ETSI TS 102 821 V1.1.1 (2003-12)

Technical Specification

Digital Radio Mondiale (DRM); Distribution and Communications Protocol (DCP)

Reference
DTS/JTC-DRM-05

Keywords
broadcasting, digital, DRM, radio

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.org

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

Introduction

A large number of communications protocols have been developed to allow reliable exchange of data using a wide variety of different techniques. Some have relied on two-way communication to allow requests for re-tries of missing or corrupted messages, while others have relied on Forward Error Correcting codes such as Reed Solomon to rebuild the original message. Unfortunately most of the protocols are tightly coupled to the application they were originally developed for, do not scale well in multicast networks or are unsuitable for use over the uni-directional circuits often found in distribution systems. When the development of a distribution protocol for Digital Radio Mondiale broadcasts was considered, none of the available protocols was deemed suitable and so it was decided to develop a general purpose, low-level, reliable communications protocol suitable for both uni-directional and bi-directional data links which would meet the needs of DRM but would also hopefully be flexible enough to meet the needs of other applications as well.

1 Scope

The present document gives the specification for a general purpose distribution link suitable for multicasting data to many recipients using a uni-directional communications network.

2 References

The following documents contain provisions that, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents that are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] IETF RFC 2396: "Uniform Resource Identifiers (URI): Generic Syntax".

[2] IETF RFC 2718: "Guidelines for new URL Schemes".

3 Definitions, symbols, abbreviations and conventions

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

AF Packet: collection of TAG Items with a header carrying a cohesive and self-contained block of data

application: any DCP-based protocol (specified outside the scope of TS 102 281)

Application Framing (AF): layer of the DCP providing a logical grouping of a number of TAG Items

byte: collection of 8-bits

Distribution and Communication Protocol (DCP): transport layer communications protocol providing fragmentation, addressing and/or reliable data transmission over errored channels using a Reed Solomon code to provide Forward Error Correction

TAG Item: DCP elemental type combining in a single logical data the name, length and value of the data

TAG Name: name field within an individual TAG Item used to identify an individual piece of information

TAG Packet: collection of TAG Items with a header carrying a cohesive and self-contained block of data

TAG Value: payload of a TAG Item

3.2 Symbols

For the purposes of the present document, the following symbols apply:

N_x	The value *N* is expressed in radix *x*. The radix of *x* shall be decimal, thus $2A_{16}$ is the hexadecimal representation of the decimal number 42.
$\lceil x \rceil$	The smallest integral value numerically greater than *x*. Sometimes known as the "ceiling" function.
$\lfloor x \rfloor$	The largest integral value numerically less than *x*. Sometimes known as the "floor" function.
$\frac{x}{y}$	The result of dividing the value *x* by the value *y*.
$MIN\{a,...,z\}$	The smallest value in the list.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AF	Application Framing
CRC	Cyclic Redundancy Check
DRM	Digital Radio Mondiale
FEC	Forward Error Correction
IP	Internet Protocol
LSb	Least Significant bit
LSB	Least Significant Byte
MSb	Most Significant bit
MSB	Most Significant Byte
MTU	Maximum Transmit Unit
PFT	Protection, Fragmentation and Transportation
RS	Reed Solomon
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

3.4 Conventions

The order of bits and bytes within each description shall use the following notation unless otherwise stated:

- in figures, the bit or byte shown in the left hand position is considered to be first;
- in tables, the bit or byte shown in the left hand position is considered to be first;
- in byte fields, the Most Significant bit (MSb) is considered to be first and denoted by the higher number. For example, the MSb of a single byte is denoted "b_7" and the Least Significant bit (LSb) is denoted "b_0";
- in vectors (mathematical expressions), the bit with the lowest index is considered to be first.

The order of transmission (MSb-first or LSb-first) shall be the conventional order for the physical link in use. Where both orders are common, MSb-first shall be used.

4 General description

4.1 System overview

The Distribution and Communication Protocol is specifically designed to permit reliable multicast communication from a central server to a number of receivers. Errors on the communications link(s) can be detected and corrected using a Reed-Solomon Forward Error Correction code. As a result the communications links employed can be uni-directional, offering a potentially huge cost saving.

4.2 System architecture

The application data is carried from the server to the receiver through a number of layers as shown in figure 1. The data at each layer is encapsulated in a series of packets. The TAG layer encapsulates the elementary arbitrary length data items, while the AF Layer combines the elementary data into a cohesive block of related data. The optional PFT layer allows fragmentation of the potentially large AF packets, and adds the possibility of having addressing and FEC. The AF Packets or the PFT Fragments can then be transported by any one of a number of physical links, including (but not limited to) asynchronous serial, UDP/IP and even stored as a file on a disc. Some examples of the possible link layer options are shown in figure 2.

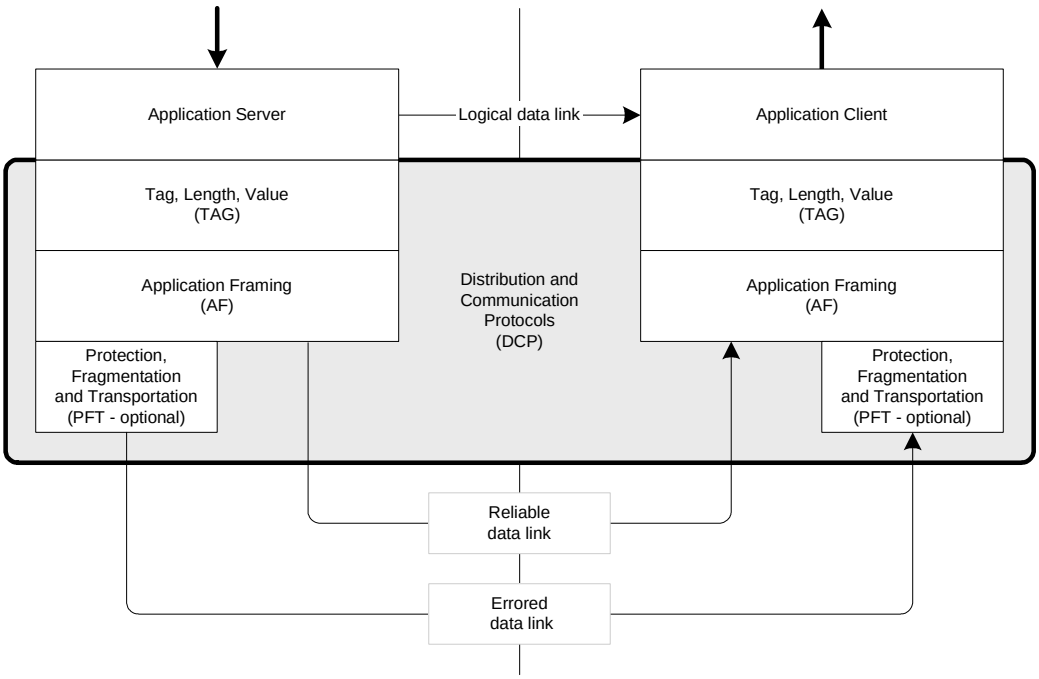


Figure 1: DCP protocol stack

Although shown used with errored data links, the PFT layer is also helpful when using reliable data links which do not provide a transport addressing function.

