ETSI GS NIN 005 V1.1.1 (2022-11)

GROUP SPECIFICATION

Non-IP Networking (NIN); Signalling messages and protocols

Reference

DGS/NIN-005

Keywords

ASN.1, core network, fixed networks, flexilink, layer 3, network management, next generation protocol, QoS, routing support, SDN, self-organizing, signalling, slicing, synchronization, UNI

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 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Non-IP Networking (NIN).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies procedures and packet formats for network processes including: setting packet flows up, modifying and re-routing them and clearing them down; exchanging information on timing and synchronization and conveying user messages that are not part of a flow.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

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

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI GS NGP 013: "Next Generation Protocols (NGP); Flexilink: efficient deterministic packet forwarding in user plane for NGP; Packet formats and forwarding mechanisms".

[2] IEEE™: "Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID)".

NOTE: Available from https://standards.ieee.org/wp-content/uploads/import/documents/tutorials/eui.pdf.

[3] IANA: "SMI Private Codes".

NOTE: Available from https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-25.

[4] Recommendation ITU-T E.164: "The international public telecommunication numbering plan".

NOTE: Available from https://www.itu.int/itu-t/recommendations/rec.aspx?rec=10688.

[5] IETF RFC 791: "Internet Protocol".

[6] IETF RFC 8200: "Internet Protocol, Version 6 (IPv6) Specification".

[7] IETF RFC 1213: "Management Information Base for Network Management of TCP/IP-based internets: MIB-II".

[8] IEEE 802™-2014: "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture".

NOTE: Available from https://ieeexplore.ieee.org/browse/standards/get-program/page/series?id=68.

[9] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[10] IETF RFC 4122: "A Universally Unique IDentifier (UUID) URN Namespace".

[11] SMPTE ST 330: "Unique Material Identifier (UMID)".

[12] IETF RFC 2579: "Textual Conventions for SMIv2".

[13] IETF RFC 1700: "Assigned Numbers".

[14] The Unicode Consortium: "The Unicode® Standard - Core Specification".

NOTE: Version 15.0 is available from https://www.unicode.org/versions/Unicode15.0.0/UnicodeStandard-15.0.pdf.

[15] Recommendation ITU-T X.690: "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

NOTE: Available from https://www.itu.int/itu-t/recommendations/rec.aspx?rec=14472.

[16] AES67: "AES standard for audio applications of networks - High-performance streaming audio-over-IP interoperability".

[17] IEEE 802.3™: "IEEE Standard for Ethernet".

NOTE: Available from https://ieeexplore.ieee.org/browse/standards/get-program/page/series?id=68.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Recommendation ITU-T Q.850: "Usage of cause and location in the Digital Subscriber Signalling System No. 1 and the Signalling System No. 7 ISDN user part".

NOTE: Available from https://www.itu.int/itu-t/recommendations/rec.aspx?rec=13695.

[i.2] IETF RFC 1034: "Domain Concepts and Facilities".

[i.3] IETF RFC 3261: "SIP: Session Initiation Protocol".

[i.4] IETF RFC 4271: "A Border Gateway Protocol 4".

[i.5] ATM Forum af-pnni-0055.002: "Private Network-Network Interface Specification Version 1.1".

NOTE: Available from https://www.broadband-forum.org/technical/download/af-pnni-0055.002.pdf.

[i.6] AES5: "AES recommended practice for professional digital audio - Preferred sampling frequencies for applications employing pulse-code modulation".

[i.7] AES51: "AES standard for digital audio - Digital input-output interfacing - Transmission of ATM cells over Ethernet physical layer".

[i.8] ETSI TS 136 413: "LTE; Evolved Universal Terrestrial Radio Access Network (E-UTRAN); S1 Application Protocol (S1AP) (3GPP TS 36.413)".

[i.9] IEEE 802.1D™: "IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS NGP 013 [1] and the following apply:

cloud: set of islands connected together by loosely synchronized links

island: set of units connected together by tightly synchronized links

loosely synchronized link: See clause 7.1.1.

management information base: collection of objects, each of which is identified by an object identifier and has a value that can be expressed in ASN.1

network port: connector that is part of a unit and can terminate a point-to-point link that carries Flexilink packets

object identifier: globally unique value, in the form of a sequence of integers, associated with an object to unambiguously identify it

tightly synchronized link: See clause 7.1.1.

transaction: exchange of a set of messages between control plane entities that results in a change in the state of the system

unit: physical piece of equipment, such as a network switch or interface, or a virtual object with a similar role

3.2 Symbols

Void.

3.3 Abbreviations

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

AES	Audio Engineering Society
ASN.1	Abstract Syntax Notation 1
BER	Basic Encoding Rules
CRC	Cyclic Redundancy Check
DARS	Digital Audio Reference Signal
DHCP	Dynamic Host Configuation Protocol
EUI	Extended Unique Identifier
IANA	Internet Assigned Numbers Authority
IE	Information Element
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering TaskForce
IP	Internet Protocol
ISP	Internet Service Provider
ITU-T	International Telecommunication Union - Telecommunication standardization sector
MAC	Media Access Control
MIB	Management Information Base
NGP	Next Generation Protocols
NUL	Null (Unicode code point zero)
OID	Object IDentifier
OUI	Organizationally Unique Identifier
PHY	PHYsical layer
PTP	Precision Time Protocol
SDN	Software Defined Networking
SFP	Small Form Pluggable
SMPTE	Society of Motion Picture and Television Engineers

TAI	International Atomic Time
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UMID	Unique Material IDentifier
URI	Uniform Resource Identifier
UTF-8	Unicode Transformation Format - 8 bit
UUID	Universally Unique IDentifier

4 Identifiers

4.1 Bit and byte order

All multiple-octet quantities shall be coded with the most significant octet first.

Bits within any value shall be numbered from left to right, with bit 0 being the most significant bit.

NOTE: It follows that if the octets of a multiple-octet value are numbered from the left with the most significant octet being octet 0, bit *n* of octet *m* will be bit 8*m*+*n* of the value.

4.2 Unit identifiers

A "unit" is typically a piece of physical equipment such as a network switch, a computer, or an IoT device, but may also be a virtual object with a similar role.

Each unit that is part of a Flexilink network shall have a globally-unique 64-bit identifier constructed as follows.

If bits 6 and 7 are 00, it shall be an EUI-64 identifier as specified in [2].

If the first octet contains the binary value 00000001, the next few octets shall be a Private Enterprise Number [3] coded as in an arc of an ASN.1 OID according to BER, and the remainder shall be assigned by the owner of the Private Enterprise Number.

NOTE 1: This provides blocks of globally-unique values that do not require an OUI to be purchased. Private Enterprise Numbers are allocated sequentially, and as of 15th October 2021 the largest was 58 012. Numbers up to two million can be coded in 3 octets, so the owner has a block of four billion identifiers that can be used.

All other code points with 0 in bit 6 are reserved.

Any identifier with 0 in bit 6 shall be assigned permanently to the unit.

An identifier with 1 in bit 6 shall have the same structure as the equivalent identifier with 0 in bit 6, but shall be assigned transiently by an adjacent unit. The assignment only lasts for as long as its link to the unit that assigned the identifier is up.

NOTE 2: This is intended for assigning identifiers to devices with "simple" interfaces and to processes running in equipment that is not Flexilink-aware.

Equipment connected via other technologies (e.g. RS232 or UDP/IP) may have its own identifier, or may be identified indirectly via its point of attachment (see address type 9 in Table 4.4.1).

Each interface that is capable of supporting standards in the IEEE 802 [8] family shall also have a 48-bit MAC address. A MAC address is not required for an interface that only supports Flexilink frames.

NOTE 3: A MAC address identifies an interface, not a piece of equipment. There is no relationship between the various MAC addresses and the identifier as far as the protocols are concerned, though in practice they can all be related to the unit's serial number.

A unit may also have other addresses such as an E.164 address [4], an IPv4 address [5] and/or an IPv6 address [6], and other forms of identification such as the MIB object `sysName` [7].

4.3 Call, route and flow identifiers

4.3.1 Calls

Flows that have the same endpoints may be bundled together to form a "call".

EXAMPLE 1: A call between a studio and an outside broadcast unit can be composed of flows which carry, respectively: programme audio; programme video; talkback; and automation data.

There shall be two kinds of call, unicast and multicast.

A unicast call conveys data between two units. It is created by one of the units originating a FindRoute transaction, and flows and routes can be added by AddFlow transactions or removed by Terminate transactions, see Table 5.4.1.

A multicast call conveys data between a "source" unit and one or more "destination" units. It is created by a destination unit originating a FindRoute transaction, and further destination units can request to be added to the call by originating FindRoute transactions. A destination can leave the call by originating a Terminate transaction.

NOTE 1: Multicast calls are intended for guaranteed service flows in the direction from source to destination. To carry basic service flows in the same direction switches would need to support copying of basic service packets, and a switch that did not have that capability would not be able to carry the flow. Similarly, the way in which guaranteed service flows are routed restricts any flow towards the source to come from a single destination. A basic service flow can be carried from multiple destinations to the source, provided the packet payloads include an identification of the sender.

NOTE 2: Connection to a multicast is controlled by the destination; this is similar to the way point-to-point audio and video are switched, where destinations "take" from a source. It is different from multicast calls in ATM, where destinations were added or removed by the source, and avoids the requirement for an "endpoint identifier" which in ATM was 15 bits and thus limited the number of destinations to 32 767.

EXAMPLE 2: Remote performance can be set up using a single call with each performer being a destination and the mixing desk being the source, and a separate guaranteed service flow from each performer to the desk. The mix is then carried by a single flow to all the performers. Listeners can join the same call by simply not requesting the flow towards the desk.

4.3.2 Routes

A "route" is the set of links over which a call is carried.

Copies of a flow may be transmitted over different routes to add resilience.

Different routes may carry different sets of flows, for instance a backup route may be set up to carry only the most important flows, or different kinds of flow may have different backup routes, or a backup route may be established without connecting any flows over it.

4.3.3 Identifiers

Each flow shall be identified in the control plane by a 128-bit value formatted as shown in Figure 4.3.1.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                           owner                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        call reference                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  route ref  |D|                flow reference                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4.3.1: Structure of a flow identifier

The first 64 bits shall contain the identifier of the unit which created the flow identifier. This unit is referred to in the present document as the "owner" of the call, and of the flow identifier.

In the case of a unicast call, the owner shall be the unit that originated the FindRoute transaction. In the case of a multicast, the owner shall be the source or a unit associated with the source.

NOTE 1: "A unit associated with the source" can be another unit that, for resilience, is transmitting the same content over a different route. It can also be a control entity that manages media sources.

NOTE 2: If multicast flows were not owned by the source, in the case where A asks to consume content hosted by B and then C asks for it also and then A disconnects, there would be a call between B and C that is owned by A, and A would not in general be able to know when the call reference can be re-used.

The next 32 bits shall contain a non-zero reference number, chosen by the owner, such that the first 96 bits are a globally unique "call identifier" for the call of which the flow is a part. Call references should be chosen in a way that maximizes the time from when a call ceases to exist until its identifier is re-used for a new call.

The next 7 bits shall contain a non-zero reference number, chosen by the owner and unique within the call, selecting the route followed by the flow. The call identifier and the route reference together form a "route identifier".

The next bit shall contain a "direction" indicator which shall be 0 for a flow in the direction away from the owner and 1 for a flow in the direction towards the owner.

NOTE 3: A route can carry flows in both directions. When connecting to a server, the client asks for an uplink flow and a downlink flow. The client is the owner of the call, and specifies the two flows it needs. The routing table entries for a flow are set up as the FindRoute messages are passed through the network.

The remaining 24 bits shall contain a non-zero reference number, chosen by the owner, identifying the flow uniquely within the call. Where a pair of flows in opposite directions carries a two-way protocol such as TCP, the same flow reference value shall be used for each of them.

NOTE 4: If a call is connected over several different routes (for instance, to give resilience in the event of failure) the route reference distinguishes between them. The flow reference distinguishes between different flows within a call.

NOTE 5: The flow reference is orthogonal to the route reference, so flows carrying the same content over different routes have the same flow reference, and flows carrying different content have different flow references even if they do not follow the same route.

NOTE 6: Most calls will only have a very small number of flows, but the flow reference field is able to support a large number for applications such as that in example 2 to clause 4.3.1.

Zero in the route reference field shall indicate all routes for the call. Except where specified otherwise, zero in both the "direction" bit and the flow reference field shall indicate all flows that are included in the call, and zero in the flow reference with 1 in the "direction" bit is reserved.

A caller wishing to join a multicast shall use a temporary call identifier (which it owns) until the source's identifier for the multicast has been discovered.

NOTE 7: An existing multicast will already have an identifier (and if the "openness" value specified in clause 5.7.16 is 00 or 01 the caller can join it without the FindRoute propagating all the way to the source); otherwise the source creates it when it sends the response message.

NOTE 8: Flow identifiers are globally significant and globally unique, so can be used in control plane messages to unambiguously identify a call, route, or flow at any point in the system. They are not intended to be used in packet headers, where a locally significant identification (label or slot allocation) is both more economical and more secure.

4.4 Addresses

An address may serve to identify the endpoint to which a call is to be connected, such as a particular service, interface, unit, or piece of content; or to locate it within the topology of the network; or both.

NOTE 1: Addresses are not restricted to a single addressing scheme, and can include a number of different forms of identification, including but not limited to: conventional interface addresses such as IPv4 [5], IPv6 [6], and IEEE 802 [8] MAC (see IEEE 802 [8], clause 8); the unit identifiers specified in clause 4.2 of the present document; MIB objects such as `sysName [7]`; and identification of a service or piece of content (e.g. URI [9], UUID [10] or UMID [11]).

An address is represented as an octet string, and is thus a valid `TAddress` value as specified in IETF RFC 2579 [12], unless it is more than 255 octets long. The corresponding `TDomain` value is for further study.

The interpretation of an address depends on the "type" in its first octet, and anything that does not recognize the value in that octet will not be able to interpret the address or route calls to it. Values for the "type" are listed in Table 4.4.1.

Table 4.4.1: Address encoding

Type	Remainder of octet string
0	second octet contains *n*, next *n* octets are an address (which shall not be of type 0) which acts as the locator, remainder is another address (the "local" address, which may be of type 0) which is to be interpreted at the specified location
1	second octet contains *n*, next *n* octets are an AbsoluteOid, remainder is the object's value (see notes 1, 3 and 4)
2	second octet contains *n*, next *n* octets are an AbsoluteOid, remainder is the object's value (see notes 2, 3 and 4)
3	second octet contains *n*, next *n* octets are a TDomain value (coded as an AbsoluteOid), remainder is a TAddress value appropriate to that domain (see IETF RFC 2579 [12]) and notes 3 and 4)
4	either 4 octets containing an IPv4 address [5] or 8 octets consisting of 4 octets containing an IPv4 subnet address followed by 4 octets containing a subnet mask
5	unit identifier (see clause 4.2)
6	IPv6 address [6]
7	URI [9]
8	second octet contains a protocol number (as in IP headers [5] and [13], e.g. 6 for TCP and 17 for UDP); remainder contains a port number for the indicated protocol
9	identifier for a point of attachment to a piece of equipment (physical or virtual; local to the equipment)
10	service name coded as a UTF-8 string as specified in The Unicode® Standard [14], clause 2.5
11	E.164 address [4]
12	IEEE 802 MAC address (see IEEE 802 [8], clause 8)
13	UUID [10]
14	UMID [11]
15 to 255	reserved

NOTE 1: Type 1 shall identify a unit by searching for one in whose MIB the object identified by the OID (such as sysName or sysLocation [7]) has the specified value. In the case of scalar objects, the OID may omit the final zero arc. Units are not required to support all (or, indeed, any) objects in their MIBs that could be used in this context.
NOTE 2: Type 2 shall identify a network port, media port or other resource within a unit, by searching the unit's MIB, usually for a columnar object such as a table of the names assigned to physical media ports. The OID omits the index arcs. As with type 1, units are not required to support all objects in their MIBs.
NOTE 3: See clause 5.6.4 for the coding of an AbsoluteOid. The value which follows the AbsoluteOid in types 1 and 2 shall be coded using ASN.1 Basic Encoding Rules (see Recommendation ITU-T X.690 [15], clause 8), including the tag and length.
NOTE 4: Types 1, 2 and 3 should only be used for address formats that cannot be supported by any of the other types. An IPv4 address, for instance, should use type 4 although it can also be expressed as a type 3 address or, if the target unit's MIB includes its IP address, as a type 1. An IPv4 address together with a port number should be type 0, with a type 4 locator and a type 8 local address. For specifying an audio interface on a piece of equipment, type 9 should be used in preference to type 2.

