by appending meaningless bytes until the next boundary is reached. The value in field 4., “Algorithm Length” depicts the exact length of the contents of this field. To calculate the number of buffer bytes (if any) following the contents, use the formula:

Buffer byte count = (4 – (Algorithm Length MOD 4)) MOD 4

ALGORITHM NAMES
“ALG_3DES_CBC”
“ALG_AES_CBC”
“ALG_AES_CFB”
“ALG_AES_CTR”

Table I.10 TRANSEC Algorithm Names

6. Signature Length: This is a count of how many bytes are in the Signature field (see 7., below).

7. Signature: The first twenty bytes of the new key are taken and encrypted through first the destination Remote’s Public RSA key, and then through the Hub’s Private RSA key.

8. Network ID: assigned during the installation of the Hub; provided to each Remote as part of its installation.

I.6.4 (NU) Key Update Acknowledgement

Note that this message type is only sent from the Remote to the Hub.

This message has the exact same format as the Certificate Solicitation message. The Acknowledgement is explicitly contained in the Transport ID Sets field (field 15), by having the Transport ID Set contain both of the valid values for TRANSEC: “2” and “3”.

If this message is being used as a Certificate Solicitation message, then either the Peer Tx Session ID (field 5), or the Peer Rx Session ID, or both, will be zero: indicating that a Certificate has not been received and successfully processed. If both of these fields do have values, and if they match what the Hub expects to see from this Remote, then this message is a Key Update Acknowledgement.

I.7 (NU) Updating of TRANSEC Keys

For security reasons, the TRANSEC key held in common between the Hub and an individual Remote should be changed out periodically. The period selected, and the means of defining it at the Hub, are outside the scope of this document. TRANSEC key exchanges must be conducted over TRANSEC clear text (not encrypted with the previous key). The Hub and each Remote will keep multiple TRANSEC keys defined, thus allowing a mechanism for switching to a "next" key, without having to immediately precede a key "roll" with a key transfer.

The Key Ring is a table with four rows, each with three entries. Table I.11 defines the contents of each single row of the Key Ring. Tables I.12, I.13 and I.14 show how the contents of the table are modified to support a Key update.

Each of the four rows of the Key Ring has a symmetric key – each of which is different. At the time that the Table I.12 version of the Key Ring is in use, the TRANSEC Header Key ID fields will all have a value of "10", indicating that the symmetric key stored at row INDEX=2 is the one that was used for encryption, and is to be used for decryption. The symmetric key stored at row INDEX=3 will be the next one used, after the current one is finished. The symmetric key stored at row INDEX=0 is a far future key. And the symmetric key stored at row INDEX=1 is the most recently exhausted – no longer in use – key.

When it is time to update the key, a new key is generated by the Hub and stored in its Key Ring table, overwriting the Fallow-2 entry (which held the most recently exhausted key). This is shown at INDEX= 2 in Table I.13.

Then, the Hub transmits this new symmetric key to the Remote. This transfer is not TRANSEC encrypted, but is RSA encrypted: using that Remote's public key and with a signature which is created by using the Hub's private key.

Next, but not necessarily immediately thereafter, the Key Ring Value for each entry is incremented (with rollover from 11 to 00), resulting in the Key Ring depicted in Table I.14. This is the actual rollover into using a new (the next) key. There is no explicit notification, from the Hub to the Remote, that the rollover has been initiated; rather, the Hub now uses the "new" current Key ID ("11" now that the Table I.14 version of the Key Ring is in use) to pick

the symmetric key for use in the encryption, and includes that value in the TRANSEC Block Header. The Remote reads this value from the TRANSEC Block Header, uses it to index into the Key Ring and retrieve the proper symmetric key for use in decrypting the Block, and begins using that index to select the symmetric key for use when encrypting its own upstream TRANSEC Blocks.