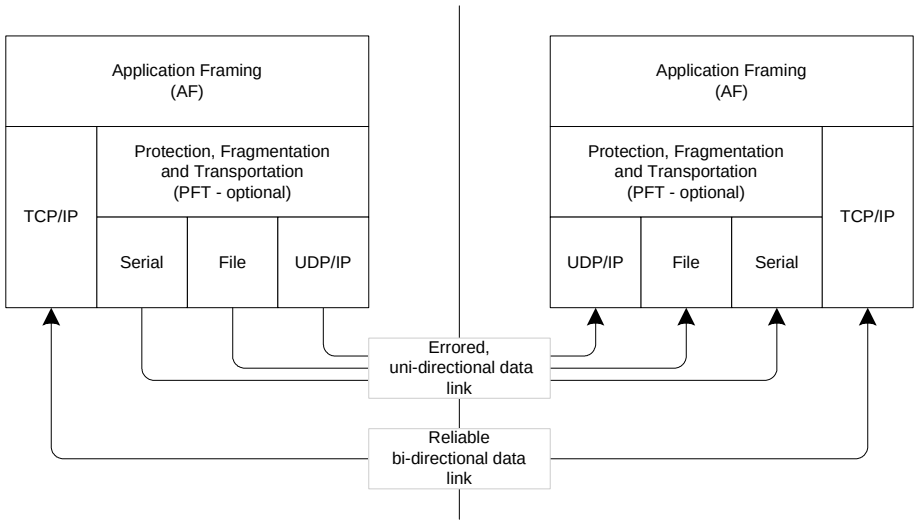


Figure 2: Example DCP link layers

4.2.1 TAG Items and AF packets and PFT fragments

A very brief overview of the structure of the data at the various layers is shown in figure 3.

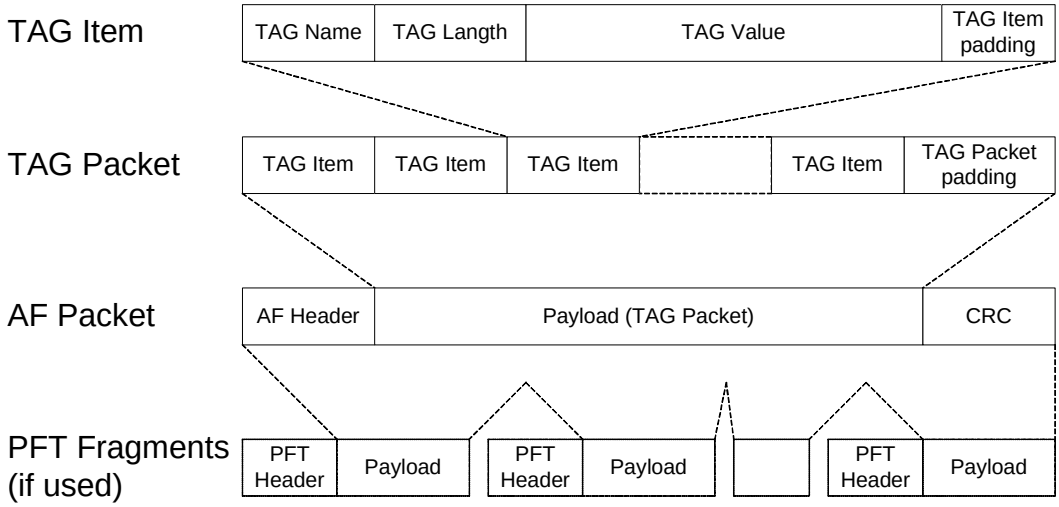


Figure 3: TAG Items, AF packets and PFT fragments

5 TAG Layer

The TAG Layer forms the interface between the application and the DCP. Within the TAG Layer, TAG Items encapsulate individual data items. TAG Items are combined to form in a TAG Packet to form a logically cohesive block of data to the application. Thus to define a new application it is merely necessary to define a series of TAG Items and to impose any necessary limits on the features of the DCP, for example specifying that the PFT Layer shall always be used or that the physical link shall always be Ethernet, etc.

5.1 TAG packet

A TAG Packet is the name given to a cohesive group of TAG Items that have some significance to the overall application. TAG Packets do not contain any synchronization or error correction and so will typically only exist within equipment.

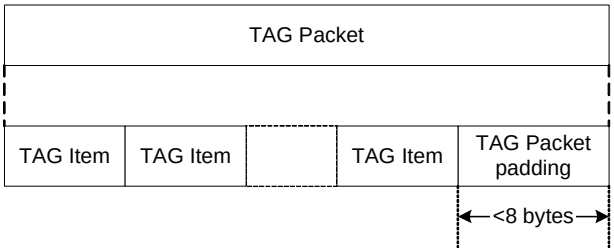


Figure 4: TAG packet

A TAG Packet may include up to 7 bytes of padding after the last TAG Item - the data carried in the padding shall be undefined. Such padding shall be ignored by all receivers. Since the shortest length of a TAG Item is 8 bytes, TAG Packet Padding can be easily identified. If more than 7 bytes of padding are required, the **dmy* special TAG Item shall be used, see clause 5.2.2.2.

It should be noted that the TAG Packet itself has no header, and there is no way of determining the total length of the TAG Items in the packet: these functions are achieved using the AF Layer described later. As a result, the TAG Packet is not a suitable structure for passing between pieces of equipment, but is a convenient abstraction that can be used in implementations if desired.

5.1.1 General rules

The application may determine whether the order of TAG Items within a TAG Packet has any significance. It is very strongly recommended that the order of TAG Items shall not be significant.

The application may determine whether a single TAG Packet may contain multiple TAG Items with the same name.

Implementations shall ignore any TAG Items included in a TAG Packet that are not recognized. This allows proprietary extensions to be made to existing protocols in a backwards compatible manner.

TAG Item Name may comprise any four bytes, and need not be restricted to ASCII characters. One general restriction is given in clause 5.2.2. Applications may define additional restrictions.

5.2 TAG item

The structure of a single TAG Item is as shown in figure 5.

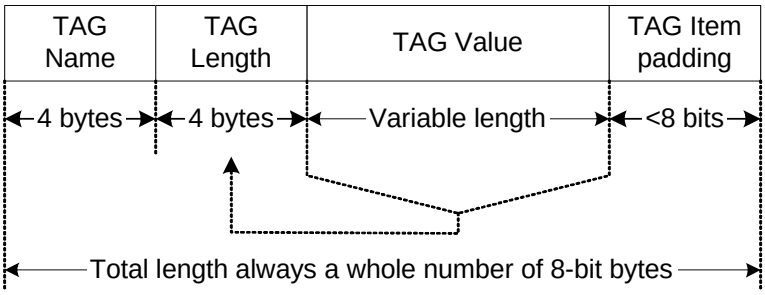


Figure 5: TAG Item

TAG Name: A four-byte name used to identify the data value carried in the TAG Item.

TAG Length: A four-byte value representing the number of bits in the TAG Value field.

TAG Value: Any value as required by the application.