NOTE 2: An address can be "source routed", i.e. consist of a series of addresses such that the call is routed to the first address, then from there to the next, etc, using type 0, with each address being interpreted in the context of the equipment, location or subnetwork specified by the previous address.

EXAMPLE 1: The address for an audio call can consist of the address of a piece of audio equipment followed by an identification of a particular audio port on that equipment.

EXAMPLE 2: The address of a gateway (or even of a specific network port on a gateway or switch) can be used as the locator for equipment accessed through that gateway.

5 Transactions and messages

5.1 Control plane entities and signalling flows

Each unit shall be associated with a control plane entity, which controls the routing in its forwarding plane. Appropriate measures shall be taken to prevent the unit's routing being configured otherwise than by its control plane entity.

NOTE 1: This is straightforward if the two parts (control plane and forwarding plane) are in the same physical unit. Where a single entity controls a number of units, as is the case with SDN, any communication link over which the commands to update the routing tables are sent will need to be adequately protected.

Defining the set of units controlled by a control plane entity as a "cluster", wherever there is a link between units that are in different clusters the two control plane entities are "neighbours" and there shall be a "signalling flow", which is a bidirectional pair of basic service flows, connected between them. Signalling flows shall not exist between any entities that are not neighbouring control plane entities. Signalling data units shall only be carried over signalling flows, and signalling flows shall only carry signalling data units.

Where needed for clarity, flows that are not signalling flows are referred to in the present document as "user flows".

If there is more than one link between a pair of clusters, the control plane entities may be linked by a single signalling flow, or one per link, or some intermediate number.

NOTE 2: If a control plane entity is located within the unit it controls, each link can carry a signalling flow. The present document does not specify how a control plane entity communicates with a unit's forwarding plane, nor how signalling flows are carried between neighbouring entities when they are not located within the units they control.

The specifications in the present document assume end equipment (i.e. a unit that can terminate flows) does not share control plane entities with switches (units through which flows can be routed), even if they share the same physical hardware. Control plane entities that control end equipment are "terminal" entities; those that control switches or clusters of switches are "intermediate" entities.

NOTE 3: This allows the terminal entity to take the responder role while the other takes the intermediate role, see clauses 5.3 and 6.1.3.1.

Each control plane entity shall keep information on each flow that passes through any unit that it controls, including (but not limited to) information sufficient to allow a Terminate transaction to remove all forwarding plane routing for the flow.

NOTE 4: If most of the information from the FindRoute transaction is retained, flows can be more easily rerouted around a failure and FindRoute requests to receive an existing multicast can be recognized.

5.2 Transaction configuration and message classes

5.2.1 Data units

The control plane entity transmitting a signalling data unit is referred to in the present document as the "sender". The data unit shall be sent to a specific neighbouring control plane entity which is referred to as the "recipient".

There are two kinds of signalling data unit: message and acknowledgement.

The transaction of which a data unit is part is identified by the transaction type and the IEs listed in Tables 5.5.1 (for messages) or 5.5.2 (for acknowledgements).

Transactions shall be implemented by the exchange of signalling messages between neighbouring control plane entities.

Each acknowledgement shall identify a specific message which was transmitted in the opposite direction over the same link, and shall serve to confirm that the message has been received and will be processed unless prevented by a condition which also causes the link to be disconnected. The recipient of an acknowledgement which does not match any message for which an acknowledgement is awaited should silently ignore it.

The sender of a message shall treat an acknowledgement or reply as an indication that the message has been received and has been (in the case of a reply) or will be (in the case of an acknowledgement) processed. A control plane entity shall not send an acknowledgement if it is possible that the message could be lost (for instance, through lack of buffer space while awaiting processing in the protocol stack) unless such a loss would be accompanied by a reset of the entity or its link to the sender.

NOTE: The sender of an acknowledgement therefore takes responsibility for further processing of the transaction of which the message is a part. See also clause 5.3.2.

5.2.2 Transaction configuration

The entities participating in a transaction and the signalling flows connected between them shall form a tree, with the "originator" at the root and a "responder" at each leaf. Originators and responders are together referred to as "end" entities. Each other control plane entity on the tree is an "intermediate" entity.

"Uptree" is used in the present document to indicate the direction from originator to responder, and "downtree" for the direction from responder to originator.

5.2.3 Message classes

Message classes are specified in Table 5.2.1. Each transaction type is specified to use message classes 0 to *n* inclusive, where $0 \leq n \leq 3$.

Table 5.2.1: Message classes

Class	Name	Direction (see note)
0	request	uptree
1	response	downtree
2	confirmation	uptree
3	completion	downtree
NOTE: Acknowledgements are transmitted in the opposite direction		

A transaction shall be initiated, and the tree formed, by the originator sending a request message to one or more of its neighbours; each recipient of the message shall either take on the responder role or forward a (possibly modified) copy of the request message to one or more of its neighbours.

Messages are also identified as "interim" or "final". The specification of each transaction type includes whether interim messages can be used for each class, and if so how they can be distinguished from final messages.

EXAMPLE: If an intermediate entity has passed a FindRoute request onto two of its neighbours, it can forward the first response it receives downtree as an interim message. The originator can then choose whether to accept the response or wait until it has heard from the other responder.

A "normal" reply to a message of class *n* ($0 \leq n \leq 2$) is a message of class n+1 that is part of the same transaction. Such a reply shall not be sent unless a reply message is specified in the subclause of clause 6 appropriate to the message type.

A Terminate request message also acts as a reply to any message which is part of a transaction that it requests to be abandoned.

5.3 Transaction process

5.3.1 General

In the most general case, the tree has more than one branch so there are multiple responders and each responder sends a response message. The originator chooses one of them and sends it a confirmation message; if required, the responder then sends a completion message. The transaction is completed when the last message is acknowledged. The other responders are sent an indication that they should abandon the transaction; this will usually be a Terminate request message.

On receiving a signalling message an entity shall first consider whether the message is a repetition of one it has already processed; if so, it shall take no other action other than to send an acknowledgement.

NOTE: Terminate messages are actioned immediately and then forgotten, so a repetition will not be recognized as such and the repeated message will be processed, see clause 5.3.6.

Otherwise it shall consider whether the message is acceptable; if not, it shall (depending on the detail of the transaction) either reply with a message to abandon the transaction or silently ignore the incoming message. Unacceptable messages include a request message for a transaction that already exists, and a message of any other class for a transaction that the recipient does not recognize or for which a different class is expected.

EXAMPLE: A request for an existing FindRoute transaction is a repetition if received from the same neighbour as the original. Otherwise it indicates that either the request has gone round a loop or there is more than one route from the sender, in which case the second request can be rejected.

Otherwise it shall process the message and take the action specified in the present document, including in clauses 5.3.3, 5.3.4, 5.3.5, 5.3.6 and 6. It shall also send an acknowledgement (except where otherwise specified in clause 5.3.6) unless it sends a reply immediately; it may send an acknowledgement in any case.

5.3.2 Timeouts

On transmission of any message, a timer Ta shall be started and on expiry the message shall be repeated and the timer restarted.

On reception of an acknowledgement or reply to the message, the timer shall be stopped and the message can be discarded.

Loss of the link to the neighbouring control plane entity shall be treated as a request to abandon the transaction.

NOTE 1: In the case of an uptree link where the entity has other uptree links, the request applies only to that branch, not to the whole transaction.

The time until expiry is not specified in the present document. It is an implementation detail which may depend on the nature of the entities and of the link between them, the message type, and other factors.

Likewise, an implementation may set a maximum number of repetitions, after which it takes some action such as testing the link or simply assuming it (or the neighbour) has failed.

NOTE 2: No timeout for completion of a transaction is specified; cases where a transaction stalls are expected to be rare, for instance a software error in a control plane entity that only affects a particular message type. Timeouts can be defined by an implementer, perhaps initiating a CheckFlow transaction rather than abandoning the transaction. Some transactions can include interaction with a user (e.g. to accept a voice call or to agree payment), which would need a comparatively long timeout.

5.3.3 Responder

When a control plane entity receives a request message for which it takes the responder role, if the transaction does not use response messages it is complete. Otherwise, the entity shall send a response message in reply, and if the transaction does not use confirmation messages it will be complete when the response is acknowledged.

When a responder receives a confirmation message, if the transaction does not use completion messages it is complete. Otherwise, the entity shall send a completion message in reply, and the transaction will be complete when the completion message is acknowledged.

5.3.4 Intermediate entities

5.3.4.1 Request messages

When a control plane entity receives a request message for which it takes the intermediate role, it shall send a (possibly altered) copy to one or more of its neighbours, but not to the neighbour from which it received it; see also note to clause 6.1.3.1.

If the transaction does not use response messages, it will be complete when all the forwarded messages have been acknowledged.

5.3.4.2 Other messages when there is a single uptree flow

When an intermediate entity which has exactly one uptree flow receives a response message, it shall send a (possibly altered) copy downtree. If the transaction does not use confirmation messages it will be complete when the forwarded message is acknowledged.

When an intermediate entity which has exactly one uptree flow receives a confirmation message, it shall send a (possibly altered) copy uptree. If the transaction does not use completion messages it will be complete when the forwarded message is acknowledged.

When an intermediate entity which has exactly one uptree flow receives a completion message, it shall send a (possibly altered) copy downtree. The transaction will be complete when the forwarded message is acknowledged.

5.3.4.3 Response messages when there are multiple uptree flows

When an intermediate entity has sent, on two or more signalling flows, request messages for a transaction that includes response messages, and independently of whether they have been acknowledged, each of those flows is in the "expecting a final response" state for the transaction; in that state response messages (both interim and final, as specified in clause 5.2.3) are "acceptable" (in the sense of clause 5.3.1) and the most recently received such message shall be stored. When a final response message is received, the flow shall transition to "final response received" state, in which no messages are acceptable, for the transaction.

NOTE 1: The state applies separately for each transaction. It also applies separately to each signalling flow, even if two or more flows connect to the same neighbour.

When the last final response is received (i.e. when the last signalling flow transitions to "final response received" state) the information in the stored messages shall be used to compose a final response message which shall be sent downtree. Any uptree flows that will not be referenced in the downtree final response message shall be removed from the transaction, including sending a message to abandon the transaction or to clear any state it has set up, as appropriate. This may be done without waiting for the last final response to be received.

NOTE 2: In the case of a FindRoute transaction, the message will be a Terminate request for all routes and/or flows that are set up by the FindRoute; sending it is not conditional on whether confirmation messages are used, nor on whether the flows will have been set up in the forwarding plane by that stage.

An interim response, composed from the information in the stored messages, may be sent downtree at any time before the last final response is received, provided the previous such response (if any) did not contain identical information.

5.3.4.4 Tree pruning

When an intermediate entity receives a message requesting abandonment of a transaction, the flow on which the message arrived shall no longer participate in the transaction. The entity shall end its participation in the transaction unless the message arrived on an uptree flow and there is at least one other uptree flow.

EXAMPLE: An entity has passed an incoming FindRoute request to two neighbours. On receiving a Terminate reply from one of those neighbours, it removes that neighbour from the tree. On receiving a Terminate reply from the second neighbour, it sends a Terminate message downtree to indicate that it does not have a route to the destination and deletes any state it was keeping for the transaction or the route, including in the forwarding plane.

5.3.4.5 Confirmation messages when there are multiple uptree flows

When an intermediate entity that has more than one uptree flow receives a confirmation message, it shall send a (possibly altered) copy towards each responder identified in the message. It shall also send a message to abandon the transaction on any uptree flow on which it does not send a confirmation. If the transaction does not use completion messages it will be complete when all the uptree messages have been acknowledged).

EXAMPLE: A typical situation is a FindRoute transaction where the call has been offered to multiple units, some of which have offered to accept it, and more than one of the offers has been included in the message that was sent downtree. The originator then sends a confirmation message accepting one of the offers and by implication refusing the others.

5.3.4.6 Completion messages when there are multiple uptree flows

The case where a completion message is expected from more than one neighbour is for further study.

5.3.5 Originator

5.3.5.1 Request messages

The originator initiates the transaction by sending one or more request messages as specified in clause 5.2.3. If the transaction does not use response messages, it will be complete when the request messages have been acknowledged.

Any incoming request message will be rejected as specified in clause 5.3.1.

5.3.5.2 Other messages when there is a single uptree flow

When an originator which has exactly one uptree flow receives a response message, if the transaction does not use confirmation messages it will be complete when the message has been processed. Otherwise a confirmation message shall be sent, and if the transaction does not use completion messages it will be complete when the confirmation message is acknowledged.

When an originator which has exactly one uptree flow receives a completion message, the transaction will be complete when the message has been processed.

5.3.5.3 Response messages when there are multiple uptree flows

When an originator has sent a request message on two or more flows, and independently of whether the individual messages have been acknowledged, each of those flows is in the "expecting a final response" state, in which response messages (both interim and final) are "acceptable" (in the sense of clause 5.3.1) and the most recently received such message shall be stored. When a final response message is received, the flow shall transition to "final response received" state, in which no messages are acceptable.

NOTE 1: The state applies separately for each transaction. It also applies separately to each signalling flow, even if two or more flows connect to the same neighbour.

When the last final response is received (i.e. when the last signalling flow transitions to "final response received" state) the information in the stored messages shall be processed. If the transaction uses confirmation messages, a confirmation message or a message to abandon the transaction (as appropriate) shall be sent on each uptree flow. Otherwise, a message to clear any state it has set up shall be sent to any neighbour for which it is appropriate.

NOTE 2: In the case of a FindRoute transaction, a Terminate for any routes and/or flows that are set up by the FindRoute is sent to any neighbour that was sent the FindRoute request but for which the flows will not be connected in the forwarding plane.

The information in the stored messages may also be processed at any time before the last flow transitions to "final response received", and action taken as if all final responses had been received.

EXAMPLE: The originator has sent a FindRoute request to two neighbours. One replies with a route that has the required QoS. The originator sets the flow up to that neighbour without waiting for the response from the second neighbour. It also sends a Terminate to the second neighbour.

5.3.5.4 Tree pruning

When the originator receives a message requesting abandonment of a transaction, the flow on which the message arrived shall no longer participate in the transaction. If there are no other uptree flows, the transaction terminates.

5.3.5.5 Completion messages when there are multiple uptree flows

The case where a completion message is expected from more than one neighbour is for further study.

5.3.6 Abandoning transactions

A control plane entity may request abandonment of a transaction or of its participation in the transaction, by sending a Terminate request message.

The Terminate transaction uses only one message class and is restricted to a single link between control plane entities, so there are no intermediate entities, only the originator and responder. It requests removal of resources, identified by variable IEs in the message, associated with the signalling flow over which it is sent and/or any corresponding forwarding plane links. One or more Cause IEs should be used to indicate the reason for requesting termination.

If the serial number in the message is zero, the responder is not required to send an acknowledgement.

NOTE 1: This is intended for the case where the message acts as a negative acknowledgement for a message the sender has received, and for which the sender has not kept any state.