INDEX	NAME	USE
1	Activity	Current, Next, Fallow-1, or Fallow-2
2	Active Key Value	00, 01, 10, or 11
3	Key	Symmetric Key: generated and stored

Table I.11 TRANSEC Key Ring Entries

INDEX	Activity	Value	Key
1	Fallow-1	00	future symmetric key
2	Fallow-2	01	symmetric key no longer in use
3	Current	10	symmetric key currently in use
4	Next	11	symmetric key to shift to next

Table I.12 TRANSEC Key Ring, pre-Rolling

INDEX	Activity	Value	Key
1	Fallow-1	00	future symmetric key
2	Fallow-2	01	newly generated key
3	Current	10	symmetric key currently in use
4	Next	11	symmetric key to shift to next

Table I.13 TRANSEC Key Ring, new Key inserted

INDEX	Activity	Value	Key
1	Fallow-1	01	newly generated key
2	Fallow-2	10	symmetric key rolled away from
3	Current	11	symmetric key rolled into (now current)
4	Next	00	future symmetric key (now next)

Table I.14 TRANSEC Key Ring, post-Rolling

ANNEX J
QoS PRINCIPLES

CONTENTS

J.1 (NU) Overview

Quality of Service (QoS) is outside the scope of this document, in part because most implementation is done independently on the Hub and Remote sides, without coordination (and thus also without any need for standardization). However, there are several aspects of SATCOM Engineering Order Wire for Control and Command which, while themselves inside the scope of this document, and thus described in other Annexes, do touch upon QoS as something that they can interact with. Accordingly, it is appropriate to discuss some Principles of QoS, describe in a broad manner different ways that QoS might be implemented, and touch upon potential ramifications of such implementations.

J.1.1 (NU) Introduction

Quality of Service is defined as the way that IP traffic is classified and prioritized as it flows through the components of a communications network. At its most basic, QoS is concerned with getting data from senders to receivers, in the most timely manner possible, as looked at from the viewpoint of all traffic in the network. QoS can, however, also be applied to maximizing the timeliness of transmissions for individual links (sender-receiver pairs), where those transmissions have been defined as having a very high priority.

When discussing QoS, at least four interrelated measures should be considered. These are Throughput, Latency, Jitter, and Packet Loss.

Throughput is a measure of capacity and indicates the amount of user data that is received by the end user application.

Latency is a measure of the amount of time between events. Unqualified latency is the amount of time between the transmission of a packet from its source and the receipt of that packet at the destination. If explicitly qualified, latency may also mean the amount of time between a request for a network resource and the time when that resource is received. In general, latency accounts for the total delay between events and it includes transit time, queuing, and processing delays.

Jitter is a measure of the variation of latency on a packet-by-packet basis. As such, this measure is only concerned with QoS for individual links.

Packet Loss is a measure of the number of packets that are transmitted by a source, but not received by their destination. The most common cause of packet loss on a network is network congestion. Congestion occurs whenever the volume of traffic exceeds the available bandwidth. In these cases, packets are filling queues internal to a network device at a rate faster than those packets can be transmitted from the device. When this condition exists, the most common response is for network devices drop packets, both to keep their own internal queues from being overwhelmed, and also to keep the network in a stable condition.

J.1.2 (NU) Bandwidth in Support of QoS

Normally, bandwidth is not a QoS concern in the downstream direction: data is transmitted in a determined order, and there either is, or isn't, enough bandwidth to transmit data faster than the Hub's internal queues are filling up. No bandwidth-related QoS can be applied in this direction.

In the upstream direction, however, the amount of bandwidth allocated to an individual Remote is directly related to that Remote's ability to provide a good QoS. Of all the QoS Principles discussed in this Annex, this is the only one which requires coordination between the Hub and Remote to implement. As described in detail in Annex F, "Link Maintenance", each Remote is constantly informing the Hub of that Remote's need, or at least desire, for bandwidth. The Hub must balance all these requests when determining how much bandwidth to allocate to each Remote.

J.1.3 (NU) QoS Queues

In either direction, upstream or downstream, the data transfer limit is compelled by the simple fact that there can be only one next data block to be transmitted. QoS, then, revolves around the selection of which data block, out of all those waiting, should be next.

One approach to this question is to define a number of Queues, along with a methodology for prioritizing which queues are more important than which other queues; and also whether more important queues must be emptied prior to less important queues being dealt with, or if several queues must be processed in some form of collaborative schema. An assumption here is that all data blocks on the same queue have equal importance, and thus treating each queue as a FIFO is appropriate.

This approach allows the logic – for selecting the next data block to transmit – to operate very quickly, as most of the decision process was instigated when selecting the proper queue onto which to place each data block.

J.1.4 (NU) Assignment of Data Blocks to Queues

Whether the multi-queue approach described above is used or not, QoS can only be implemented if there is some means of differentiating between separate data blocks, and determining thereby that one data block is more important than another. This hierarchy of importance among data blocks is commonly referred to as “Service Levels”. There are a number of criteria which can be rapidly examined and used for assigning a service level to a data block; and by extension for then selecting a queue onto which to place this data block.