TAG Item padding: Up to seven bits of undefined value as required to make the total length of the TAG Item a whole number of bytes.

5.2.1 Hierarchical TAG items - Coding example

If required by the application, a single TAG Item may encapsulate further TAG Items, as illustrated in figure 6. The depth of the hierarchy may be limited by the application if appropriate.

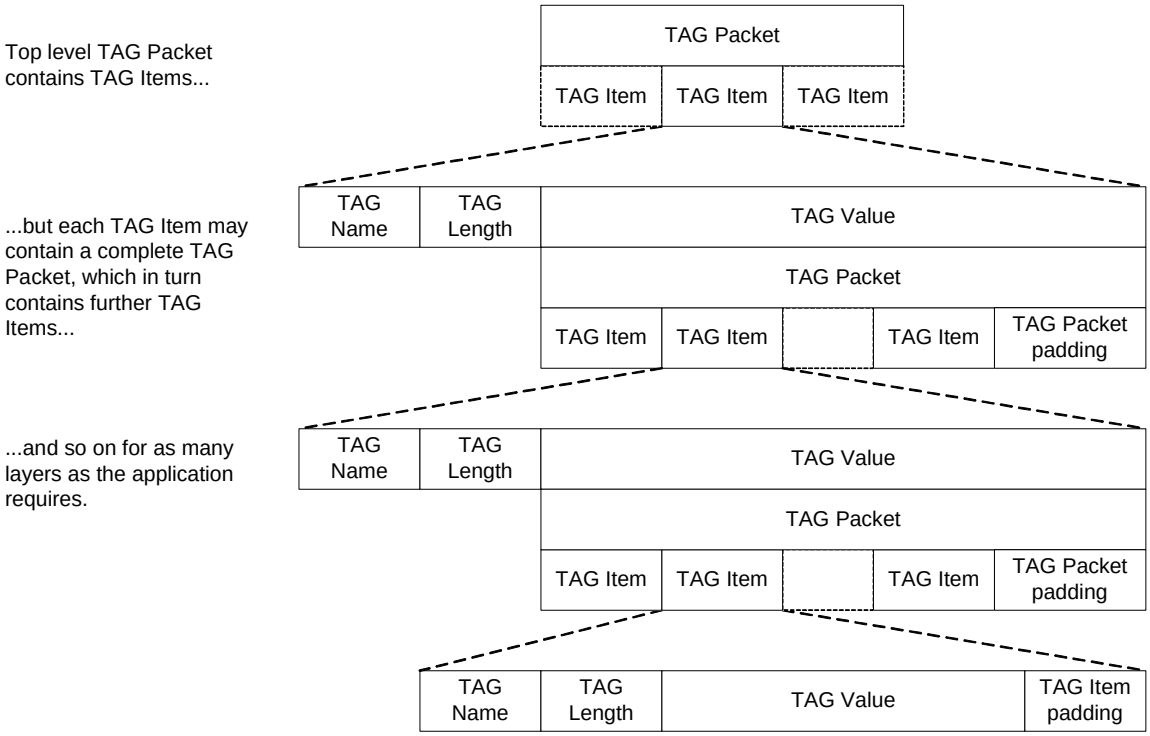


Figure 6: Hierarchical TAG Items

Since every lowest-layer TAG Item will contain TAG Item padding to ensure it is a whole number of 8-bit bytes long, the higher layer TAG Items shall never require TAG Item padding.

5.2.2 Special TAG items

Each application is largely free to define TAG Item names as appropriate, the only exception being that all names beginning with the ASCII "*" character, 42 (decimal) or $2A_{16}$, are reserved as control TAG Items.

5.2.2.1 Protocol type and revision, *ptr

It is highly recommended that every application using the DCP should declare a protocol type and revision in every TAG Packet using the *ptr TAG Item as shown in figure 8.

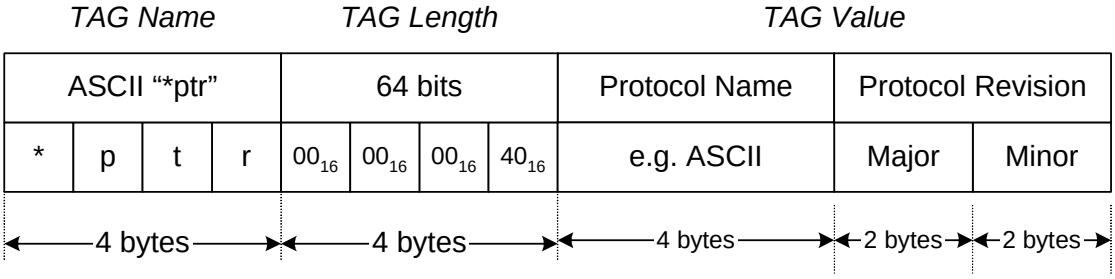


Figure 7: Protocol type and revision

Protocol type: the name of the protocol. Typically this will be encoded using ASCII values in the range 20_{00} to $7F_{16}$, but values outside this range may be used if desired.

Major revision: A binary counter representing the major version number of the protocol, starting from 0000_{16}.

Minor revision: A binary counter representing the minor version number of the protocol, starting from 0000_{16}.

This TAG Item requires no TAG Item padding.

5.2.2.1.1 Revision numbering

Each application is permitted to use any revision numbering scheme as required, however it is highly recommended that the minor revisions are backwards compatible, thus an application implementing version 4.3 of protocol WXYZ should be able to decode versions 4.0, 4.1 and 4.2 in addition to 4.3, but may not necessarily support versions 3.1 or 5.0. Additionally, version 4.5 packets shall be processed as if they were version 4.3 with any new features added in version 4.4 or 4.5 being ignored.

As a general principal, the addition of new TAG Items or the definition of previously reserved bits should be reflected by a minor version number change. Non-backwards compatible re-definition (including lengthening) of existing TAG Items should be reflected by a major version number change.

5.2.2.2 Dummy padding, *dmy

To allow word alignment to be achieved when necessary, the *dmy TAG Item allows more than 8 padding bytes (of undefined data) to be inserted into a TAG Packet: if less than 8 bytes of padding are required, the TAG Packet Padding shall be used instead.

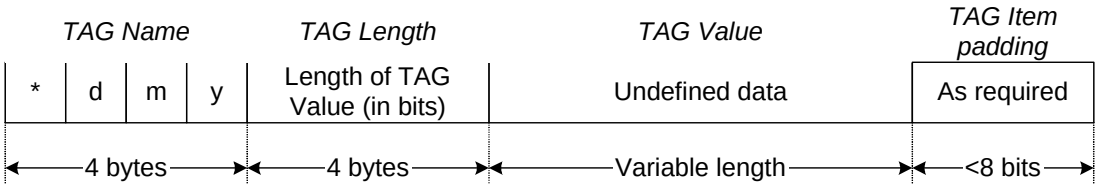


Figure 8: Dummy (padding) TAG Item

6 Application Framing (AF) layer

The AF layer encapsulates a single TAG Packet in a simple structure that is suitable for passing between equipment connected via error-free links. Such links may be provided by existing networks, such as TCP/IP, or by the PFT Layer described in clause 7.