If the message contains a MessageType IE, it acts as a negative reply to a message of the indicated type, and reports an error such as that the responder does not recognize (or does not implement) the transaction type or the responder's record of the state of the transaction indicates that the message class is inappropriate. Additional IEs may be included if necessary to identify the specific transaction.

If the message contains one or more Route IEs it requests clearing down of routes and/or user flows as specified in clause 6.2, as well as abandonment of any transactions that refer to them.

NOTE 2: Terminate messages with Route IEs are used both for clearing down existing user flows and for rejecting attempts by FindRoute transactions to set new user flows up.

In all cases, the recipient shall silently ignore any part of a Terminate request message for which it cannot identify the resource referred to.

EXAMPLE: A request to clear down a user flow that has already been cleared down is simply ignored.

5.4 Transaction types

5.4.1 Summary

Transaction type codes are specified in Table 5.4.1, which also lists the clause in which the detail of each transaction type is specified.

Table 5.4.1: Transaction types

Code	Transaction	Clause
0-4	reserved for link-specific messages	5.4.2
5	reserved for LinkInfo	5.4.2
6	FrameSyncInfo	7.3
7	TimeSyncInfo	8.2
8	FindRoute	6.1
9	Terminate	5.3.6, 6.2
10	AddFlow	6.3
11	NetworkData	6.4.2
12	EndToEndData	6.4.3
13	ChangeAllocation	6.4.4
14	CheckFlow	6.4.5
15-28	reserved	
29	reserved for long messages	5.5.5
30	EnterpriseSpecific	5.5.3
31	ExtendedType	5.5.3

5.4.2 Link-specific transactions

Transaction types 0-4 are reserved for carrying messages that are specific to a particular type of link, on links of that type, and are not specified in the present document.

Transaction type 5 is reserved for exchanging configuration information between the entities at either end of a link, in circumstances where there is no opportunity to negotiate before switching to Flexilink format. The formats and protocols are for further study.

5.5 Data unit format

5.5.1 General

Except where clause 5.5.5 applies, each signalling data unit shall be the payload of a basic service packet and each packet on a signalling flow shall carry a single signalling data unit.

The specifications in the present document define the "abstract" format specified in clause 5.5.2. The format transmitted shall use one of the codings specified in later subclauses of clause 5.5. The coding to be used on each signalling flow shall be signalled when the flow, or the link that carries it, is set up.

5.5.2 Abstract format

Each data unit shall consist of its type followed by a "fixed part", optionally followed by a "variable part".

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| A | class |  transaction type |
+---+---+---+---+---+---+---+---+
```

Figure 5.5.1: Structure of a signalling data unit type

The data unit type shall be a single octet partitioned as follows (and as illustrated in Figure 5.5.1):

bit 0:	"acknowledgement" flag: 0 = message, 1 = acknowledgement
bits 1-2:	message class, as specified in Table 5.2.1
bits 3-7:	transaction type code, as specified in Table 5.4.1

The fixed and variable parts shall each consist of "Information Elements", also called "IEs".

Bits 1 to 7 inclusive of the data unit type are the "message type".

For each transaction type, the IEs that shall be present in the fixed part are specified in Table 5.5.1 and the IEs that shall (or may) be present in the variable part are specified (separately for each message class) in the clause shown in Table 5.4.1.

Each type of "fixed" IE shall consist of a defined number of octets, as specified in Table 5.6.1.

Each "variable" IE shall consist of its type followed by a "fixed part" followed by a "variable part".

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| V |          IE type          |
+---+---+---+---+---+---+---+---+
```

Figure 5.5.2: Structure of the type of a variable IE

The IE type shall be a single octet partitioned as follows (and as illustrated in Figure 5.5.2):

bit 0:	0 = variable part is empty, 1 = variable part contains one or more variable IEs
bits 1-7:	IE type code, as specified in Table 5.7.1

The fixed part shall consist of a defined sequence of zero of more fixed IEs.

The variable part shall consist of zero or more variable IEs, referred to as "nested" IEs.

For each IE type, the IEs that shall be present in the fixed part are specified in the clause shown in Table 5.7.1.

NOTE: Fixed IEs are identified by their location in the fixed part of the containing message or IE. Variable IEs are identified by their type. Unless otherwise specified (e.g. in clause 5.7.19) variable IEs may appear in any order.

The IEs that form the variable part of a data unit or IE are referred to in the present document as being "directly contained" in it. An IE is "contained" in a data unit or IE if it is directly contained in it or in another IE which in turn is contained in it. "Indirectly contained" means contained but not directly contained.

Wherever a particular variable IE type is required, there may instead be an Alternatives IE containing IEs of the required type, directly or in Group IEs. There may be more than one Alternatives IE, in which case they are separate sets of alternatives to be applied orthogonally (for instance, one for the called address and another for the data format and flow descriptors). When the message is passed on, any alternatives that cannot be supported (for instance, needing more capacity than is available on the link) shall be omitted.

5.5.3 Message fixed part

The fixed part of a message shall depend on the transaction type, as shown in Table 5.5.1.

NOTE: IE formats are listed in Table 5.6.1.

Table 5.5.1: Fixed part of message

Transaction	First IE in fixed part	Further IEs in fixed part
FrameSyncInfo	SyncSerial IE	see note 1
TimeSyncInfo	SyncSerial IE, Time IE, TimingSource IE	see note 2
FindRoute	RouteId IE	none
Terminate	SerialNumber IE	none
AddFlow	RouteId IE	none
NetworkData	RouteId IE	none
EndToEndData	RouteId IE	none
ChangeAllocation	RouteId IE	none
CheckFlow	RouteId IE	none
EnterpriseSpecific	RelativeOid IE	specified by owner of enterprise number
ExtendedType	AbsoluteOid IE	specified in standard identified by the OID
NOTE 1: If the subtype in the SyncSerial IE is coded as "handover" there shall be no further IEs. If it is coded as "upstream" or "downstream" the SyncSerial IE shall be followed by a TimingSource IE which may in turn be followed by one or more TimingRelay IEs. NOTE 2: The TimingSource IE may be followed by one or more TimingRelay IEs.		

EnterpriseSpecific messages shall be used for proprietary and experimental transaction types. The syntax and semantics of the remainder of the message (following the RelativeOid IE), for each message type, shall be specified by the owner of the enterprise number which is the first arc of the RelativeOid.

ExtendedType messages shall be reserved for use in future standards. The syntax and semantics of the remainder of the message (following the AbsoluteOid IE), for each message type, shall be specified in the standard identified by the AbsoluteOid.

EXAMPLE: An OID beginning 0.4.0.*n* would identify a transaction specified in an ETSI standard with *n* in the last five digits of its number. An OID beginning 1.0.*n* would identify a transaction specified in ISO or IEC standard number *n*.

5.5.4 Acknowledgement

The fixed and variable parts of an acknowledgement shall depend on the transaction type, as shown in Table 5.5.2.

NOTE 1: IE formats are listed in Table 5.6.1.

Table 5.5.2: Fixed and variable parts of acknowledgement

Transaction	Fixed part	Variable part
FrameSyncInfo	SyncSerial IE	empty
TimeSyncInfo	SyncSerial IE	empty
FindRoute	RouteId IE	InterimOffer IEs
Terminate	SerialNumber IE	empty
AddFlow	RouteId IE	FlowDescriptor IE (omitting variable part)
NetworkData	RouteId IE	empty
EndToEndData	RouteId IE	empty
ChangeAllocation	RouteId IE	FlowDescriptor IE (omitting variable part)
CheckFlow	RouteId IE	FlowDescriptor IE (omitting variable part)
EnterpriseSpecific	RelativeOid IE; see notes 1 and 3	see notes 1 and 3
ExtendedType	AbsoluteOid IE; see notes 2 and 3	see notes 2 and 3
NOTE 1: Any further IEs to be copied from the message shall be specified by the owner of the enterprise number which is the first arc of the RelativeOid. NOTE 2: Any further IEs to be copied from the message shall be specified in the standard identified by the AbsoluteOid. NOTE 3: Acknowledgements should contain only the IEs that are necessary to identify the message that is being acknowledged.		

The variable part shall contain only those IEs that are listed in Table 5.5.2 and are present in the message that is being acknowledged.

In the abstract format, the message type and each IE shall have identical values to the corresponding fields in the message that is being acknowledged, except that where an IE is annotated "omitting variable part" its bit 0 (the V flag) shall be zero and its variable part shall be empty.

NOTE 2: Further changes can be needed when clearing the V flag, depending on which coding is used (see clauses 5.5.6 and 5.5.7).

5.5.5 Long messages

Fragmentation of data units that do not fit into a basic service packet is for further study. The first octet of each fragment shall contain a data unit type as illustrated in Figure 5.5.1, coded with transaction type 29.

5.5.6 Transparent coding

Transparent coding shall convey the fixed part of each data unit or IE as the sequence of octets defined as its abstract format.

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| A | class |  transaction type |      length of fixed part     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.5.3: Data unit header

The first octet of a data unit shall be its type and the second shall be the number of octets in its fixed part, as illustrated in Figure 5.5.3. The fixed part shall begin in the third octet, and shall be immediately followed by the variable part.

```
  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| 0|      IE type       |             length of fixed part              |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 5.5.4: Header of a variable IE with no variable part

```
 0 1          7 8                            23 24          31
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IE type   |   number of further octets    |fix part length|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.5.5: Header of a variable IE which has a variable part

The first octet of a variable IE shall be its type. If it has no variable part (i.e. when V=0), the second and third shall be the number of octets in its fixed part (coded as a 16 bit unsigned integer), as illustrated in Figure 5.5.4, and the fixed part shall begin in the fourth octet.

A variable IE which has a variable part (i.e. when V=1) shall be coded with the second and third octets containing one more than the total number of octets in its fixed and variable parts (coded as a 16 bit unsigned integer) and the number of octets in the fixed part in the fourth octet, as illustrated in Figure 5.5.5. The fixed part shall begin in the fifth octet, and shall be immediately followed by the variable part.

NOTE: In either case, bits 8 to 23 of a variable IE code the number of octets in the remainder of the IE.

5.5.7 ASN.1 coding

The use of Packed Encoding Rules in a similar way to ETSI TS 136 413 [i.8] (S1 Application Protocol) is for further study. It will require an Annex containing the machine readable version.

5.6 Syntax of fixed IEs

5.6.1 Summary

Fixed information element types are listed in Table 5.6.1.

Table 5.6.1: Fixed information element types

Name	Length	Contents	Clause
RouteId	13 octets	Route identifier with D=0	5.6.2
SerialNumber	3 octets	Identification of a transaction	5.6.3
RelativeOid	see note to clause 5.6.4	Relative OID (ASN.1 basic encoding rules)	5.6.4
AbsoluteOid	see note to clause 5.6.4	Absolute OID (ASN.1 basic encoding rules)	5.6.4
SyncSerial	2 octets	Subtype and serial number for SyncInfo	5.6.5
Time	9 octets	Information about network time	5.6.6
TimingSource	4 octets	Reference for frame sync or timing	5.6.7
TimingRelay	12 octets	Local source of frame sync or timing	5.6.8

NOTE: Fixed IEs do not include a header; the type and length are determined by the syntax of the containing object.

5.6.2 Route identifier

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                             owner                             +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        call reference                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  route ref  |0|
+-+-+-+-+-+-+-+-+                                               .
```

Figure 5.6.1: RouteIdentifier IE

A RouteId IE shall consist of 103 bits containing the route identifier (see clause 4.3.3) for the route, followed by a single zero bit, as illustrated in Figure 5.6.1.

5.6.3 Serial number

A SerialNumber IE shall consist of 3 octets containing a serial number chosen by the originator of the transaction. The recipient shall not attribute any significance to its value, except as specified in clause 5.3.6.

NOTE: The only purpose served by the serial number is to distinguish between different transactions. The recipient cannot, for instance, assume that a gap in serial number values means that a message has been lost.

5.6.4 OIDs

A RelativeOid shall consist of a relative OID rooted at 1.3.6.1.4.1, coded in the same way as in ASN.1 Basic Encoding Rules (BER), but omitting the tag and length.

An AbsoluteOid shall consist of an OID coded in the same way as in ASN.1 Basic Encoding Rules (BER), including the compressed form for the first two arcs, but omitting the tag and length.

NOTE: There is no explicit specification of the length of the OID in either case. When used in an EnterpriseSpecific or ExtendedType transaction or an ExtendRel or ExendAbs IE, the recipient of a message can match each OID it supports against the start of the fixed IE. The only way this could be ambiguous is if the owner of an OID has allocated two message types, both descended from that OID, such that one is a subset of the other. In other cases, such as address types 1 to 3, the length is signalled separately.

5.6.5 Serial number for SyncInfo

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|subtype|                   serial number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.6.2: SyncSerial IE

A SyncSerial IE shall consist of two octets coded as illustrated in Figure 5.6.2.

Bits 0 and 1 shall contain a subtype value coded as:

- 1 = downstream;
- 2 = upstream;
- 3 = handover;
- code point 0 is reserved.

Bits 2 to 15 inclusive shall contain a serial number chosen by the originator of the transaction. The recipient shall not attribute any significance to its value.

NOTE: The only purpose served by the serial number is to distinguish between different transactions. The recipient cannot, for instance, assume that a gap in serial number values means that a message has been lost.

5.6.6 Information about network time

```
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |sec|                      nanoseconds                          |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                            seconds                            |
       +               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |               |
       +-+-+-+-+-+-+-+-+                                               .
```

Figure 5.6.3: Time IE

A Time IE shall consist of 9 octets, as illustrated in Figure 5.6.3.

The first 4 octets shall contain the time in Flexilink frames received from the link partner relative to the sender's network time, in the same format as the 32-bit "timing" field in Flexilink frames (see clause 8.1.3), or all-ones if the information is not available.

NOTE 1: This allows the latency on the link to be estimated, see clauses 8.1.4 and 8.2.

The other 5 octets shall contain the integer part of the sender's network time when the message was constructed, or all-ones if the information is not available. The least significant 2 bits correlate with the bits 0 and 1 of the timing field of Flexilink frames; when setting the recipient's network time the value should be incremented if necessary so that the those two bits match.

NOTE 2: Assuming the transmission and software processing delays are less than two seconds, the recipient is able to set its network time equal to the sender's within a tolerance that depends on how symmetrical the delays on the link are; see also clause 8.2.

5.6.7 Source of timing

```
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | timing source |        uncertainty relative to source         |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.6.4: TimingSource IE

A TimingSource IE shall consist of 4 octets (32 bits) as illustrated in Figure 5.6.4, and shall define the timing in (or of) frames transmitted by a unit.

NOTE: The semantics, including the definition of "uncertainty", are further specified in clauses 7.2 and 7.3 for frame alignment and clauses 8.2.2 and 8.2.3 for network time.

Bits 0 to 7 inclusive shall describe the source of the timing, coded as a clockClass value from AES67 [16], Table A.2.

Bits 8 to 31 inclusive shall contain the total number of nanoseconds of uncertainty between the source of the timing and the transfer of frames transmitted by the unit to the physical layer.

Clock classes that are shown in AES67 [16], Table A.2 as using the PTP timescale shall indicate that the source is TAI.

If the source is internal to the unit, with no external reference, including the case where inter-frame gaps are being adjusted to match a previous rate (see clauses 7.2.2.4 and 7.3.6), bits 0 to 7 shall be coded with the value 248 (decimal).

5.6.8 Unit relaying timing

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                        unit identifier                        +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           uncertainty relative to local reference             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.6.5: TimingRelay IE

TimingRelay IEs shall only occur immediately following either a TimingSource IE or another TimingRelay IE.

A TimingRelay IE shall consist of 12 octets (96 bits) as illustrated in Figure 5.6.5, and shall define the timing in (or of) frames transmitted by a unit.

NOTE 1: The semantics, including the definition of "uncertainty", are further specified in clauses 7.2 and 7.3 for frame alignment and clause 8.2.3 for network time.