J.1.5 (NU) Tailoring Data Blocks for QoS

All QoS methodologies work better if the resources needed for the transmission of a data block are roughly the same for all data blocks. For example, a data block which is two thousand bytes long will use up more resources than one which is twenty bytes long. Trying to balance resources when there is such a discrepancy in resource needs becomes overly complex.

One way to alleviate this concern is to break all data blocks down into pieces which are roughly the same size, and then reassemble them on the other side of the transmission. The optional Data Segmentation step, described in Annex H, provides exactly that service, for specifically this purpose. It is therefore recommended that Data Segmentation be enabled, so as to enhance the ability of QoS methods.

J.2 (NU) Bandwidth in Support of QoS

In the downstream direction, there is no modification possible of the available bandwidth, thus there is no QoS aspect in this direction.

In the upstream direction, the amount of bandwidth allocated to each Remote is determined by the timeplan generated by the Hub. Annex F, “Link Maintenance”, in section F.6, “Bandwidth Request from Remote”, details the format through which each Remote continuously informs the Hub of that Remote’s queued-up data blocks; and by implication how much bandwidth that Remote would like to be allocated.

The Remote’s most basic information, that it sends to the Hub in this Demand Header, is how many time slots worth of data blocks it has waiting in its queues, for sending to the Hub. Under the assumption that realtime data (e.g., VoIP) is more time critical – and thus requires a higher priority than “ordinary” data – the Remote has a field in the Demand Header to tell the Hub what percentage of the total requested time slots are for the transmitting of realtime data blocks. Finally, the Demand Header has a single bit, Priority, which can be set to tell the Hub that the Remote considers all of its queued-up data to be important enough to justify expedited handling.

The Hub will take into consideration all of this information, as gathered from every Remote, when constructing the next timeplan it sends out. Note that the algorithms used by the Hub to make decisions on constructing the timeplan – thus on allocating the upstream bandwidth across all the Remotes – is outside the scope of both the main document and also this general discussion Annex.

Likewise, the methodology by which a Remote determines what to put into its Demand Headers is outside the scope of this Annex. However, some general rules of thumb may be stated.

The Demand field, stating how many remaining time slots worth of data are queued up in the Remote, is a straightforward datum, and independent of any QoS strategy.

The Realtime field, giving the percentage of the Demand field which contains time-sensitive data blocks, requires that the Remote keep track of those time-sensitive data blocks in some manner that partitions them from “ordinary” data blocks. The multiple queues methodology discussed below is one such possible manner. In addition to keeping an accurate count of both time-sensitive and ordinary data blocks – for the purpose of reporting those numbers to the Hub – it is also necessary, in support of QoS, for the Remote to transmit time-sensitive data blocks before ordinary data blocks. Again, the queuing mechanism discussed below is one approach for meeting that requirement.

Finally, the Remote needs to have some algorithm for deciding when it is appropriate to set the Priority bit. Through the use of a multiple queue methodology, the Remote could decide to set the bit under conditions such as: when a high priority queue has any data blocks; when a set of higher priority queues exceed a data block count threshold; when some ratio of higher priority data blocks (data blocks on higher priority queues) to lower priority data blocks is exceeded.

J.3 (NU) QoS Queues

Just about any means of partitioning and tracking different types of data blocks will support QoS. The queuing methodology discussed here is an attempt to be comprehensive, generic, and flexible. Pretty much any subset of this methodology will still support QoS: with less fineness of control, but also with less effort.

Theoretically, a single queue could supply all the support needed for not only data transmission but also for QoS support. Each data block, as it arrives from the outside world, would be compared with those data blocks already on the queue, and would then be slipped into place between whichever two existing data blocks are immediately higher and lower than it, priority-wise. In practice, this is cumbersome, time-consuming, and difficult to maintain.

Given an approach involving multiple queues, it is worthwhile to consider implementing multiple types of queues, with several queues defined under each type. More queues allows a finer tuning of the QoS, while the different queue types permits a more flexible approach to QoS.

There are three common types of queues which are useful in support of QoS: Priority Queues; Class-Based Weighted Fair Queues (CBWFQ); and Best Effort Queues. QoS support could be set up with any one of these three types. If more than one type is implemented, then it is highly recommended that the different types be assigned a hierarchal order. For example: all Priority Queues must be empty before any data blocks on CBFWQs are transmitted; and all Priority and CBFWQs must be empty before any data blocks on Best Effort Queues are sent. Naturally, if a new data block were added to a higher queue type (or, within a single queue type, added to a more important queue), then that data block would take precedence over all other waiting data blocks (but would not interrupt the transmission of the current data block).