6.1 AF packet structure

The basic structure of an AF Packet is as shown in figure 9.

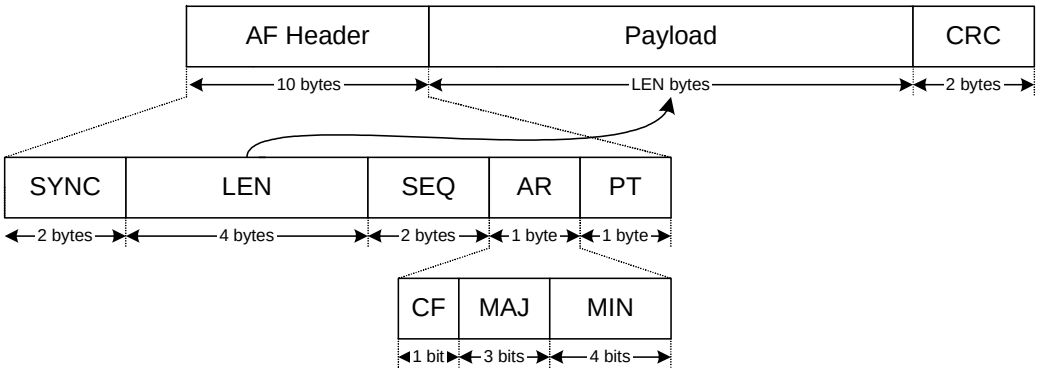


Figure 9: AF Layer

SYNC: two-byte ASCII representation of "AF".

LEN: length of the payload, in bytes.

SEQ: sequence number. Each AF Packet shall increment the sequence number by one for each packet sent, regardless of content. There shall be no requirement that the first packet received shall have a specific value. The counter shall wrap from $FFFF_{16}$ to 0000_{16}, thus the value shall count, $FFFE_{16}$, $FFFF_{16}$, 0000_{16}, 0001_{16}, etc.

AR: AF protocol Revision - a field combining the ***CF***, ***MAJ*** and ***MIN*** fields.

CF: CRC Flag, 0 if the ***CRC*** field is not used (CRC value shall be 0000_{16}) or 1 if the ***CRC*** field contains a valid CRC.

MAJ: major revision of the AF protocol in use, see clause 6.2.

MIN: minor revision of the AF protocol in use, see clause 6.2.

Protocol Type (PT): single byte encoding the protocol of the data carried in the payload. For TAG Packets, the value shall be the ASCII representation of "T".

CRC: CRC calculated as described in annex A if the ***CF*** field is 1, otherwise 0000_{16}.

6.2 Revision history

Table 1: Revision history

Major revision	Minor revision	Date	Changes
01_{16}	00_{16}	2003-01-28	Initial public release

7 PFT layer

The optional Protection, Fragmentation and Transport, or PFT Layer provides, as its name suggests, three separate functions. The first is error protection using a Reed-Solomon code that can detect and correct individual bit errors and also rebuild entire lost packets. The second is fragmentation, splitting large packets into smaller units suitable for data links that enforce lower MTU. Finally, the PFT layer allows a limited form of transport addressing so that those lower layers that do not include addressing (for example RS232 serial links) can be used with multiple transport streams. It is not compulsory that all three functions are used simultaneously, and the use of optional header fields minimizes the overhead when a specific feature is not required. The following combinations are permitted:

- Encapsulation.
- Simple Fragmentation.
- Reed Solomon FEC and Fragmentation.

In addition, each of the above three options can be combined with transport addressing if desired. These options are summarized in figure 10.

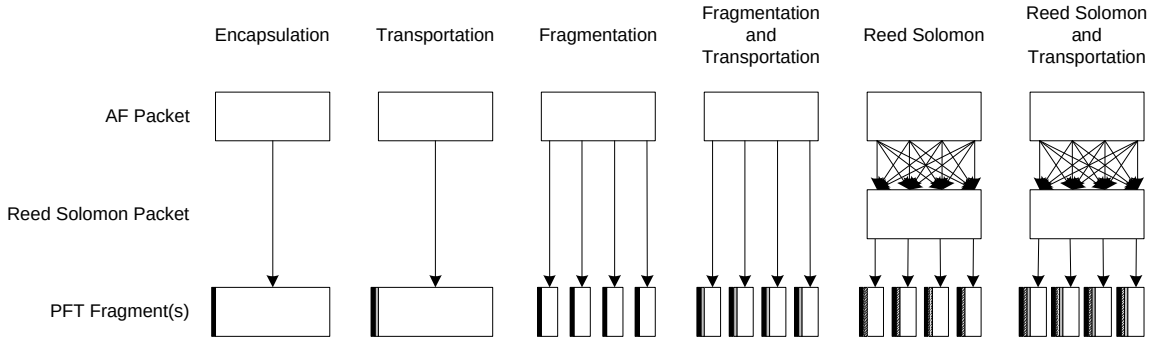


Figure 10: Options available when using PFT Layer

7.1 PFT fragment structure

The structure of a PFT Fragment is as shown in figure 11.

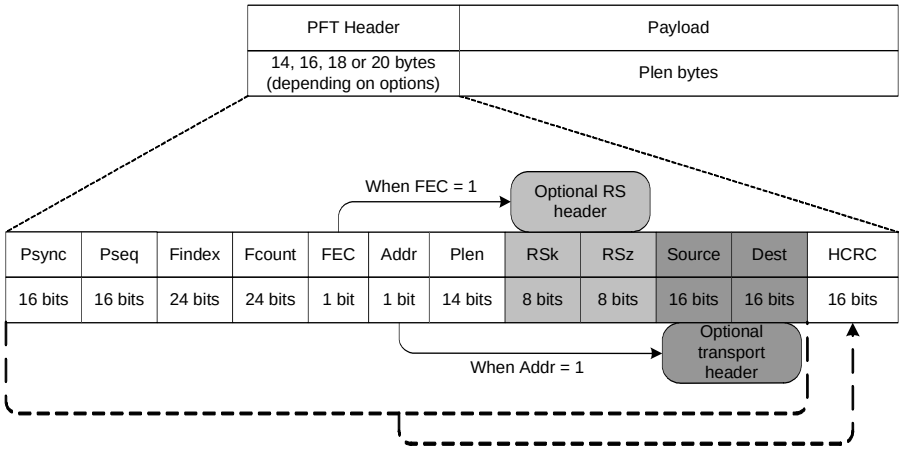


Figure 11: PFT Fragment

Psync: the ASCII string "PF" is used as the synchronization word for the PFT Layer.

Pseq: 16-bit counter incremented by one for each AF Packet. The value shall wrap around from 2^{16}-1 to 0, e.g. …, 2^{16}-2, 2^{16}-1, 0, 1, … The receiver shall not expect a specific value in the first fragment received. The value of the ***Pseq*** field has no link to the value of the AF Packet's ***SEQ*** field.

Findex: 24-bit counter incremented by one for each fragment that forms part of a single AF Packet. The first fragment of each AF Packet shall have the value zero. The value shall not wrap, thus imposing a maximum limit on the AF Packet size that can be carried. The maximum value varies depending on the MTU of the link, but is typically several gigabytes.