Bits 0 to 63 inclusive shall contain the identifier of the unit whose timing is defined by the preceding IE.

Bits 64 to 95 inclusive shall contain the total number of nanoseconds of uncertainty between frames transmitted by the unit whose identifier is in bits 0 to 63 and those transmitted by the unit whose timing is defined by the TimingRelay IE.

NOTE 2: The uncertainty relative to the source of the timing is therefore the sum of the uncertainty values in the TimingRelay IE, the TimingSource IE, and any intervening TimingRelay IEs.

5.7 Syntax of variable IEs

5.7.1 Summary

Variable information element types are listed in Table 5.7.1.

The coding of each IE type is specified in the clause indicated in Table 5.7.1.

Table 5.7.1: Variable information element types

Code	IE type	Clause
0-2	Reserved	
3	CalledAddress	5.7.2
4	FlowDescriptor	5.7.3
5	DataType (format or protocol)	5.7.4
6	StartTime	5.7.5
7	EndTime	5.7.5
8	Importance	5.7.6
9	ServiceName (includes programme name)	5.7.7
10	SourceName	5.7.7
11	DestinationName	5.7.7
12	Reserved	
13	Password	5.7.8
14	Charge (for call)	5.7.9
15	ClientIdentity	5.7.10
16	RouteMetric	5.7.11
17	Bandwidth	5.7.12
18	TrafficParams (for basic service)	5.7.13
19	SlotAlloc (link-specific resource allocation for guaranteed service)	5.7.14
20	BasicAlloc (link-specific resource allocation for basic service)	5.7.14
21	Delay	5.7.15
22	McastRoute (route identifier of multicast)	5.7.16
23	Cause	5.7.17
24	Route	5.7.18
25	Alternatives	5.7.19
26	Group	5.7.20
27	InterimOffer	5.7.21
28	DestCount (number of destinations)	5.7.22
29	UserData	5.7.23
30	MessageType	5.7.24
31 to 63	Reserved	
64 to 71	Reserved for link-specific resource management	
72 to 125	Reserved	
126	ExtendRel (extended IE types with relative OID rooted at 1.3.6.1.4.1)	5.7.25
127	ExtendAbs (extended IE types with absolute OID)	5.7.25

5.7.2 Called address

The fixed part of a CalledAddress IE shall contain the address of the called party, in the form specified in clause 4.4.

The variable part shall be empty.

5.7.3 Flow descriptor

```
 0 1 2     5 6 7 8                                           31
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|G|T| resvd |A|T|               flow reference                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.7.1: Fixed part of FlowDescriptor IE

The fixed part of a FlowDescriptor IE shall consist of four octets as illustrated in Figure 5.7.1.

Bit 0 shall be 1 for a guaranteed service flow, 0 for a basic service flow

Bit 1 shall be 1 if the flow should only be connected across tightly synchronized links, 0 if it may also be connected across loosely synchronized links.

Bits 2 to 5 inclusive are reserved, and shall be coded as zero by the sender and ignored by the recipient.

Bits 6 and 7 shall indicate the direction as:

- 1 = towards the owner;
- 2 = away from the owner;
- 3 = bidirectional pair of flows;
- 0 reserved.

The other three octets shall contain a flow reference as specified in clause 4.3, except that zero shall indicate that the flow reference is unspecified. Unlike the case where zero indicates "all flows", the "direction" coding in bits 6 and 7 shall still be valid when the flow reference is coded as zero.

The direction and flow reference shall be combined with a route identifier specified elsewhere in the message, to form the flow identifier(s) for the flow or pair of flows to which the IEs in the variable part, and the flags in bits 0 and 1, apply.

The variable part may contain IEs that give further information about the flow, as specified in clause 6.

5.7.4 Data format or protocol

The fixed part of a DataType IE shall consist of an AbsoluteOid that specifies a property of the call, route, or flow to which it applies.

The variable part shall be empty.

EXAMPLE: The variable part of a FlowDescriptor IE for a flow carrying audio data can include a DataType IE specifying the number of audio channels, another specifying the audio coding, and another specifying the encapsulation on the network.

5.7.5 Start and end time

The fixed part of a StartTime IE shall consist of 9 octets coded with the time by which the route or flow is to be connected, expressed as the number of seconds and nanoseconds since 1970-01-01T00:00:00TAI.

The fixed part of an EndTime IE shall consist of 9 octets coded with the time after which the route or flow may be released, expressed as the number of seconds and nanoseconds since 1970-01-01T00:00:00TAI.

The number of seconds shall be coded as an unsigned integer in the first 5 octets, and the number of nanoseconds as an unsigned integer with value less than 1 000 000 000 in the remaining four octets.

The variable part shall be empty.

5.7.6 Importance

The fixed part of an Importance IE shall consist of 1 octet coded with an "importance" value which shall be an 8-bit unsigned integer with zero being the least important.

The variable part shall be empty.

NOTE: Each flow has an importance value (default zero) which can be used for various purposes including prioritizing re-routing around a fault. Use of specific non-zero values is for further study.

5.7.7 Names

The fixed part of a ServiceName IE shall consist of a name or other identifier for the content carried by a flow, coded as a UTF-8 octet string as specified in The Unicode® Standard [14], clause 2.5.

The fixed part of a SourceName IE shall consist of a name or other identifier for the source of the content carried by a flow, coded as a UTF-8 octet string as specified in The Unicode® Standard [14], clause 2.5.

The fixed part of a DestinationName IE shall consist of a name or other identifier for the destination of the content carried by a flow, coded as a UTF-8 octet string as specified in The Unicode® Standard [14], clause 2.5.

In each case, NUL characters at the end of the string are not required but shall be ignored if present.

The variable part shall be empty.

NOTE: These IEs are intended for applications such as broadcast studios where the ServiceName can be used to identify a broadcast channel and source and destination can be studios, transmitters, etc.

5.7.8 Password

The content of a Password IE shall provide identification and/or authentication of the originator of a FindRoute transaction. The format shall be as required by the intended destination of the call and is thus application-specific and not specified in the present document. It shall not include information that is needed to identify the branch of a multicast flow that terminates at the originator.

NOTE 1: Intended uses include providing identification of a user wishing to consume content or access a service. In the case of a point-to-point bidirectional call from a client to a server, the server can simply send replies on the call on which the request was received, without needing to know the client's address.

NOTE 2: Control plane entities do not need to store the information in the Password IE; see also clause 5.7.10.

5.7.9 Charge for call

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|context|       reserved        |
+---+---+---+---+---+---+---+---+
```

Figure 5.7.2: First octet of the fixed part of a Charge IE

The fixed part of a Charge IE shall consist of one octet coded as illustrated in Figure 5.7.2 followed by an AbsoluteOid followed by further information whose syntax and semantics are indicated by the AbsoluteOid. IEs permitted or required in the variable part and their interpretation shall also be indicated by the AbsoluteOid.

The first two bits shall be coded as 00 if the IE indicates the maximum the originator of the IE is willing to pay, 01 if the IE indicates a charge which the originator of the IE wishes to levy, or 10 if the IE indicates a charge which the originator of the IE accepts and will pay. The code point 11 is reserved and shall not be used.

NOTE: These codings were chosen to match the message class (see clause 5.2).

The remainder of the first octet is reserved and should be coded as zero by the sender of the message and ignored by the recipient.

EXAMPLE 1: An OID might be defined which indicates a charge which is to be paid by the caller to the caller's ISP. In this case the IE would be originated by the caller (and removed by the ISP) in request and confirmation messages, and originated by the ISP in response messages.

EXAMPLE 2: An OID might be defined which indicates that the charge is to be paid to the called party (as content owner) by a rights management organization identified by an IE in the variable part.

5.7.10 Caller's identity

The content of a ClientIdentity IE shall provide identification of the originator of a FindRoute transaction, including information that is needed to identify the branch of a multicast flow that terminates at the originator. The format shall be as required by the intended destination of the call and is thus application-specific and not specified in the present document.

For information that is not needed to identify the branch of a multicast flow that terminates at the originator, the Password IE (see clause 5.7.8) should be used instead.

NOTE 1: Intended uses include supplying the source address when setting up a tunnel to carry Ethernet, IP, etc., packets, and identifying the consumer of a multicast live media flow.

NOTE 2: The information contained in the ClientIdentity IE is part of the state of the route or flow and the sender of a multicast can use it to disconnect an individual destination if required; see clauses 6.2.3 and 6.4.3.

5.7.11 Route metric

```
  0   1   2   3   4   5   6   7 | 8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| status|                   number of links                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 5.7.3: First two octets of the fixed part of a RouteMetric IE

The first two bits of the fixed part of a RouteMetric IE shall contain its status, as illustrated in Figure 5.7.3.

A status value of 0 shall indicate that the value for the direction in which the message passes is being accumulated, and the value coded is the accumulated value up to and including the link over which the message is transmitted.

A status value of 1 shall indicate the same as status value 0 and additionally that the end-to-end value should be reported back to the originator of the message.

A status value of 2 shall indicate that the value for the direction opposite to that in which the message passes is being accumulated, and the value coded is the accumulated value starting with the link over which the message is transmitted.

A status value of 3 shall indicate that the IE reports the end-to-end value for the opposite direction.

The next 14 bits shall be coded with an unsigned integer which is the number of links over which the route passes.

NOTE 1: The coding of the remainder of the IE, including a measure of the proportion of available space required on each link or on the smallest-capacity or most congested link, is for further study.

Inclusion of information beyond the first two octets shall be optional. Equipment designed before the coding of this additional information is defined may continue to send two-octet RouteMetric IEs and should ignore anything beyond the first two octets in incoming IEs.

NOTE 2: This IE does not report latency, which is reported by the Delay IE (see clause 5.7.15).

5.7.12 Bandwidth

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        slots per second                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.7.4: Fixed part of Bandwidth IE

The fixed part of a Bandwidth IE shall consist of four octets containing an unsigned 32-bit integer, as illustrated in Figure 5.7.4, containing the number of slots per second required for a guaranteed service flow.

The variable part shall be empty.

NOTE: This IE is not needed if a SlotAlloc IE is present.

5.7.13 Basic service traffic parameters

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         mean data rate                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       maximum data rate                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                         total octets                          |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.7.5: Fixed part of TrafficParams IE

The fixed part of a TrafficParams IE shall consist of one to three unsigned integers, as illustrated in Figure 5.7.5. Bits 0 to 31 shall code the mean data rate and bits 32 to 63 shall code the maximum data rate, each in units of 256 octets (2 048 bits) per second. Bits 64 to 127 shall code the total number of octets to be transmitted.

In each case zero shall indicate "unspecified" and all-ones shall indicate "more than the largest value that can be coded". Where there are fewer than three integers, the remaining values shall be interpreted to be unspecified.

The variable part shall be empty.

NOTE: The basic service parameters are intended to give an estimate of the load that the flow will place on the network and on the recipient, to aid load balancing, but not to form a basis for QoS guarantees. The parameters for the direction towards the sender of a message indicate the maximum load expected from incoming traffic.

5.7.14 Link-specific resource allocations

5.7.14.1 General

The detail of the format and interpretation of SlotAlloc and BasicAlloc IEs shall be specific to the networking technology used on the link over which the message passes.

However, the format should where appropriate conform to the descriptions in the following subclauses of clause 5.7.14.

NOTE: These IE types are intended for information related to the allocation of forwarding plane resources on the link. See clause A.4 for examples.

5.7.14.2 Guaranteed service on tightly synchronized links

The SlotAlloc IE on tightly synchronized links should include a specification of the slots allocated to the flow, expressed as the location within the allocation period of one of the slots (the "anchor" slot) and a table showing the gap (i.e. the number of slots not allocated to the flow) between each allocated slot and the next, beginning with the anchor slot.

The table should be preceded by the following three integers:

- location of the anchor slot, expressed as the number of slots preceding it in the allocation period;
- the number of gaps (which is one less than the number of slots); and
- *e*, the number of bits in each entry in the table.

The table should consist of a sequence of "entries", each of which is an integer in the range 0 to *n* inclusive, where $n = 2^e – 1$. A gap of $(k * n) + j$ slots, where $j < n$, should be represented as *k* entries coded with the value *n* followed by one entry coded with the value *j*.

NOTE 1: The total number of entries is not signalled explicitly. The total length of the table can be minimized by choosing *e* such that for most of the gaps $k = 0$ and $j > n / 2$. This caters both for allocations consisting of a small number of slots with large gaps between them and also for those consisting of a large number of slots with small gaps between them.

NOTE 2: If *e* = 1 the table becomes a bit map with a 0 for each slot that is allocated to the flow and a 1 for each slot that is not.

5.7.14.3 Guaranteed service on loosely synchronized links

On loosely synchronized links, guaranteed service packets shall be encapsulated in basic service packets as specified in ETSI GS NGP 013 [1], clause 7.1.

NOTE: Typically the basic service flow is connected by a BasicAlloc IE in the FindRoute request message and is thus available for the guaranteed service flow to be connected by a SlotAlloc IE in the FindRoute response message.

The SlotAlloc IE on loosely synchronized links should begin with four octets containing the number of slots per second, as in the Bandwidth IE (see clause 5.7.12).

Further information in the SlotAlloc IE, possibly including a specification of the number of basic service packets to be transmitted per allocation period (or, equivalently, the number of slots in each basic service packet), is for further study.

5.7.14.4 Basic service

The BasicAlloc IE should include the label to be used on packets transmitted on the link in the direction towards the sender of the message, including any protection bits.

5.7.15 End-to-end delay

A Delay IE shall code the upper and lower bounds to the delay which will be experienced by data units transmitted on the flow if it is directly contained in a FlowDescriptor IE, or on any guaranteed service flows along the route if it is directly contained in a message. In the latter case, the reported figure should cover all possible guaranteed service flows, not just flows that exist or are being connected at the time the delay is reported.

NOTE: The Delay IE is intended to be used for guaranteed service flows. Basic service flows can be expected to experience unbounded queuing delays.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| S |                    maximum latency                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        minimum latency                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.7.6: Fixed part of Delay IE

The fixed part of a Delay IE shall consist of either four or eight octets, as illustrated in Figure 5.7.6.

The first two bits shall contain the status, coded in the same way as for a RouteMetric IE (see clause 5.7.11). The remainder of the first four octets shall be coded with a 30-bit unsigned integer which is the upper bound to the delay, in units of sixteen nanoseconds.

If the fixed part is eight octets, the second four shall be coded with a 32-bit unsigned integer which is the lower bound to the delay. If the fixed part is four octets, this shall indicate that the lower bound is unspecified and should be assumed to be zero.

The variable part may be empty. IEs that would give more detail of the service the flow is expected to receive are for further study.

5.7.16 Route identifier of multicast

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                           owner                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        call reference                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  route ref  |   reserved  |A|I|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               .
```

Figure 5.7.7: Fixed part of RouteIdentifier IE

The fixed part of a McastRoute IE shall consist of 14 octets (112 bits) as illustrated in Figure 5.7.7. The first 103 bits shall contain the route identifier assigned by the source. The next 7 bits shall be coded as zero by the sender and ignored by the recipient.

The last two bits shall contain an "openness" value coded as: 00 if the source of the multicast does not wish to be informed when destinations are added or removed; 01 if the source requires to be informed of the total number of destinations; or 11 if addition of a new destination requires the approval of the source. The code point 10 is reserved and shall not be used.

The variable part shall be empty.

NOTE: In Figure 5.7.7, "I" indicates that the source is to be informed and "A" that its approval is needed.