Priority Queues are quite straightforward. There are some number of queues, each with criteria for how data blocks are assigned to that queue. Each priority queue is assigned a priority number. Usually “1” is assigned to the highest priority, “2” to the second highest, etc. Data blocks that are on the highest priority queue are transmitted first; data blocks on lower

priority queues are not transmitted until all higher priority queues are empty. At its most flexible, it is possible to assign the same priority to more than one queue; in this case data blocks are taken off of those "equal" queues in a round-robin manner, until all queues of that priority are empty.

Best Effort Queues are even more straightforward. Data blocks are taken off of all of the best effort queues in a round-robin manner. As such, the entire set of best effort queues could be considered as a special case of priority queues with equal priority. But since best effort queues have no priority assignable to them, they would effectively be either an implicit "lowest" priority (if best effort queues were placed after priority queues in the hierarchy) or else an implicit "highest" priority (if best effort queues were placed before priority queues in the hierarchy). The other advantage of defining best effort queues is that, since they are a separate type, it is possible to insert the CBFWQ type in between priority queues and best effort queues in the queue type hierarchy.

Class-Based Weighted Fair Queues are more complex. Each queue of this type is assigned a "queue cost" (also "service level cost" or "class cost"). Each data block on a queue is assigned a "transmission cost" which is computed by multiplying that queue's queue cost times the length in bytes of the data block. When it is time to select the next data block to transmit, from among all of those on CBWFQs, that data block with the lowest transmission cost is selected. [Note that this means new data blocks should be placed on each queue such that the shortest data blocks are at the top of the queue. Note that in the upstream direction, every data block is sized to fit in one time slot, thus all data blocks are the same size, and each CBWFQ can be treated is a simple FIFO. Note that in the downstream direction, if the optional Data Segmentation step has been applied, then all data blocks will be of the same size, and each CBWFQ can be treated as a simple FIFO.]

This accounts for the "Class-Based Weighted" aspect of these queues. The "Fair" aspect works as follows. Whenever a data block is selected for transmission, from one of the CBWFQs, all other (non-empty) CBWFQs' queue cost is temporarily decreased by a constant amount. This guarantees that no matter how large an initial weight - queue cost - was assigned to one CBWFQ, eventually its being constantly "skipped over" will result in sufficient cumulative decreases that its first data block now is the data block with the lowest transmission cost. Every time a CBWFQ is selected to have its next block transmitted, that CBWFQ's queue cost is returned to its original assigned value.

As a final note: because the QoS methodology is completely independent of the data processing on the reception side of the link, there is no need to coordinate – at runtime, link establishment time, or remote installation time – the QoS methodology used at a Remote and that used at the Hub, or at any other Remote.

J.4 (NU) Assignment of Data Blocks to Queues

After the queue types have been selected, and the individual queues within each type (for Priority and CBWFQ) have been defined, it is necessary to have a means of appropriately placing each incoming packet onto the appropriate queue. The exact set of "rules" which will be used will depend on the environment and desired results, and are not a matter for even the most general speculation in this Annex. However, the hard data, associated with each packet, upon which those rules will act, are most commonly some or all of these IP packet fields:

- Source IP Address(es)
- Destination IP Address(es)
- Source Port(s)
- Destination Port(s)
- Protocol(s)
- TOS priority
- TOS precedence
- VLAN ID

J.5 (NU) Tailoring Data Blocks for QoS

In the upstream direction, data blocks are sized to exactly fit one time slot. In the downstream direction, if the Data Segmentation option has been activated, all data blocks will be of the same size. However, regardless of which direction the data is flowing, the likelihood of a single IP packet being inside a single data block is vanishingly small. Accordingly, the information about which queue an IP packet was assigned to, as determined at the time of that IP packet's arrival at either the Hub or Remote, must stay attached to every data block which makes up that IP packet, so that those data blocks will, at the appropriate step in the processing chain, be placed onto the appropriate queue. Thus as each IP packet is, possibly, split up into multiple data segments, and then, possibly, split into multiple data fragments, it will be necessary for that IP packet's queue assignment to be attached in some way, and carried with the data segment, or data fragment, until the processing chain places each data block onto the appropriate queue.

[This page intentionally left blank.]