Fcount: number of fragments produced from this AF Packet, in the range $1…2^{24}$-1. The value zero shall not be used.

FEC: when this single-bit flag is set (1), the ***Optional RS Header*** is present.

Addr: when this single-bit flag is set (1), the ***Optional Transport Header*** is present.

Plen: the length, in bytes, of the payload of this fragment.

RSk: the length of the Reed Solomon data word - see clause 7.2.2. Only present when the ***FEC*** field is 1.

RSz: the number of padding bytes in the last Reed Solomon block - see clause 7.2.2. Only present when the ***FEC*** field is 1.

Source: free-format 16-bit source identifier. Only present when the ***Addr*** field is 1.

Dest: free-format 16-bit destination identifier. Only present when the ***Addr*** field is 1.

HCRC: PFT Header CRC calculated over the PFT Header fields from ***Psync*** including any optional headers present. The CRC shall be calculated as described in annex C.

When both ***FEC*** and ***Addr*** are set (i.e. when both optional headers are present), the two headers shall appear in the order indicated in figure 11.

7.2 Definitions

Throughout the following text, the following definitions shall apply.

7.2.1 Known values

- l is the total length of the original AF Packet, including the header and CRC.
- k_{max} is the maximum value of k and has the value 207.
- p is the number of bytes of Reed Solomon parity per chunk and has the value 48.
- m is the maximum number of fragment losses per packet that the Reed Solomon is to be able to recover. When recovery after fragment loss is not required, and when Reed Solomon is not used, m shall be zero. Values of m greater than 5 are not recommended due to the overhead of sending many small fragments.
- *MTU* is the Maximum Transmit Unit size (in bytes) for the underlying transport layer. When the transport layer has no MTU and when the MTU is greater than 2^{14}, then the value of *MTU* shall be 2^{14}.
- h is the PFT header length in bytes. The value shall be 12, 14, 16 or 18 bytes depending on the options in use.

7.2.2 Calculated values

- c is the number of Reed Solomon chunks (fixed as zero if Reed Solomon not in use).
- k is the data length of each chunk and is carried in the ***RSk*** field of the PFT Header (zero if Reed Solomon not used).
- z is the number of zero-bytes added to the last Reed Solomon chunk and is carried in the ***RSz*** field of the PFT Header (zero if Reed Solomon not in use).
- s_{max} is an intermediate result representing the maximum payload size, in bytes, for a single fragment.
- f is the number of fragments and is carried in the ***Fcount*** field of the PFT Header.

- *s* is the actual size of the fragment(s), in bytes.
- *L* is the length (in bytes) of the packet to be fragmented. When Reed Solomon is used, *L* has the value *f.s*, otherwise *l*.

$$c = \left\lceil \frac{l}{k_{\max}} \right\rceil$$

$$k = \left\lceil \frac{l}{c} \right\rceil$$

$$z = c.k - l$$

$$s_{\max} = MIN\left\{ \left\lfloor \frac{c.p}{m+1} \right\rfloor, MTU - h \right\} \qquad \text{for } m > 0$$

$$s_{\max} = MTU - h \qquad \text{for } m = 0$$

$$f = \left\lceil \frac{l + c.p + z}{s_{\max}} \right\rceil$$

$$s = \left\lceil \frac{l + c.p + z}{f} \right\rceil$$

7.3 Encoding

The following steps shall be completed in order to encode a single AF Packet. In the event that an option (e.g. Reed Solomon) is not enabled, that step is simply not performed.

7.3.1 Reed Solomon

The original packet is first broken down into *c* Reed Solomon Chunks of *k* bytes each - *z* zero-bytes of padding are added to the last chunk if necessary. The Reed Solomon parity bytes are then calculated and appended to each chunk, and the resulting RS Block is interleaved to form an RS Packet. This is shown diagrammatically in figure 12.

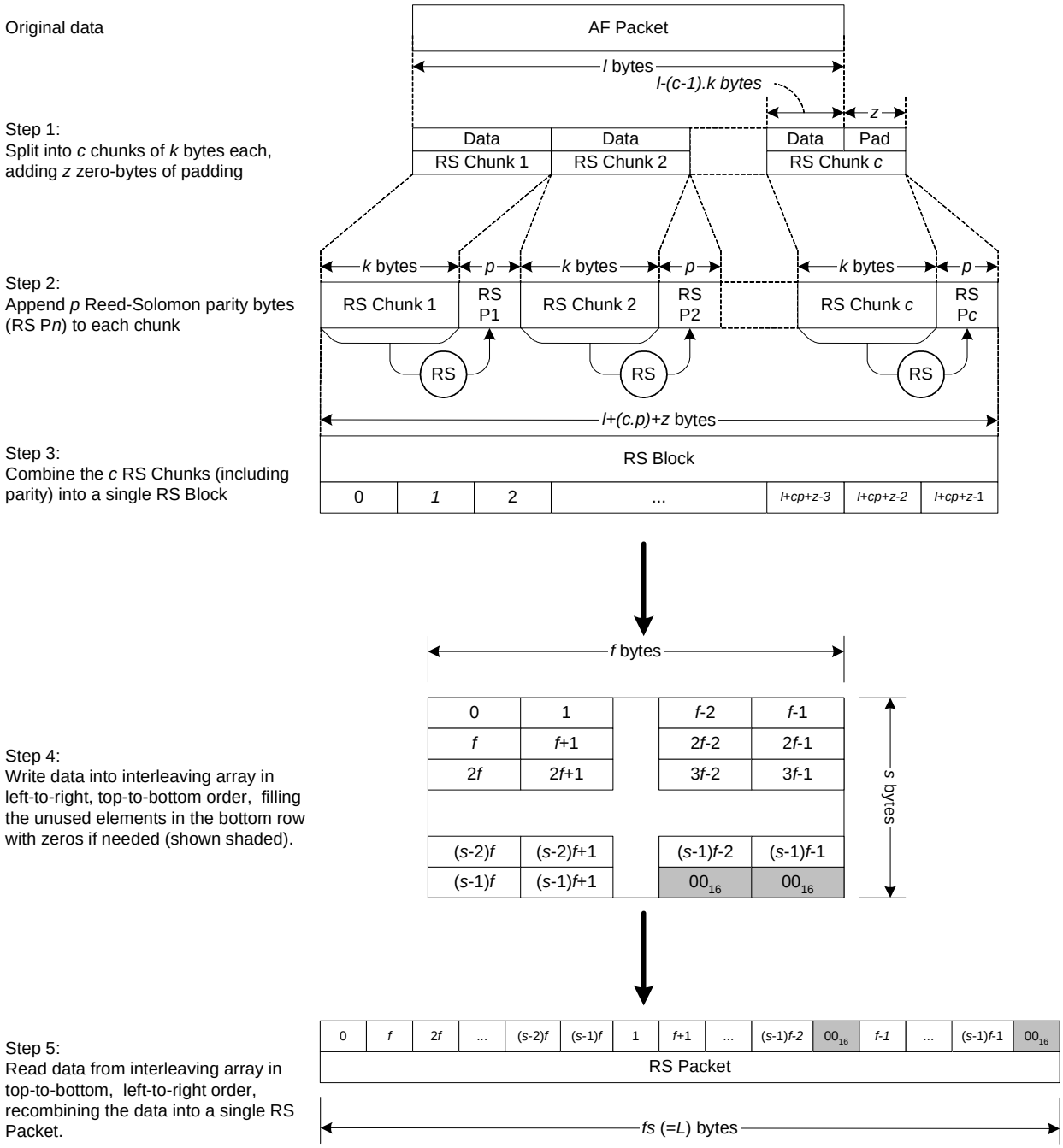


Figure 12: Generation of Reed Solomon Packet

The full Reed Solomon code used shall be RS(255,207) calculated over the Galois Field GF(2^8) using the generator polynomial $P(x)=x^8+x^4+x^3+x^2+1$.