5.7.17 Cause

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| R |   reserved    |  domain   |
+---+---+---+---+---+---+---+---+
```

Figure 5.7.8: First octet of the fixed part of a Cause IE

Cause IEs shall be used to signal the reason for clearing a flow down or for abandoning a transaction. The fixed part shall begin with one octet coded as illustrated in Figure 5.7.8.

Bit 0 shall be coded as 1 if the connection should be retried by another route, 0 otherwise. Bits 1 to 4 shall be coded as zero by the sender and ignored by the recipient.

Bits 5 to 7 shall specify how the remainder of the fixed part is coded. When coded as 0, it shall be followed by an AbsoluteOid indicating the reason. When coded as 1, it shall be followed by a RelativeOid indicating the reason; the root of the RelativeOid is for further study, and code point 1 shall not be used until it is defined. Code points 2 to 7 are reserved. In all cases, the variable part may contain additional information relevant to the indicated condition.

NOTE 1: OIDs for use with code point 0 are for further study. Use of the reserved code points for causes specified in standards such as Recommendation ITU-T Q.850 [i.1] is for further study.

During call set-up, bit 0 should be coded as 1 if the destination could not be reached but might be reachable by another route, as 0 if it was located and refused the call. When clearing a flow down, it should be coded as 1 if the cause was a broken link or other equipment failure, 0 if the remote party disconnected.

NOTE 2: Normal call clearing is indicated by the absence of a Cause IE.

5.7.18 Route and flow selection

The fixed part of a Route IE shall consist of a RouteId IE. The variable part may contain one or more FlowDescriptor IEs.

A Route IE with a non-empty variable part shall select the flow(s) indicated by the FlowDescriptor IE(s) contained in it, but not the route itself, even if it does not carry any other flows.

A Route IE with an empty variable part shall select the route and all flows carried on it.

5.7.19 Alternatives

The fixed part of an Alternatives IE shall be of zero length. The variable part shall contain two or more IEs, which shall each be an IE that could replace the Alternatives IE, in decreasing order of preference.

A control plane entity forwarding a message that contains an Alternatives IE shall remove from it any directly contained IEs that cannot be supported. If there is then one IE remaining it shall replace the Alternatives IE with the remaining IE unless it is a Group IE in which case it shall instead replace it with the IEs that are directly contained within the Group IE. If no IEs remain it may abandon the transaction.

EXAMPLE 1: An Alternatives IE in a FindRoute message can contain several Group IEs each of which contains a Data IE specifying video compression parameters and a Bandwidth IE specifying the data rate needed to convey the compressed data, listed in order from the highest to the lowest bandwidth, and hence also from the highest to the lowest video quality. The flow will then be connected using the highest quality that can be supported by the available bandwidth.

NOTE: In most cases the IEs directly contained in the Alternatives IE will all be the same IE type (or, in the case of a Group, set of types). However, the possibility that they can be different is not excluded. Also, it is possible for some of them to be single IEs and others to be groups.

EXAMPLE 2: In an Alternatives IE in a FindRoute message one directly contained IE can be a CalledAddress IE and another can be an IE of a different type (or a group of IEs) indicating the intended destination.

5.7.20 Group

The fixed part of a Group IE shall be of zero length. The variable part shall contain two or more IEs. The Group IE shall only occur directly contained in an Alternatives IE.

See example 1 to clause 5.7.19.

5.7.21 Interim offer

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                           host unit                           +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      identifier for offer     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               .
```

Figure 5.7.9: Fixed part of InterimOffer IE

The fixed part of an InterimOffer IE shall consist of 10 octets as illustrated in Figure 5.7.9. The first eight shall contain the identifier of a unit which hosts a control plane entity and the remaining two shall contain a serial number, chosen by the unit, which identifies an "offer" (and the neighbour from which it was received, also the control plane entity itself if the unit hosts more than one) uniquely within the context of the unit and the call for which the offer is made.

The variable part shall be empty.

A FindRoute response message shall be an "interim offer" if it contains at least one InterimOffer IE, a "final offer" if it does not contain any InterimOffer IEs.

5.7.22 Number of destinations

The fixed part of a DestCount IE shall consist of a single unsigned integer whose value is the number of destinations of a multicast flow.

The variable part shall be empty.

NOTE: The length is not specified, and there is no requirement to use the minimum number of octets. In many cases one or two octets will be enough, but for some applications (live broadcast of a sports event, for instance) a larger number will be needed, and the sender can choose to always use four octets. It is possible that for some applications (for instance, distributing global information in ubiquitous sensor networks) more than four octets will be needed.

5.7.23 User data

The fixed part of a UserData IE shall contain an octet string. The variable part shall be empty.

NOTE: Because there is no variable part, transparent coding supports octet strings longer than 255 octets; see also clause 5.5.5.