When the calculated value for *k* is less that 207, bytes *k*…206 (inclusive) encoded by the RS(255,207) code shall all be zero and shall not be included in the resulting RS Block, thus producing an RS(*k*+*p*,*k*) code.

The code polynomial shall be $G(x) = \prod_{i=1}^{48} (x - \alpha^i)$.

7.3.2 Fragmentation

Fragmentation may be applied directly to an AF Packet or to a pre-processed RS Packet.

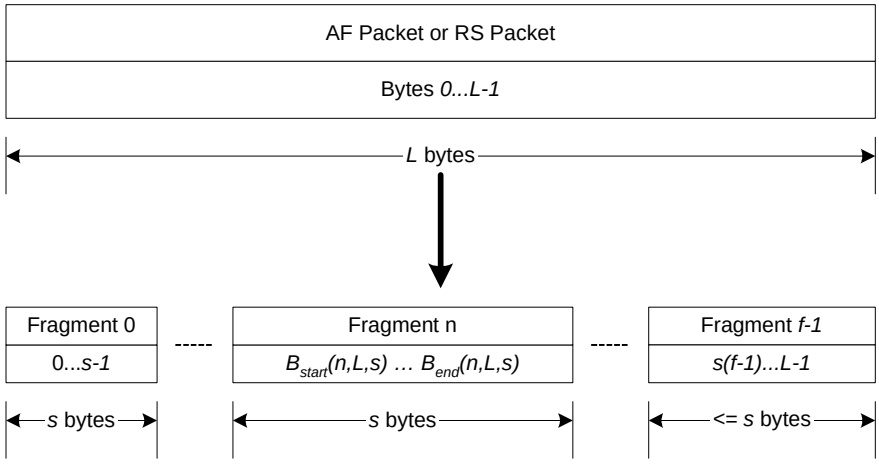
The fragmentation splits the data from the original AF or RS Packet into a number of separate fragments. When carrying an interleaved Reed Solomon Packet, up to *m* of these fragments can be lost from a each packet without losing data. The fragmenting process is shown diagrammatically in figure B.3.

Every PFT Fragment produced from a single AF or RS Packet shall have the same values in all of the PFT Header fields except for the ***Findex***, ***Plen*** and ***HCRC*** fields.

The ***Findex*** field shall contain a count that shall start with zero and increment by one for each fragment.

The ***Plen*** field of all fragments shall be the *s* for the initial *f-1* fragments and *s - (L%f)* (modulus operator) for the final fragment. Note that when Reed Solomon has been used, all fragments will be of length *s*.

The ***HCRC*** field shall be calculated correctly for each PFT Fragment.



$$B_{start}(n, L, s) = n.s$$
$$B_{end}(n, L, s) = MIN[s(n+1) - 1, L - 1]$$

Figure 13: Fragmentation

7.3.3 Transport addressing

The PFT Layer addressing fields ***Source*** and ***Dest*** are intended to be used to identifier the sender (***Source***) and recipient (***Dest***) of a packet. The value $FFFF_{16}$ shall be used to indicate 'broadcast', all other values indicate a specific address. If a device is configured with specific source and/or destination addresses, it shall ignore all PFT Fragments received with an incorrect non-broadcast address.

7.4 Decoding process

Decoding the PFT Fragments is a four stage process. The four stages are, in order:

- synchronize;
- discard incorrectly addressed fragments (if transport addressing enabled);
- de-fragment (if either Reed Solomon or Simple Fragmentation has been used);
- Reed Solomon error detect and correct (if Reed Solomon enabled).

7.4.1 Synchronization

For streaming links (e.g. asynchronous serial or TCP/IP) the following process shall be used for synchronization of the incoming stream. Synchronization may also be applicable when reading a file.

i) Detect the bit-pattern 0101000001000110_2 (5046_{16}) corresponding to the ASCII sync-word "PF" to find the start of a candidate PFT Header.

ii) Calculate the CRC over the candidate header - this may be performed efficiently by a continuous byte-wide CRC implementation utilizing the fact that the CRC of the PFT Header including the ***CRC*** field itself will result in the constant value $1D0F_{16}$.

iii) Check that the header length matches the options selected: if too short, continue from step (ii), if too long, return to step (i), if correct, synchronization has been achieved and the ***Plen*** field can be used to determine the length of the fragment.

For packet-based links (e.g. UDP/IP), the concept of synchronization is unnecessary as the transport protocol will present complete fragments to the PFT layer. It is only necessary to check that the CRC and length are correct before passing the packet on for further processing. In the event that the CRC or length are incorrect, the packet may be assumed to be corrupted and discarded.

7.4.2 Transport addressing

Each PFT Fragment may contain a source and destination address. If present, the address fields shall be examined and if they do not match the configuration of the unit, the entire fragment shall be discarded.

PFT Fragments which do not contain the optional Transport header shall never be discarded.

7.4.3 Defragmentation

Defragmentation is the reverse process to fragmentation. Memory management is kept simple since it is known that all fragments (except the last one) are the same size. The last fragment will be the same size or smaller than the preceding fragments.

If Reed Solomon is not in use, every fragment must be received correctly and completely in order to re-construct the original AF Packet.

When Reed Solomon is in use, the Forward Error Correction code can be used to attempt recovery of the original AF Packet before all of the fragments have been received, see clause 7.4.4.

7.4.4 Reed-Solomon decoding

Reed Solomon decoding is the reverse process to encoding.

The length of the RS Packet can be calculated as follows, where:

- *f* is the number of fragments produced from this packet, carried in the ***Fcount*** header field.
- *s* is the size in bytes of the PFT fragments as carried in the ***Plen*** header field of all fragments except for the last fragment (where ***Fcount*** equals ***[Findex - 1]***).
- *k* is the data size in bytes of the Reed Solomon code as carried in the ***RSk*** header field.
- *z* is the number of Reed Solomon padding bytes as carried in the ***RSz*** header field.
- *p* is the number of Reed Solomon parity bytes and shall have the value 48.
- c_{max} is the maximum number of Reed Solomon chunks that may have been sent.
- Rx_{min} is the minimum number of fragments which must be received before Reed Solomon decoding can be attempted. Additional fragments may be required if errors have occurred or if one of the fragments received is the last fragment.

$$c_{\max} = \left\lfloor \frac{fs}{k+p} \right\rfloor$$

$$Rx_{\min} = \left\lceil \frac{c_{\max} n}{s} \right\rceil$$

Once Rx_{min} PFT Fragments have been received, the remaining bytes can be filled with zeros and Reed Solomon decoding attempted, with success being indicated by a valid AF Packet being produced with a correct CRC. In the case that bit-errors have occurred, more PFT Fragments will be needed before the original AF Packet can be correctly decoded.

Note that the padding bytes added during the Reed Solomon interleaving process may result in more data being recovered than was originally sent for very large packets. This additional data will be all zeros, and can be differentiated from the *z* zero bytes added during Reed Solomon encoding using the value of the ***RSz*** header field. The size of the final AF Packet can be determined from the AF ***LEN*** field.

Annex A (normative): Calculation of the CRC word

The implementation of Cyclic Redundancy Check codes (CRC-codes) allows the detection of transmission errors at the receiver side. These CRC words shall be defined by the result of the procedure described in this annex.

A CRC code is defined by a polynomial of degree *n*:

$$G(x) = x^n + g_{n-1}x^{n-1} + \ldots + g_2x^2 + g_1x + 1$$

with $n \geq 1$

and $$g_i \in \{0,1\}, \quad i = 1.....n-1$$

The CRC calculation may be performed by means of a shift register containing *n* register stages, equivalent to the degree of the polynomial (see figure B.3). The stages are denoted by b_0 ... b_{n-1}, where b_0 corresponds to 1, b_1 to *x*, b_2 to x^2, b_{n-1} to x^{n-1}. The shift register is tapped by inserting XORs at the input of those stages, where the corresponding coefficients g_i of the polynomial are "1".

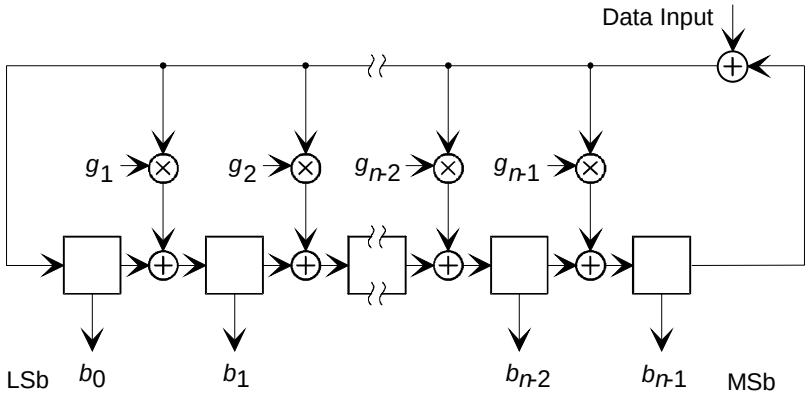


Figure A.1: CRC Generator

At the beginning of the CRC calculation, all register stage contents are initialized to all ones.

After applying the first bit of the data block (MSb first) to the input, the shift clock causes the register to shift its content by one stage towards the MSb stage (b_{n-1}), while loading the tapped stages with the result of the appropriate XOR operations. The procedure is then repeated for each data bit. Following the shift after applying the last bit (LSb) of the data block to the input, the shift register contains the CRC word, which is then read out. The data and CRC words are transmitted MSb first.

The CRC shall be inverted (1's complemented) prior to transmission.

The generator polynomial $G(x) = x^{16} + x^{12} + x^5 + 1$ shall be used.

If the CRC is appended to the original data, a second CRC calculated over the entire length will result in the constant value $1D0F_{16}$.

Annex B (normative): Physical mapping

The actual transmission of PFT (or AF) Packets shall be possible over as many kinds of existing transmission infrastructure as possible. Three possible physical mappings are defined in this annex, however this list is neither exhaustive nor prescriptive. New mappings may be defined in the future.

B.1 Packet links

Packet switching networks are becoming very common, and in many cases either PFT Fragments or AF Packets may be mapped one-for-one onto link packets. The MTU for the link layer needs to be observed and PFT Fragmentation or PFT Reed Solomon Fragmentation should used to ensure that a single source packet will not be fragmented by the link layer.

B.1.1 UDP/IP

UDP/IP is one of the most popular packet-switching networks at present. Both PFT Fragments and AF Packets may be mapped one-for-one onto UDP/IP packets provided due care is taken to observe the MTU of the UDP/IP layer. PFT Fragmentation or PFT Reed Solomon Fragmentation should be used to ensure that a single source packet would not be fragmented. Any defined physical interface for UDP/IP may be used, including (but not limited to) 10Base-T Ethernet or PPP.

UDP/IP provides source and destination port numbers, hence the use of the optional PFT Transport Header is not likely to be necessary, however its use is not prohibited.

UDP/IP does not guarantee delivery of packets, hence the use of the PFT Reed Solomon Header is strongly recommended, but is not mandated.

B.2 Streaming links

A streaming connection in this context describes any type of non-packetized connections. Examples of such connections include asynchronous serial links, synchronous serial links and TCP/IP links. The distinguishing feature of such links is that the higher level layers (e.g. AF or PFT layer) is required to establish packet-synchronization before attempting to decode the stream.

The use of the PFT layer is strongly recommended as this has been designed to offer improved synchronization reliability compared to the raw AF layer, however the raw AF layer may be used directly if required.

For physical links which do not guarantee error-free reception, it is recommended that the optional Reed Solomon FEC mechanism provided by the PFT Layer is used. Links such as TCP/IP are guaranteed to provide in-order error-free links and so do not need the Reed Solomon protection, however RS-232 links may be subject to errors and hence benefit greatly from the addition of Reed Solomon FEC.

B.3 File

PFT Fragments or AF Packets may be stored in a file for offline distribution, archiving or any other purpose. A standard mapping has been defined using the Hierarchical TAG Item option available, however other mappings may be defined (or this one extended) in the future. The top level TAG Item has the TAG Name *fio_* and is used to encapsulate a TAG packet, one part of which is the AF Packet or PFT Fragment in an *afpf* TAG Item. Additional TAG Items have been defined to monitor the reception or control the replay of the packets.

B.3.1 File IO (fio_)

The fio_ TAG Item is the highest layer TAG Item in the file TAG Item hierarchy.

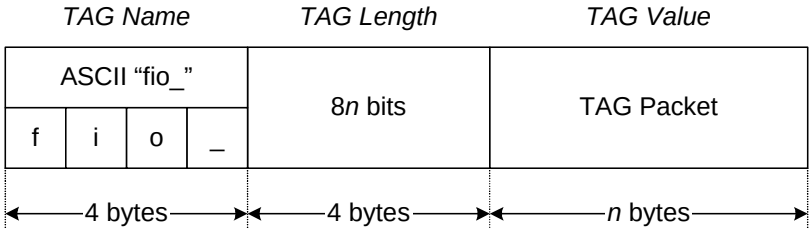


Figure B.1: File Input/Output

This TAG Item acts as a container for an *afpf* TAG Item. A *time* TAG Item may optionally be present.