5.7.24 MessageType

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| T | class |  transaction type |
+---+---+---+---+---+---+---+---+
```

Figure 5.7.10: Fixed part of MessageType IE

The fixed part of a MessageType IE shall consist of one octet partitioned as follows (and as illustrated in Figure 5.7.10):

bit 0:	0 = message type, 1 = transaction type
bits 1-2:	see below
bits 3-7:	transaction type code, as specified in Table 5.4.1

The fixed part codes either a message type or a transaction type, depending on the value in bit 0. When bit 0 is 0, bits 1 and 2 hold the message class, as specified in Table 5.2.1. When bit 0 is 1, bits 1 and 2 shall be coded as zero and shall be ignored by the recipient.

5.7.25 Extended IE types

The fixed part of an ExtendRel IE shall begin with a RelativeOid. The fixed part of an ExtendAbs IE shall begin with an AbsoluteOid. In either case, the format of the remainder of the fixed part, IEs permitted in the variable part (if any), and the associated semantics, shall be specified by the owner of the OID (see note to clause 5.6.4).

NOTE 1: This is intended to allow proprietary and experimental IE types to be carried in standard messages and to be uniquely identified.

NOTE 2: ExtendRel IEs provide a more efficient coding in the case of OIDs descended from the "enterprises" node.

6 Protocols for route and flow management

6.1 Establishing a route

6.1.1 Connection of flows

The procedures specified in the present clause 6.1 may be used to establish a route on which there are no flows (it can still carry EndToEndData messages, and can have flows added later, as specified in clause 6.3), or one or more flows may be connected at the same time as the route is established. In the latter case, guaranteed service flows shall be connected by messages passing in the same direction as the flow, while basic service flows shall be connected by messages passing in the opposite direction. A SlotAlloc or BasicAlloc IE (as appropriate) shall be present in the variable part of the FlowDescriptor IE that describes a flow if, and only if, the flow is connected by the message.

NOTE 1: The specifications for the two kinds of flow are different because guaranteed service flows need slots to be allocated starting from the source (otherwise different branches of a multicast cannot be synchronized) whereas routing of basic service flows is simpler if the destination (where the routing table is located) allocates the flow label.

A FindRoute transaction may use any number of message classes. The request message is sufficient to establish a guaranteed service flow carrying data from the originator, or a basic service flow carrying data towards it, or a route that does not carry any flows. Request and response are sufficient to establish any combination of flows. Additional message classes permit negotiation to take place before setting up the data path in the forwarding plane, for instance to agree the capacity to be used for a guaranteed service flow, to authenticate the communicating parties, or to agree payment. A flow may be connected by either of the message classes that passes in the appropriate direction.

NOTE 2: The process of connecting or disconnecting a flow in the forwarding plane is specific to the technology used for each link, and is additional to the actions specified in the present clause 6.

6.1.2 Request message

To establish a route, a control plane entity, which is referred to in the present document as the "caller", shall send one or more FindRoute requests to neighbouring control plane entities. If more than one is sent, they shall all be sent on different links. The route is then "pending" until further action is taken, as specified in clause 6.1.4.2, unless the transaction only uses one message class.

The message shall include sufficient information to identify the called party.

NOTE 1: This will usually be a CalledAddress IE but can, for instance, also be a ServiceName or SourceName IE or a DataType IE identifying the protocol used by a service.

The message shall directly contain a FlowDescriptor IE for each flow that is to be connected on the route, except where the existence of the flow is implied by other parts of the message and the caller has nothing to say about it, for instance where the call is to "take" from a source, with a default flow reference, in whatever format the called party chooses to transmit.

The flow reference may be zero, to indicate that it shall be chosen by the called party, unless the called party requires explicit flow reference values.

The IEs permitted or required inside a FlowDescriptor IE depend on the flags in its fixed part. For a guaranteed service flow, a Bandwidth IE may be included. For a basic service flow, a TrafficParams IE may be included. For flows towards the caller (which is also the owner of the route identifier) Bandwidth and TrafficParams shall specify the maximum values it can support; for flows in the opposite direction they shall specify parameters of the data stream it intends to transmit. Inclusion of SlotAlloc and BasicAlloc IEs is specified in clause 6.1.1.

The following IE types may be directly contained in each FlowDescriptor IE, where they apply to the flow, and in the message, where they apply to the route:

a) One or more DataType IEs, each specifying a different aspect of the data format, encapsulation, protocol, etc. Although the present document does not require any DataType IEs to be included, the called party may refuse the call if the message does not adequately specify information (such as the data format) that is conveyed by DataType IEs.

b) One StartTime IE; if present, then (provided the network supports delayed set-up) the flow(s) should not be set up in the forwarding plane until shortly before the indicated time.

c) One EndTime IE; if present, then after the indicated time the flow may be cleared down if any of the resources it uses are required for other calls, and if all flows have been cleared down the route may be cleared down also.

NOTE 2: The StartTime IE makes it possible to reserve capacity ahead of time for scheduled events such as live outside broadcasts. The EndTime IE can make it possible for a flow to use resources which have been reserved for a flow that is scheduled to be connected later.

d) One or more Importance, ServiceName, SourceName, DestinationName, Password, and/or ClientIdentity IEs.

e) One or more ClientIdentity IEs, which shall be "associated" with the route and form part of the state of the flow.

f) One or more Charge IEs, showing the maximum the caller is willing to pay.

g) One RouteMetric IE with status 0 or 1, and/or one with status 2.

6.1.3 Action on receiving a FindRoute request

6.1.3.1 Initial processing

A control plane entity receiving a FindRoute request shall, when it has taken the actions specified in clause 5.3, consider each condition in Table 6.1.1 in turn, and take the first action that applies.

Table 6.1.1: Action on receiving a FindRoute request

Condition	Action	Clause
Request is for a guaranteed service flow which already passes through a unit controlled by the entity	Join multicast	6.1.3.2
Requested destination is a unit controlled by the entity	Take responder role	6.1.3.3
At least one neighbouring entity is a possible route to the requested destination	Take intermediate entity role	6.1.3.4
None of the above applies	Reject request	6.2

NOTE: The mechanism for deciding whether a neighbouring entity is a possible route to the requested destination is not specified by the present document. It can use network-specific legacy protocols such as DNS [i.2], SIP [i.3], BGP [i.4] or PNNI [i.5]. Another option would be to develop a new standard for exchanging network topology information, using one or more of the transaction types which are shown as "reserved" in Table 5.4.1. Within a local network requests can simply be flooded to all neighbours, because loops are easy to detect (see the Example to clause 5.3.1).

6.1.3.2 Joining an existing multicast

The action taken to join an existing multicast shall depend on the "openness" of the multicast (see clause 5.7.16).

If the addition of a new destination requires the approval of the source, the request (modified as required, e.g. with RouteMetric IEs updated) shall be passed on to the next network element in the direction towards the source.

Otherwise, the entity shall take the responder role as specified in clause 6.1.3.3.

If the source requires to be informed of the total number of destinations, the responder shall also send a NetworkData request containing a DestCount IE (see clauses 6.4.2 and 5.7.22) towards the source, no more than five seconds after the flow is connected. There should be an interval of at least two seconds between any two such messages for the same route.

NOTE: The two second interval limits the load when multiple changes occur for the same multicast within a short space of time.

6.1.3.3 Accepting a call

A control plane entity taking the responder role shall proceed as follows.

Any Alternatives IE, shall be replaced by one of the IEs directly contained in it, except that if the chosen IE is a Group IE the Alternatives IE shall be replaced by the IEs directly contained in the Group IE. In general, any tentative data (e.g. flow references coded as zero) should be replaced with actual data.

EXAMPLE: If there is an Alternatives IE containing two or more Group IEs each containing a DataType IE and a Bandwidth IE, the Alternatives IE is replaced by a DataType IE and a Bandwidth IE, both coming from the same Group IE.

If the message does not contain any FlowDescriptor IEs that have not connected their flows, and there are no other IEs that need a reply, no further action is required other than sending an acknowledgement and any actions specified outside of the present clause 6.

NOTE 1: Such actions can include link-specific actions to connect a flow in the forwarding plane. IEs that can need a reply include Charge, RouteMetric and Delay IEs.

Otherwise, it shall send a FindRoute response in reply, containing the same IEs as in the request except for the following:

a) Charge IEs from the request message shall be removed. Charge IEs shall be added as required to show the actual charge that will be levied when the confirmation is received. Charge IEs in the response are not required to correspond to Charge IEs in the request message.

b) Any RouteMetric or Delay IE with status 0 or 2 shall be removed. Any RouteMetric or Delay IE with status 1 shall have its status changed to 3. Any of the following IEs may be added (but not more than one of each):

   - RouteMetric with status 0 or 1.
   - Delay with status 0 or 1.
   - RouteMetric with status 2.
   - Delay with status 2.

c) Any FlowDescriptor IE that connected its flow shall be removed. A SlotAlloc (for guaranteed service) or BasicAlloc (for basic service) IE shall be added if, and only if, the flow is to be connected by the response message. When a SlotAlloc IE is added, the corresponding Bandwidth IE (if present) shall be removed.

NOTE 2: In the case of a guaranteed service flow that is connected by the response message, the requirement to add a SlotAlloc IE applies even if the flow is already being transmitted to a unit controlled by the neighbour.

d) Additional FlowDescriptor IEs may be included, for instance for a flow which is implied by a DataType or ServiceName or SourceName IE but for which no FlowDescriptor IE was included in the request message.

e) If the call was to "take" a multicast, an McastRoute IE shall be added, showing the path identifier and openness assigned to the multicast by its source. There is no correlation between the path selectors in the two identifiers, but the flow reference values shall correspond. If necessary, the flow reference values in FlowDescriptor IEs shall be changed to those used in the call indicated by the McastRoute IE. (For instance, if it is clear from embedded DataType IEs which flow is which, or there is only one flow.) However, the "direction" flag in the low bit of the first octet of the FlowDescriptor IE shall still be coded as appropriate for the RouteId IE in the fixed part of the message (e.g. set to 1 for a flow towards the caller).

f) Any CalledAddress, DataType, StartTime, EndTime, Importance, ServiceName, SourceName, DestinationName, Password, and/or ClientIdentity IEs that are directly contained in the message or in FlowDescriptor IEs should be omitted. New IEs of these types may be added.

6.1.3.4 Passing the request on

A control plane entity taking the intermediate entity role shall store any information that will be needed for processing subsequent messages (see clause 5.1), and pass on a copy, modified as required (e.g. with RouteMetric and Delay IEs updated), to one or more neighbours.

NOTE: The content of ClientIdentity IEs does not need to be stored unless the neighbour is a terminal entity.

Charge IEs may be added or removed, as appropriate, for instance an IE referring to a charge paid by a user to an ISP will be removed when the ISP passes the message on. The ISP may also add a Charge IE referring to peering charges, etc.

6.1.4 Action on receiving on a FindRoute response

6.1.4.1 Action by intermediate entities

When an intermediate entity receives a valid FindRoute response message, if it has no record of the route, or has already sent a final offer (a FindRoute response message without an InterimOffer IE) for the route downtree, or has received a FindRoute confirmation message for the route, it shall send a Terminate request in reply as specified in clause 6.2.

Otherwise, if there is more than one uptree neighbour it shall proceed as specified in clause 5.3.4.3; otherwise it shall send a response downtree based on the incoming message.

Any response sent downtree shall be based on a message from one uptree neighbour and if the message sent downtree is a final offer a Terminate request shall be sent to each of the other uptree neighbours as specified in clause 6.2. A Terminate request may be sent to any of the uptree neighbours in any case, based on comparison of the incoming message with the stored messages; however, this should not be done in the case of an uptree neighbour whose response has been forwarded as an interim offer.

EXAMPLE: A request including an Alternatives IE similar to the one in example 1 to clause 5.7.19, offering three options, has been forwarded to several neighbours. One has responded offering the last of the three options. When a response offering the middle option is received, the first response can be rejected. The new response is stored, awaiting possible offers from other neighbours; if there are still offers awaited after a delay, it can be sent downtree as an interim offer. If a better offer then arrives, the second response needs to be retained in case the originator accepts the interim offer.

NOTE 1: An implementation does not need to support interim offers; it can always wait until it can send a final offer and ignore any incoming interim offers.

The message that is sent downtree shall be a copy of the incoming or stored message, modified as follows.

a) If the message being sent is an interim offer and there is more than one uptree neighbour, an InterimOffer IE as specified in clause 5.7.21, identifying the sender of the message on which it was based, shall be added.

NOTE 2: If there is only one uptree neighbour, the message will not be an interim offer unless the one on which it is based is also an interim offer, so already includes an InterimOffer IE.

b) IEs such as RouteMetric and Delay (except where the status is 3) shall be updated to add the values for the link over which the message is to be sent.

c) Charge IEs referring to charges to be paid to the operator of the control plane entity may be added. Charge IEs referring to charges to be paid by the operator of the entity should be removed.

d) Link-specific IEs such as SlotAlloc and BasicAlloc shall be modified as required to apply to the downtree link.

6.1.4.2 Action by the caller

When the originator receives a valid FindRoute response message it shall proceed as follows:

If the route is no longer pending, or for any reason the offer is unacceptable, it shall reject the offer by replying with a Terminate request as specified in clause 6.2.

NOTE 1: Reasons for rejecting an offer can include unacceptable parameters such as a data format which the unit cannot process, or an offer which is in some sense "better" (e.g. shorter route) having already been received.

Otherwise, if the message is an interim offer, or there are other recipients of the request message from which a final offer (or a Terminate request) has not yet been received, the originator may store it for later processing (when the final offer arrives, or after a timeout).

Otherwise, if the message does not contain any Charge IEs, nor any flows that are not connected by the response message, nor a RouteMetric or Delay IE with status 1, and no previous interim offers have been received, then establishment of the route is complete, the route is no longer pending, and no further action is required other than that specified in clause 5.3 and any actions specified outside of the present document.

NOTE 2: Such actions include sending an acknowledgement, noting that the transaction has been completed (so can be removed from its memory), and link-specific actions required to connect flows in the forwarding plane.

Otherwise, a FindRoute confirmation as specified below shall be sent in reply, possibly after awaiting the user's agreement that the offer is acceptable.

NOTE 3: User confirmation is particularly relevant in the case where the message contains a Charge IE. The mechanism for requesting user confirmation is application-specific and outside the scope of the present document.

A stored interim offer may be processed at any time, either to reject it (for instance because a better offer has been received) or to send a confirmation. When a confirmation of an offer is sent, all other offers awaiting processing shall be rejected.

When a confirmation of an offer has been sent, the route is no longer pending.

The FindRoute confirmation message shall contain the same IEs as in the response (though they may be in a different order), with the following exceptions:

a) All variable IEs other than FlowDescriptor, Charge, RouteMetric, Delay, McastRoute, and InterimOffer shall be removed.

b) Any RouteMetric or Delay IE with status 0, 2, or 3 shall be removed. Any RouteMetric or Delay IE with status 1 shall have its status changed to 3. Any of the following IEs may be added (but not more than one of each):

- RouteMetric with status 0 or 1.
- Delay with status 0 or 1.
- RouteMetric with status 2.
- Delay with status 2.

c) Any FlowDescriptor IE for a flow that was connected by the response message shall be removed.

d) Within each FlowDescriptor IE, DataType, StartTime, EndTime, Importance, ServiceName, SourceName, DestinationName, Password, and ClientIdentity IEs that are unchanged from the response message should be removed. A SlotAlloc (for guaranteed service) or BasicAlloc (for basic service) IE shall be added if, and only if, the flow is in the correct direction to be connected by this message. When a SlotAlloc IE is added, the corresponding Bandwidth IE (if present) shall be removed.

NOTE 4: Whereas the responder can choose whether to connect a flow by the response message or wait until it sends the completion message, the caller has no discretion in the case of the confirmation message, because it is the last opportunity to connect the flow.

e) Additional FlowDescriptor IEs may be included, for instance for a flow which it would be inappropriate to connect before the route has been chosen. SlotAlloc and Delay IEs (for guaranteed service) or a BasicAlloc IE (for basic service) shall be included if, and only if, the flow is in the correct direction to be connected by this message.

f) If there is a Charge IE in the response message, there shall be an identical Charge IE in the confirmation message, showing that the caller accepts the indicated charge. Otherwise no charge shall be levied. If there is no Charge IE in the response message, there shall be no Charge IE in the confirmation message.

6.1.5 Passing on a FindRoute confirmation

When an intermediate entity receives a valid FindRoute confirmation message, it shall first check whether the message includes an InterimOffer IE that identifies itself. If so, before processing the message it shall remove the IE (but not any other InterimOffer IEs) and send a Terminate request to all uptree neighbours other than the one identified in the IE.

Unless there is now exactly one uptree neighbour, the transaction shall be abandoned by sending a Terminate request to each neighbour that is participating in it, as specified in clause 6.2.

NOTE: The number of uptree neighbours is expected to have been reduced to 1 when the response message was sent, unless it was an interim offer, in which case the confirmation is expected to include the entity's InterimOffer IE.

Otherwise, it shall pass the message on uptree. Link-specific IEs such as SlotAlloc and BasicAlloc shall be modified as required to apply to the link over which the message will be sent. IEs such as RouteMetric and Delay shall be modified as required. Any Charge IEs that correspond to Charge IEs which the entity added to the response message shall be removed, and Charge IEs corresponding to any Charge IEs which the entity removed from the response message shall be added. Also, if the entity added a Charge IE to the response message but no Charge IE corresponding to it is found in the confirmation message, then no charge shall be levied for the call and the entity may abort the call by sending Terminate request messages towards both the responder and the caller as specified in clause 6.2.

6.1.6 Action of the responder on receiving a FindRoute confirmation

When a control plane entity receives a valid FindRoute confirmation message for a FindRoute transaction for which it is a responder, if the message does not contain any flows that are not connected by the confirmation message, nor a RouteMetric or Delay IE with status 1, no further action is required other than that specified in clause 5.3 and any actions specified outside of the present document.

NOTE 1: Such actions include sending an acknowledgement, noting that the transaction has been completed (so can be removed from its memory), and link-specific actions required to connect flows in the forwarding plane.

Otherwise, it shall send in reply a FindRoute completion message which shall contain the same IEs as in the confirmation, with the following exceptions:

a) All IEs directly contained in the message except FlowDescriptor, RouteMetric, Delay and McastRoute shall be removed.

b) Any RouteMetric or Delay IE with status 0, 2 or 3 shall be removed. Any RouteMetric or Delay IE with status 1 shall have its status changed to 3.

c) Any FlowDescriptor IE for a flow that was connected by the confirmation message shall be removed.

d) Within a FlowDescriptor IE, DataType, StartTime, EndTime, Importance, ServiceName, SourceName, DestinationName, Password, and ClientIdentity IEs that are unchanged from the confirmation message should be removed. A SlotAlloc (for guaranteed service) or BasicAlloc (for basic service) IE shall be added.

NOTE 2: The flow is assumed to be in the correct direction to be connected by this message, because the last opportunity to connect any flows in the other direction is the confirmation message. However, the entity can check and regard the message as invalid if the flow is not able to be connected by the completion message.

e) Additional FlowDescriptor IEs may be included, but only for flows that are connected by this message. Any such FlowDescriptor IEs shall therefore include a SlotAlloc or BasicAlloc IE as appropriate.

6.1.7 FindRoute completion message

When an intermediate entity receives a valid FindRoute completion message, it shall proceed as follows:

If it has no record of the route, it shall send a Terminate request in reply.

Otherwise, it shall pass the message on downtree, with IEs such as RouteMetric and Delay updated as appropriate. Link-specific IEs such as SlotAlloc and BasicAlloc shall also be modified as required to apply to the downtree link.

When the caller receives a valid FindRoute completion message, it shall take no action other than that specified in clause 5.3 and any actions specified outside of the present document.

NOTE: Such actions include sending an acknowledgement, noting that the transaction has been completed (so can be removed from its memory), and link-specific actions required to connect flows in the forwarding plane.

6.2 Disconnection of routes and flows

6.2.1 General