B.3.1.1 AF Packet / PFT Fragment (afpf)

The *afpf* TAG Item contains an entire AF Packet or PFT Fragment as the TAG Value.

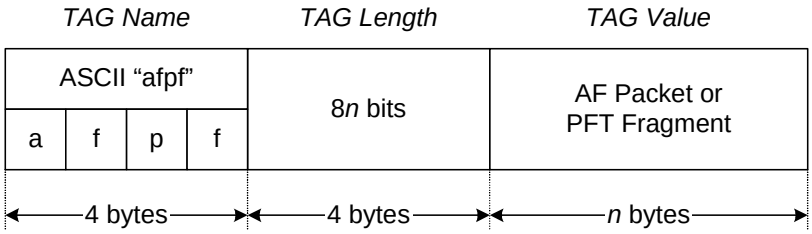


Figure B.2: AF Packet or PFT Fragment

B.3.1.2 Timestamp (time)

The *time* TAG Item may occur in the payload of any *fio_* TAG Item. It may record the time of reception of the payload, or it may indicate the intended time of replay. The time value given in the TI_SEC and TI_NSEC fields may be relative to the start of the file, or any other reference desired.

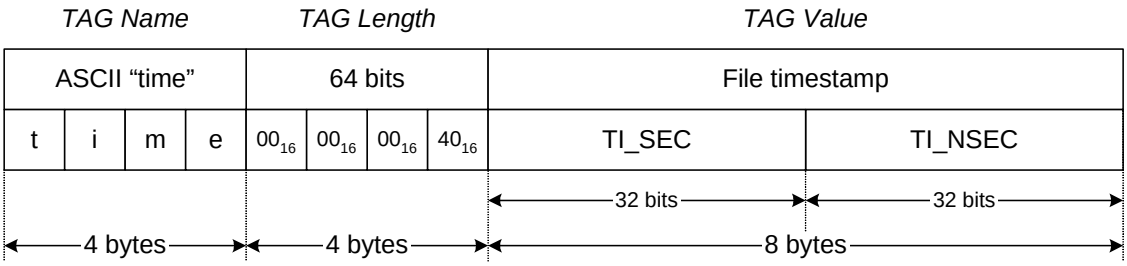


Figure B.3: File timestamp

TI_SEC: the number of whole SI seconds, in the range 0…(2^{32}-1).

TI_NSEC: the number of whole SI nanoseconds, in the range 0…999 999 999. Values outside of this range are not defined and shall not be used.

Annex C (informative): Signalling of basic transmission layers and parameters

Since several basic transmission protocols for PFT (or AF) Packets are defined, it is sensible to also define a common method for addressing and identifying each of these protocols together with their individual parameters. These informative definitions are intended as suggestions for organizations using the DCP protocol to be able to describe DCP sources and targets in a common way.

The following definitions are formatted according to the URI specifications ("Uniform Resource Identifier"; see RFC 2396 [1] and RFC 2718 [2]). This notation is also used for other protocols like FTP and HTTP.

- Items in square brackets identify optional elements.
- Items in angle brackets identify named elements.
- Scheme strings and parameter names/values should be treated as case-insensitive.
- Most parameters are optional; if such a parameter is not specified, the (default) value shall be assumed (if no (default) value is specified, the parameter is NOT optional).
- Additional parameters may be defined by applications using the Distribution and Communication Protocol (e.g. maximum transmission time or maximum number of bytes recorded).

General DCP address layout:

- <scheme>:<target>[:[<src-addr>:]<dst-addr>]] [?<param>=<val>[&<param>=<val>[&…]]]

General items:

- <scheme> specifies the basic protocol to be used, for example "*dcp.udp.pft*" or "*dcp.tcp*". All DCP-based schemes shall begin with the text "dcp.". If ".pft" is present, the PFT protocol is used together with the specified basic transport protocol; if absent, AF Packets are directly transmitted using the specified basic transport protocol.
- <src-addr> and <dst-addr> both default to 0; if both are omitted, the optional address header is not included in the PFT Packet Header; both values are ignored if ".pft" is not present in the <scheme> section (thus pure AF Packets are transmitted, while the PFT Layer is bypassed) except if required by the basic transport protocol itself. It is not possible to specify different addresses to both the PFT layer and link-layer simultaneously.

General parameters (available identically for all basic transport layers if not explicitly mentioned):

- "crc" (disabled: "f", "false", "0"; enabled: "t", "true", "1"; default: "1") enables or disables the CRC calculation for the AF Layer.
- "fec" (disabled: "0"; enabled: "1"…"9"; default: "0") enables/disables the FEC protection mechanism of the PFT Layer (ignored if <scheme> does not contain ".pft"); value 1..9 defines the strength of the protection (may e.g. influence the number of expected packet losses).
- "maxpaklen" (no limit: "0"; default: "0") specifies, in bytes, the MTU of the link for the PFT layer. Not supported if the PFT layer is not in use.

C.1 DCP over UDP/IP

- <scheme> has the value "dcp.udp[.pft]".
- <target> shall be a hostname or IP address preceded by "//".
- <src-addr> is the UDP/IP port number at the source host in the range 0…(2^{16}-1).
- <dst-addr> is the UDP/IP port number at the destination host in the range 0…(2^{16}-1).

EXAMPLE: dcp.udp.pft://192.168.0.1:3002?fec=9&crc=0.

C.2 DCP over transparent (serial) links

- <scheme> has the value "dcp.ser[.pft]".
- <target> is a system specific device identifier, e.g. 'COM4' or '/dev/ttyS1'.
- <src-addr> has the value of the PFT Packet Header field SRC, in the range 0…(2^{16}-1).
- <dst-addr> has the value of the PFT Packet Header field DST, in the range 0…(2^{16}-1).

The following additional parameters are defined for this scheme:

- "bitrate" (numeric, e.g. "115200").
- "flowctrl" ("xonxoff", "rtscts"/"hw" or "none" (default)).

EXAMPLE 1: dcp.ser.pft:/dev/ttyS3:1:2?bitrate=4800&fec=4&flowctrl=hw.

EXAMPLE 2: dcp.ser:COM2:200?bitrate=115200.

C.3 DCP to/from a file using File IO

- <scheme> has the value "dcp.file[.pft]".
- <target> shall be a system specific file name, including any path required.
- <src-addr> has the value of the PFT Packet Header field SRC, in the range 0…(2^{16}-1).
- <dst-addr> has the value of the PFT Packet Header field DST, in the range 0…(2^{16}-1).

EXAMPLE 1: dcp.file:/tmp/record_1/test.dcp.

EXAMPLE 2: dcp.file.pft:c:\temp\test.dcp:99:100.

C.4 DCP over TCP/IP

- <scheme> has the value "dcp.tcp[.pft]".
- <target> shall be a hostname or IP address preceded by "//".
- <src-addr> is the TCP/IP port number at the source host in the range 0…(2^{16}-1).
- <dst-addr> is the TCP/IP port number at the destination host in the range 0…(2^{16}-1).

EXAMPLE: dcp.tcp.pft://192.168.0.1:1002:3002?fec=1&crc=0.

History

Document history		
V1.1.1	December 2003	Publication