Whereas a FindRoute transaction extends from one end of the route to the other, disconnection proceeds independently between each pair of neighbours. This makes the process more robust, simplifies clearing down a route from the middle (for instance, when a link breaks), and allows calls to be rerouted around a broken link or failed node without tearing down and rebuilding the whole route. It also simplifies the situation where a branch of a multicast is being pruned back from a destination that has ceased to receive it at the same time as another destination in the same part of the network is joining it.

When a route is cleared down, all flows that follow the route shall also be cleared down. Other routes for the same call shall not be affected; to clear a call down, each route shall be cleared down separately.

Alternatively, an individual flow may be cleared down along a route, leaving other flows in place. Again, to remove a flow from a call that has several routes the flow shall be cleared down separately for each route.

Clearing down of a flow or route may be initiated by any of the units through which it passes.

NOTE 1: A single message can request clearing down of multiple flows and/or routes. This is useful in the event of a broken link.

The Terminate transaction only uses one message class.

NOTE 2: There is no reply to a Terminate request, only an acknowledgement which guarantees that the recipient will ensure the indicated routes and/or flows do not exist at its end of the link.

6.2.2 Terminate request message

A control plane entity shall request clearing down of one or more flows and/or routes across a link by sending a Terminate request message containing one or more Route IEs to its neighbour at the other end of the link. Except when a ClientIdentity IE is included, it shall also take whatever action is necessary to remove the flows in the forwarding plane and adjust (or remove) its record of the route.

NOTE 1: The action to clear a flow down in the forwarding plane is specific to the implementation of the link.

NOTE 2: Including a ClientIdentity IE signals that only the identified destination is to be cleared down.

A Route IE that directly contains at least one FlowDescriptor IE shall request clearing down of the selected flow(s), leaving the route in place even if all its flows have been removed. A Route IE with no FlowDescriptor IEs directly contained in it shall request clearing down of the route and all its flows.

In the case of a multicast for which a FindRoute response has been received by the sender of the Terminate message, the Route IE shall show the source's route identifier, not the caller's. Between sending a FindRoute response and receiving the reply, an entity shall be prepared to receive a Terminate message with either identifier and shall interpret the "direction" bit in a FlowDescriptor IE in the context of the identifier in the Route IE in which it is contained.

The message may contain one or more Cause IEs. A Cause IE directly contained in the message shall apply to all flows and/or routes in the message. A Cause IE directly contained in a Route IE shall apply to all flows selected by it, also to the route if it selects the route (i.e. if it does not contain any FlowDescriptor IEs). A Cause IE contained in a FlowDescriptor IE applies to that flow. If there is no Cause IE applying to a route or flow, the cause shall be assumed to be "normal call clearing".

The message should not contain any other IEs.

6.2.3 Action on receiving a Terminate request message

6.2.3.1 Initial processing

A control plane entity receiving a Terminate request message containing one or more Route IEs shall take the actions specified in clause 5.3, and then process each Route IE by considering each condition in Table 6.2.1 in turn and taking the first action that applies.

Table 6.2.1: Action on receiving a FindRoute request

Condition	Action	Clause
IE specifies a flow or route which the entity does not believe is present on the link, or which it is already in the process of removing	Ignore the IE	
IE does not directly contain a ClientIdentity IE	Clear the flow/route down	6.2.3.2
IE directly contains a ClientIdentity IE	Clear matching destinations down	6.2.3.3

NOTE: The first condition covers the case where the message is a repetition. It also avoids problems in the event that clearing down of a route is requested from both ends at the same time.

6.2.3.2 Clearing down a link within a route

In the case of a Route IE that does not directly contain a ClientIdentity IE the entity shall take whatever action is necessary to remove the flows in the forwarding plane and adjust (or remove) its record of the route.

NOTE 1: The action to clear a flow down in the forwarding plane is specific to the implementation of the link.

If the link carries a multicast flow in the direction away from the entity, then if there is at least one other such link no other action shall be taken other than to send a NetworkData message towards the source if required (see clause 6.4.2). If there are no other such links, the link from the source shall be cleared down.

Otherwise if there is a Cause IE coded with R=1 (see clause 5.7.17), the entity may attempt to reconnect the route.

NOTE 2: Procedures for reconnecting a flow around a fault, also for handover between base stations in wireless networks, are for further study. In the case of a multicast, reconnection around a fault is only attempted from the destination's side of the fault.

Otherwise the entity shall clear the specified route and/or flow(s) down on all other links on which it is connected.

6.2.3.3 Clearing down a link when a ClientIdentity IE is included

If a Route IE directly contains a ClientIdentity IE, the route and/or flows shall not be cleared down except as follows.

a) If the recipient of the message is a terminal entity that is identified by the contents of the ClientIdentity IE, it shall request clearing down of the route and/or flows by sending a Terminate request message that does not contain any ClientIdentity IEs, as specified in clause 6.2.2.

b) If the route is connected over one or more links where the neighbour is a terminal entity and the ClientIdentity IE matches a ClientIdentity IE remembered from the FindRoute request that set the route or flow up, then for each of those links it shall send a Terminate request containing a copy of the Route IE with the ClientIdentity IEs removed, and take the actions specified in clause 6.2.3.2 as if it had received such a message on the link.

If neither of the above applies, and the message arrived on the link that connects towards the source of the flow(s), then a Terminate request containing a copy of the Route IE shall be sent on each other link traversed by the route.

NOTE: This allows the source of a multicast to remove an individual destination. The message will be sent on every branch of the multicast but only disconnect destinations where the ClientIdentity IE matches. It is not expected to be used in any other circumstances. Intermediate entities do not need to remember any ClientIdentity values except for the case where the neighbour is a terminal entity, which is included to allow for the possibility that the terminal device might choose to continue to receive content when the source requests its disconnection.

6.3 Adding new flows to an existing route

The procedure for adding flows to an existing route shall be as specified in clause 6.1 except that the message type shall be AddFlow instead of FindRoute, the route identifier in the fixed part of the messages shall identify an existing route, and the parts of the procedure concerned with establishing the route shall be omitted.

IEs which are used only for establishing the route, such as CalledAddress, should be omitted.

6.4 Information messages related to a route

6.4.1 General

NetworkData and EndToEndData messages shall be used to convey along a route information that is not part of a flow.

A control plane entity receiving an EndToEndData message shall send an acknowledgement for it (see clause 5.3.1) and also proceed as follows:

a) If the entity has no record of the route, it shall send a Terminate request for the route in reply (in addition to the acknowledgement).

b) Otherwise, if the entity is an endpoint of the route it shall process the IEs it contains and may send a reply.

c) Otherwise, it shall pass the message on to the next entity (or entities, in the case of a message from the source of a multicast) along the route.

A control plane entity receiving a NetworkData message shall proceed as for an EndToEndData message except that in case c) it shall also take any action specified elsewhere in the present clause 6.4, and make appropriate changes to IEs such as RouteMetric and Delay that are contained in it, before passing the message on.

6.4.2 Notification of the number of destinations

A control plane entity shall generate a NetworkData request message containing a single DestCount IE to inform the source of a change in the number of destinations as specified in clause 6.1.3.2.

Before passing on a NetworkData request message containing a DestCount IE a switch shall update its own record of the number of destinations and add to the number in the IE the number of destinations reached via neighbours other than the one from which the message was received.

The message shall not be passed on until two seconds have elapsed since the previous NetworkData request message (if any) was sent for the same route. If further such messages for the same route are received during that time, only one shall be passed on.

6.4.3 Out-of-band data

An application may use an EndToEndData message to send information along a route where that information is not part of any flow. The message shall include a UserData IE holding the data unit to be conveyed, and may include one or more DataType IEs which indicate how it should be interpreted.

A message from the source of a multicast shall be delivered to all destinations. A message from a destination shall be delivered to the source.

An application shall not send an EndToEndData request message if less than two seconds have elapsed since a previous EndToEndData request message was sent for the same route.

NOTE: EndToEndData is intended for occasional protocol messages between application entities in the endpoints. It is not intended to be used instead of connecting an basic service flow.

6.4.4 Changing service parameters

The specification of the ChangeAllocation transaction is for further study.

6.4.5 Checking flows and routes

The specification of the CheckFlow transaction is for further study.

NOTE: The CheckFlow transaction is intended to be used to verify that a flow is correctly connected in the control plane and the forwarding plane

7 Frame alignment for the guaranteed service

7.1 General

7.1.1 Classification of links between units

Point-to-point links carrying Flexilink packets between units are classified in the present document as "tightly" or "loosely" synchronized.

A tightly synchronized link is one which carries a regular stream of slots which can be allocated to guaranteed service flows, such that the recipient can identify the flow to which a packet belongs by the slot in which it arrives; and on which the propagation time is constant to within a small enough tolerance that no de-jitter buffering is needed. A loosely synchronized link is one that does not meet that specification.

EXAMPLE: The format specified in ETSI GS NGP 013 [1], clause 8.2.2 is tightly synchronized. A link tunnelled over UDP is loosely synchronized.

NOTE: A link does not count as tightly synchronized unless it is in the "active" state (see clause 7.2.2.3.3).

7.1.2 Tightly synchronized links

Data on a tightly synchronized link shall be carried in "frames". It shall be possible to align outgoing frames with incoming frames as specified in clause 7.1.5, or with an external reference, without needing to synchronize clocks.

NOTE 1: That can be achieved by transmitting a continuous stream of frames with a small gap between them, and adjusting the number of gap symbols.

Frames shall be grouped into "framing periods" which are nominally 32 ms each. The actual length of a framing period shall be in the range 31,986561 ms to 32,0032 ms. There may be additional jitter in the arrival time of incoming frames; the specification of the format used for the link should set limits for this jitter.

NOTE 2: There are 64/*m* allocation periods (as specified in ETSI GS NGP 013 [1], clause 6.3.1) in a framing period. The specified range covers allocation periods based on 0,5 ms as well as on the 0,49984 ms specified in ETSI GS NGP 013 [1]; the frame format specified in ETSI GS NGP 013 [1], clause 8.2.2.1 can support the 0,5 ms figure provided the maximum number of inter-frame gap octets is increased from 16 to 18.

Frames may be divided into "subframes".

NOTE 3: Guaranteed service flows can be routed through a buffer that holds one subframe from each network port, with incoming data continuously overwriting the previous subframe. The number of slots in a subframe sets an upper limit on the latency, as well as defining the size of the buffer memory. To allocate slots on an outgoing link it is necessary to know in what part of the outgoing framing period the flow's packet will be available in the buffer. This requires the phase difference between the incoming and outgoing framing periods to be constant to within a tolerance that is much less than the size of a subframe; the outgoing framing period does not need to start at the same time as the incoming period, but it does need always to start at the same point in each incoming period.

EXAMPLE: In the case of the prototype implementation, which uses the format specified in ETSI GS NGP 013 [1], clause 8.2.2, a subframe is 30 or 31 slots; there are four subframes in a frame and 512 frames in a framing period, and the first frame in each framing period has its frame type coded as binary 0101 0000. If framing is based on the 0,49984 ms period the average number of octets of inter-frame gap will be between 13 and 15 if all clock frequencies are within 50 ppm of nominal; if based on 0,5 ms it will be between 15,5 and 17,5.

7.1.3 Loosely synchronized links

On loosely synchronized links, guaranteed service packets should be carried in basic service packets as specified in ETSI GS NGP 013 [1], clause 7.1.

NOTE: A de-jitter buffer will be needed for any flow that is received on a loosely-synchronized link and forwarded on a tightly-synchronized link.

The framing periods at the two ends of a loosely-synchronized link need to be frequency-locked if it is to carry guaranteed service flows (see clause 7.1.6).

7.1.4 Classification of groups of units

An "island" consists of one or more units that are connected to each other by tightly synchronized links (as defined in clause 7.1.1), with no tightly synchronized links to units that are not part of the island.

NOTE 1: Thus, any unit that is reachable over tightly synchronized links is part of the island. There can be loosely synchronized links within an island, for instance if the island covers two sites which are connected by a tightly synchronized link and also by a link tunnelled over legacy technology which is used as a back-up.

A "cloud" consists of one or more islands that are connected to each other by loosely synchronized links, with no loosely synchronized links to units that are not part of the cloud.

A "sync domain" is a set of units whose framing is aligned, as follows:

a) One of the units is the "local framing reference" and transmits frames that are synchronized either with an internal (free running) clock or with an external reference such as TAI.

b) Each unit other than the local framing reference aligns its transmitted frames with incoming frames on one of its links, referred to as its "upstream" link.

c) The upstream links form a spanning tree with the local framing reference at its root.

d) A "downstream" link is the link partner's upstream link.

e) All other tightly synchronized links are "off-tree" links.

If an island is not a single domain, all its sync domains should use the same external reference.

NOTE 2: The protocols will not allow a link between domains to transition to the "active" state, i.e. to become tightly synchronized, unless the two ends have a common reference. However, they will not detect if two external references with the same clock class have different frequencies.

NOTE 3: Unlike timing references used for clock synchronization, there is no quality requirement for an internal clock other than the tolerance on the length of the framing period specified in clause 7.1.2.

7.1.5 Alignment of frames within an island

Each unit that is not a local framing reference shall align its outgoing frames (on all links) with the incoming frames on its upstream link, by starting each frame when a specific octet of the incoming framing period (its "trigger" octet) is received on the upstream link.

NOTE 1: Where the incoming and outgoing links use the same frame format, all the trigger octets in a framing period will be at the same point in the incoming frame. If the links have different numbers of frames in a period, the relationship will be more complex.

In the case of units that have more than one network interface, the upstream link and the trigger octet(s) shall be selectable by the control plane entity.

NOTE 2: In the case of units with just one network interface, the trigger octet can be fixed, perhaps as the first octet in the incoming frame. This is referred to as "simple loopback" timing, and a link on which it is used is always on the spanning tree, with the unit being a leaf. See also clause 7.2.1.

Frames received on the upstream link will by definition always be closely aligned with outgoing frames; on other links there will be tolerance which is the sum of the uncertainty (as specified in clause 7.3.3.2) on the link in the inbound direction and the variation between the sender's and recipient's references. In the case of downstream links the latter will be the uncertainty on the link in the outbound direction.

For off-tree links within a sync domain, the neighbouring units will be connected to the local reference by different routes, so the variation in their references will be the sum of the uncertainties along the two routes.

NOTE 3: If the two routes from the reference share some links, the variation will in practice be the sum of the uncertainties from the point where they diverge, but the protocols are not specified to report the uncertainty from every possible point of divergence.

NOTE 4: The procedures specified in clause 7.2 ensure that any on-tree link will remain on-tree until it is disconnected, though the direction (i.e. which is upstream and which is downstream) can reverse. Connection or disconnection of a link can cause the tree to be reconfigured, and when that occurs the uncertainty on an off-tree link can increase.

In any sync domain that includes links with a propagation time of more than about 1 000 slot times, the reference should be derived from a stable external reference such as TAI.

NOTE 5: The procedures can cause a change to a different reference, which can cause a step change in the frame rate unless the timing is synchronized to an external reference. If there is a step change on a long downstream link, there can be significant drift between transmitted and received frames while the change propagates to the far end and back (see note 1 to clause 7.3.3.2).

7.1.6 Alignment of frames within a cloud

Although frames in different islands within a cloud do not need to be phase aligned, they do need their average frame rates over time to be equal because otherwise if a flow that passes from one island to another does not include any empty slots, and the second island has a lower frame rate, it will eventually fill its de-jitter buffer and lose packets.

If both ends of a loosely synchronized link have frame rates referenced to TAI, no further action is required.

NOTE 1: This can also be the case for clock classes that are not shown in AES67 [16], Table A.2 as using the PTP timescale, but the present document does not define how to establish whether two such references have the same frequency.

The procedures specified in clause 7.2 ensure that the loosely synchronized links between islands across which the timing propagates form a spanning tree in the same way as the tightly synchronized links within each island. Each unit for which such a link is an upstream link shall be its sync domain's local sync reference; the unit for which it is a downstream link may be any unit in its island.

NOTE 2: There will be significant wander in the frame timing recovered from a loosely synchronized link. However, the uncertainties for any off-tree links within a sync domain will depend only on their paths to the sync domain's local reference.

Procedures not specified in the present document may also be used, for instance if a timing reference is available from the technology over which the loosely synchronized links are implemented.

7.1.7 Establishment of sync domains

When a unit is powered on or otherwise reset, assuming at that point it does not have any links connected, it shall form a single-domain island within which it is the sync reference.

When a tightly synchronized link enters "await sync" state, the link partners shall exchange information regarding their sync domains (see clause 7.2.2.3). If they are in the same sync domain (i.e. have the same local reference) or are in domains using the same external reference, the link is an off-tree link and each unit can set up guaranteed service flows as soon as it has discovered the phase relationship between its frames and the link partner's.

NOTE: The tree is not reconfigured, even if the new link allows one of the units to get better timing information than via its existing upstream link. To do so would mean demoting a link that was previously on-tree, which would increase the phase uncertainty on that link, change the uncertainty on other off-tree links, and possibly result in further reconfigurations so that the tree becomes unstable.

If they are part of different islands, the two islands will be amalgamated. The one with the "better" reference (as specified in clause 7.2.2.3.2) will be unchanged. The other will first be reconfigured (if necessary) so that the link partner is its sync reference, and then the new link will be set to be its upstream link.

A similar process shall also be used when a loosely synchronized link is connected. If the two units are both in the same cloud or have the same frequency reference, the link is off-tree. Otherwise, the two clouds are amalgamated with the new link being on-tree and the unit for which it is the upstream link becoming the sync reference for its island.

7.1.8 Action on link down

If a link which is part of the spanning tree goes down, the unit downstream of it shall become the sync reference of a new sync domain.

NOTE 1: The change does not result in reconfiguration of any other links.

It shall freeze the frequency of the frames it generates at the value it had just before the link went down, and announce the new configuration to the units that are downstream of it in the tree, showing the state as unaligned (see clause 7.2.2.2). If any of them has an off-tree link to the previous domain, it shall act as if that link had just come up.

NOTE 2: This will normally result in the tree being quickly reconfigured, with one of the off-tree links being promoted to on-tree before the reference has time to drift significantly. There is, however, no need for a "holdover" state because any drift is measured directly, see clause 7.2.2.1.3.

7.2 Procedures

7.2.1 End equipment

End equipment with a single network interface may use simple loopback timing. If it does not also support legacy formats, it should generate a free-running stream of frames if no incoming frames are detected. If it does not support guaranteed service flows it may generate a free-running stream of frames independently of what is being received.

NOTE 1: If legacy formats are supported, the procedures specified in Annex A, or equivalent, can be used. Otherwise, the equipment needs to generate free-running frames in order to provoke the link partner into switching to Flexilink mode at link-up. A discontinuity is to be expected when the link partner's frames begin arriving if simple loopback timing (where the phase relationship between incoming and outgoing frames is hard-wired) is used.

Equipment using simple loopback timing should not be able to act as the reference for frame timing. Procedures to prevent it doing so are for further study.

NOTE 2: When acting as the reference the unit needs to generate a free-running stream of frames. If a "better" reference is identified it then needs to switch to loopback timing, which in the "simple" case will cause a discontinuity in the framing and disrupt any guaranteed service flows it is transmitting.

NOTE 3: End equipment designed for Flexilink networks can also be used for point-to-point communication, as a replacement for standards such as MADI and SDI, provided it uses process specified in clause 7.2.2.3.

7.2.2 Network elements that route between multiple interfaces

7.2.2.1 General

7.2.2.1.1 Frame alignment

Each unit that is not a sync reference shall phase-align all its outputs with the incoming frames on its upstream link by adjusting the gap between frames as required to ensure that each outgoing frame begins as close as possible to a defined point in the corresponding incoming frame.

NOTE: The outgoing frame timing is thus tightly coupled to the incoming frames; this causes a small amount of jitter in the frame timing (because the clocks are not synchronized), but the resulting uncertainty in alignment for off-tree links is less than has been found to arise from the drift that occurs with software control of the frame rate or with clock synchronization. Also, it does not need any special provision to cope with the step change in frequency that can occur when changing sync reference.

7.2.2.1.2 Maintaining frame rate

The unit shall also count the total adjustment made in 32 framing periods (approximately 1,024 s) and remember the most recent count. If its upstream link is disconnected (so that it becomes a sync reference), it shall maintain the same rate of adjustment, with the adjustments being evenly spread, for instance using a binary rate multiplier.

NOTE: This ensures that if a link carrying timing is lost, dividing a sync domain into two, off-tree links between the two parts will remain aligned for long enough to reconfigure the tree and re-unite the two parts.

7.2.2.1.3 Checking alignment

The timing of incoming frames on each link shall be monitored to ensure it is within the limits assumed for the routing of any guaranteed service flows that are received on the link. Forwarding of guaranteed service flows shall be conditional on the relevant framing being within its limits.

NOTE 1: Disconnection of an on-tree link can cause the tree to be reconfigured in a way that increases the uncertainty across some off-tree links, though in most cases the difference will be small.

NOTE 2: Monitoring of the phase relationship between frames avoids the possibility of forwarding data in the wrong slots if the uncertainty has not been estimated correctly or if the references at the two ends of an off-tree link drift apart for any reason.

7.2.2.2 Frame alignment states

The present document distinguishes three states of a unit's frame alignment, as follows (including the corresponding FrameSyncInfo message format):

a) unaligned: frame rate fixed relative to an internal clock; no upstream link; clockClass 248, no TimingRelay IE;

b) external reference: frame rate fixed relative to an external reference such as TAI; no upstream link; clockClass less than 248, no TimingRelay IE;

c) aligned: frame rate aligned with the upstream neighbour; at least one TimingRelay IE.

7.2.2.3 Link up

7.2.2.3.1 Initiating frame alignment

If, when a tightly synchronized link comes up, the information exchange with the link partner shows it uses loopback timing, the link can transition immediately to the "aligning" state.

NOTE: Procedures for identifying that the link partner uses loopback timing are currently link-specific; more generic procedures are for further study. See also clause 7.2.1.

Otherwise, it shall enter "await sync" state and send a downstream FrameSyncInfo request message (see clause 7.3.4) to the link partner.

7.2.2.3.2 "Await sync" state

A unit receiving a downstream FrameSyncInfo request message on a link that is in "await sync" state shall compare the information in the message with its own sync source, or its "new" sync source (see clause 7.3.3.3) if it has one, taking the following steps in sequence until either a "common" reference is identified or one side's reference is identified as "better":

a) The sender's "relay chain" shall be constructed, consisting of the TimingRelay IEs from the message (if any) followed by a TimingRelay IE containing the sender's identifier and an estimate of the uncertainty introduced by the link. The unit's own relay chain shall be constructed similarly.

b) For each unit identifier in the sender's relay chain, in order from last to first, if it is present in the unit's own relay chain the identified unit shall be a "common" reference, with an uncertainty value that is the sum of all the uncertainty values that follow it in each relay chain.

c) If the clockClass value in the TimingSource IE in the message is equal to the clockClass value for the unit's local frame sync reference and also less than 248, the identified source of timing should be a "common" reference, with an uncertainty value that is the sum of all the uncertainty values in both relay chains.

d) If the two clockClass values are not both greater than or equal to 248, the one with the smaller value shall be a "better" reference.

e) Otherwise, interpreting the first identifier in each relay chain as an unsigned integer, the one with the smaller value shall be "better".

NOTE: Comparing identifiers is a convenient way to decide which of the two sync references will be used, but has no other significance. The sync reference in an island is not necessarily the unit with the smallest identifier; for instance, when an on-tree link fails the unit downstream of it becomes sync reference without comparing its identifier against others in the island, to avoid unnecessary reconfiguration of the tree.

If a common reference is identified, the link shall be identified as off-tree with the calculated uncertainty value and shall enter the "aligning" state.

If its own sync source is "better" the link shall stay in the "await sync" state until a handover message or a further downstream message is received on it.

Otherwise the unit shall initiate the process for transitioning the link to on-tree (see clause 7.2.2.3.4), keeping the link in the "await sync" state until the process is complete or a further downstream message is received on it.

If a further downstream message is received on the link while it is still in the "await sync" state, any state associated with the previous message shall be deleted before processing the new message.

Similarly, when a downstream message is received on the upstream link, processing of the most recent downstream message on each link in "await sync" state shall be repeated.

7.2.2.3.3 Aligning frames

When a link is in "aligning" state, the unit shall observe the point in the incoming frames at which outgoing frames begin, and configure the trigger octets (see clause 7.1.5) appropriately.

The link shall then transition to "active" state, in which guaranteed service flows can be routed across it.

NOTE: The trigger octets on the upstream link show when to start outgoing frames. The trigger octets on all links show at what point in outgoing frames the guaranteed service packet in a given slot is available to be forwarded (see note 3 to clause 7.1.2).

The phase relationship established when a link enters the "active" state shall be maintained until it goes down.

7.2.2.3.4 Joining one sync domain to another

When, using the procedure of clause 7.2.2.3.2, a unit discovers that its link partner has a "better" sync reference, it shall create a "new" sync source as specified in clause 7.3.3.3.

If the unit is not the local sync reference for its domain it shall then send an upstream message as specified in clause 7.3.5.

The unit shall switch to using the link as an upstream link when it acknowledges a handover message on its upstream link, or immediately if it is already the local sync reference, sending a handover message to the link partner as specified in clause 7.3.6 and downstream messages on all its other links as specified in clause 7.3.4. It shall also set the link to the "aligning" state.

7.2.2.4 Link down

When an off-tree or downstream link goes down, no action is required other than re-routing flows that traverse it.

When a unit's upstream link goes down, the unit becomes a sync reference, setting the frame rate as specified in clause 7.2.2.1.2, and shall send a downstream message on all its remaining links (as specified in clause 7.3.4) announcing that it is now the sync reference.

Each unit that has one or more off-tree links, on receiving a downstream message that shows its sync reference has changed, shall start a timer, to run for a period which shall be one second or such other time (appropriate to the particular network) as may be configured.

NOTE 1: The delay needs to be long enough for the information that it the link partner is also in the new sync domain to be propagated, in the case where it is downstream of the broken link.

On expiry of the timer the unit shall follow the procedures specified in clause 7.2.2.3 for each of its off-tree links, except that they shall remain in the "active" state unless the alignment check specified in clause 7.2.2.1.3 fails.

NOTE 2: If the new domains are A and B, with A being "better", and there are several off-tree links between A and B, the unit on the B side of each of them will send an uplink message. The handover message will go to one of them, which will then change its link to an uplink and report to the rest of the units in the B domain that they are now in the A domain again - its link partner now identifies the link as downstream and all the others (on both sides) find their links are off-tree within domain A again, though probably with a different uncertainty value.

7.3 FrameSyncInfo signalling messages

7.3.1 Communicating entities

Many of the functions specified in clause 7, including setting and measuring frame timing, need to be implemented in the forwarding plane. The descriptions of the remaining functions apply to the case where each cluster (see clause 5.1) is a single unit and each link carries a signalling flow. Where that is not the case, the forwarding plane shall operate as if it were.

In any case, actions specified to be carried out by a unit shall where appropriate be carried out by its control plane entity.

7.3.2 Data unit format

The format of a FrameSyncInfo message or acknowledgement shall be as specified in Table 5.5.1 or 5.5.2 respectively.

The message class shall always be a request. None of the messages shall have a variable part.

7.3.3 State information

7.3.3.1 Information for each link

For each network port, a unit shall remember:

a) the link's state, and the phase relationship between incoming and outgoing frames;

NOTE: Guaranteed service flows can be routed across the link if, and only if, it is in the "active" state.

b) any FrameSyncInfo messages that have been sent but not acknowledged; and

c) the link's role in the tree (upstream, downstream, or off-tree), and for upstream and off-tree links the information from the downstream message that was most recently received (which defines the link partner's sync reference).

7.3.3.2 Sync source

The information a unit transmits in the TimingSource and TimingRelay IEs in downstream messages shall describe the source of its frame timing, referred to in the present document as its "sync source".

For each link, in each direction, a specific event during transmission of a slot shall be defined as the "reference point".

EXAMPLE: In the case of Ethernet, possible reference points include transmission of the first octet on the Media-Independent Interface (MII) or on the cable.

The uncertainty value in a TimingSource IE shall be the maximum difference between theoretical and actual intervals between the reference points of any two slots (not in general adjacent slots) by the unit whose identifier is in the first TimingRelay IE, or by the unit sending the FrameSyncInfo message if there are no TimingRelay IEs.

The uncertainty value in a TimingRelay IE shall be the difference between the maximum and minimum latency between the reference point of a slot transmitted by the unit whose identifier it includes and the reference point of a slot transmitted by the unit whose identifier is in the following TimingRelay IE, or by the unit sending the FrameSyncInfo message if there are no more TimingRelay IEs.

NOTE 1: Uncertainty is thus a measure of jitter and wander arising from issues such as changes in latency where signals cross the boundary between clock domains or pass through FIFOs. It does not directly relate to the accuracy of absolute time or of frequency, although a significant component of wander is the time for a change in frame rate to propagate across the tree (see note 5 to clause 7.1.5) and the worst case effect will depend on how big the change can be, i.e. the frequency tolerance.

NOTE 2: A different definition of uncertainty is used for distribution of network time, see clause 8.2.3.

In the case of a local framing reference (i.e. a unit that does not have an upstream link), the sync source shall consist of a TimingSource IE reporting the source of the unit's frame timing.

Otherwise, it shall be constructed from the sync source of the upstream neighbour (i.e. the link partner on the upstream link, as reported in the most recent downstream message received from it) by either:

a) adding the uncertainty value for the upstream link to the uncertainty value in the last TimingRelay IE, or in the TimingSource IE if there is no TimingRelay IE; or

b) adding a TimingRelay IE containing the upstream neighbour's identifier and the uncertainty value for the upstream link.

Option b) should be taken if the upstream neighbour is the local framing reference or a gateway to a wider network, or the uncertainty value in the last TimingRelay IE is large.

NOTE 3: Taking option b) reduces the uncertainty for off-tree links that are further downstream but increases the size of messages.

7.3.3.3 "New" sync source

When a unit receives an upstream or downstream message which reports a "better" sync reference as specified in clause 7.2.2.3.2, it shall construct a sync source (the "new" sync source) as specified in clause 7.3.3.2 but substituting the link on which the message was received (the "new" upstream link) for the upstream link.

The new sync source and upstream link, and whether the incoming message was upstream or downstream, shall be remembered until:

a) it becomes the sync source; or

b) the new upstream link goes down; or

c) it is removed or replaced as a result of receiving a further downstream message.

NOTE: A "new" sync source can be removed as a result of receiving a downstream message on the "new" upstream link reporting a source that is not "better".

7.3.4 Downstream messages

Each unit shall send a downstream message, coded with its current sync source, on each downstream or off-tree link (including links in "await sync" or "aligning" state) whenever its sync source changes, except as specified in clause 7.3.6. It may also send downstream messages when there has been no change, at intervals of not less than 30 s.

A downstream message received on the upstream link shall cause the unit's sync source to be updated.

Any other link on which a downstream message is received can be assumed to be off-tree; if the message is different from the previous downstream message received on that link it shall be processed as specified in clause 7.2.2.3.2.

7.3.5 Upstream messages

A unit shall send an upstream message on its upstream link to request transfer to it of the local sync reference role. Upstream messages shall not be sent on any other link. The message shall be coded with the unit's "new" sync source in the same way that its current sync source is coded in a downstream message.

An upstream message received on the upstream link shall be acknowledged but otherwise ignored.

A unit receiving an upstream message which does not report a "better" source (as specified in clause 7.2.2.3.2) shall take no action other than to acknowledge the message.

When a unit that is not the local sync source receives an upstream message reporting a "better" source it shall remember it as its "new" sync source and send an upstream message on its upstream link.

NOTE: The forwarded message will not be the same as the incoming message, because it will include the uncertainty on the "new" upstream link.

When a unit that is the local sync source receives an upstream message reporting a "better" source it shall reply with a handover message (in addition to acknowledging the upstream message).

7.3.6 Handover messages

7.3.6.1 General

A handover message notifies the recipient that the link is (or will be, when the acknowledgement is received) a downstream link.

NOTE: If it is received on the upstream link, the recipient no longer has an upstream link and therefore becomes the local sync reference.

7.3.6.2 Handover message on the upstream link

When a unit receives a handover message on its upstream link it shall switch to local timing before it sends the acknowledgement, setting the frame rate from the measurement in the most recent period (see clause 7.2.2.1.2).

If it has no new source, it shall send downstream messages as specified in clause 7.3.4 but take no other action.

NOTE 1: It will therefore become, and remain as, the local sync reference. There are various scenarios in which this can occur, including when a newly-connected link goes down, or a link to a better domain comes up, before the handover process has been completed; in that case the downstream messages can initiate further changes of sync source.

Otherwise, it shall send a handover message on the new upstream link.

If the new source was created from a downstream message the unit shall replace its sync source with its "new" sync source before sending the handover message. It shall also send downstream messages as specified in clause 7.3.4.

NOTE 2: This is the case where the new upstream link is newly-connected or being converted from an off-tree link. The unit can start aligning its frames with the other domain immediately; doing so before sending the message means the link partner can enter "aligning" state immediately on receiving the message.

If the new source was created from an upstream message, then when it receives the acknowledgement (but not before), the sender shall replace its sync source with its "new" sync source. After switching to taking its timing from the new upstream link, it should trim the phase relationship between incoming and outgoing frames on the link so that it is as close as possible to the value it had when the link was downstream.

NOTE 3: The switch to the new upstream link is delayed until the link partner has become the sync source. While the acknowledgement is in transit both units will be sync sources; in effect, the sync domain is split into two parts which are then reunited.

NOTE 4: Trimming the phase relationship reduces the likelihood that the phase will drift if connection and disconnection of links elsewhere on the network results in a link changing direction many times.

NOTE 5: The unit is not specified to send downstream messages in this case, because they would quickly be superseded as the sync reference role is handed on further.

7.3.6.3 Handover message on other links

A handover message received on an off-tree link notifies the recipient that the link is now a downstream link. If it is in "await sync" state, it should transition to "aligning" state (see clause 7.2.2.3.2 and note 2 to clause 7.3.6.2).

A handover message received on a downstream link should be acknowledged but otherwise ignored.

8 Network time

8.1 Timing information

8.1.1 General

The system needs to be able to distribute "real time" accurately enough to allow audio to be aligned to within ±5 % of a sample time, which is approximately 130 ns for 384 kHz sampling and 1 μs for 48 kHz.

NOTE: These limits are specified in AES11 [i.6].

8.1.2 Local time

Each unit shall count time since its last reboot; this is "local" time. It shall also record the offset between local time and network time, which is distributed across the network in a similar way to the phase-alignment of frames, though not (in general) from the same source or by the same route. Local time shall be monotonic; network time may have discontinuities, for instance when changing to a different reference.

8.1.3 Timing data format

Time shall be counted in seconds, with the fractional part being represented as an integer number of nanoseconds in the range 0 to 999 999 999.

NOTE 1: This format is compatible with PTP. Although PTP uses 32 bits to carry the fractional part, its maximum value is 0x3B9AC9FF so it is representable in 30 bits.

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| secs  |                    nanoseconds                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.1.1: Timing word

When network time is coded in the 32-bit "timing information" field in Flexilink frames, bits 0 and 1 shall carry the least significant 2 bits of the integer part and bits 2 to 31 inclusive shall carry the fractional part, as illustrated in Figure 8.1.1. The value transmitted shall be the sender's network time sampled on a particular event related to transmission of the frame or of the previous frame. For each type of link there should be specified either a specific event or a number of events from which one shall be selected as part of the process of setting the link up.

All-ones in the timing field shall indicate that timing information is not supplied. Other code points with a value greater than 999 999 999 in bits 2 to 31 are reserved but may be defined for particular link types.

NOTE 2: On links where no further code points are defined, the all-ones coding can be recognized by checking bits 2 to 5 inclusive.

8.1.4 Updating network time

The offset between local time and network time (see clause 8.1.2) shall be adjusted each time a value is received on the link which has been chosen by the software as the source of network timing, by resetting the offset. This provides a "hard" alignment, similar to the alignment of frames. Local time shall be trimmed to minimize the drift in the offset values; this ensures that if the reference is lost the unit will continue to track network time as accurately as possible. It can be a "soft" alignment, but it does not participate significantly in the propagation of network time across the network so any wander, etc. it introduces will not accumulate.

NOTE: Equipment can be designed such that the local clock frequency does not need to be adjusted in order to trim the rate at which local time advances. If the clock frequency is 125 MHz, local time will advance by 8 for each clock cycle; as with frame timing (see clause 7.2.2.1.2), a binary rate multiplier circuit can be used that occasionally advances it by 7 or 9 instead.

Network time is not mission-critical to the routing of guaranteed service flows, so there is no requirement for the entire network to use a single reference and there is no need for a "handover" protocol. Each unit shall advertise the source of values in the "timing" field of the frames it transmits, and neighbouring units can choose to use it or not, depending on how the quality compares with rival sources. However, network time can be mission-critical for media synchronization, so a common source should be used wherever possible.

Audio clocks can be aligned with local references such as a word clock or DARS, or with network time; in general, local "house clock" inputs should take priority over network time, because they can be assumed to define the timing required by the equipment to which the outputs are connected. Also, network time may be aligned with local references.

8.2 TimeSyncInfo signalling messages

8.2.1 Communicating entities

Many of the functions specified in clause 8, including updating the offset between local time and network time from incoming frames and setting the current network time in outgoing frames, need to be implemented in the forwarding plane. The descriptions of the remaining functions apply to the case where each cluster (see clause 5.1) is a single unit and each link carries a signalling flow. Where that is not the case, the forwarding plane shall operate as if it were.

In any case, actions specified to be carried out by a unit shall where appropriate be carried out by its control plane entity.

8.2.2 Data unit format

The format of a TimeSyncInfo message shall be as specified in Table 5.5.1, and the acknowledgement shall be as specified in Table 5.5.2.

The message class shall always be a request, and the subtype downstream.

If the clockClass value in the TimingSource IE is not shown in AES67 [16], Table A.2 as using the PTP timescale, there may be a variable IE (not specified in the present document) identifying the source.

8.2.3 Sync source

The information a unit transmits in the TimingSource and TimingRelay IEs shall describe the source of its network time, referred to in the present document as its "time source".

The uncertainty value in a TimingSource IE shall be the maximum difference between the reference indicated by the clockClass value and network time as reported by the unit whose identifier is in the first TimingRelay IE, or by the unit sending the TimeSyncInfo message if there are no TimingRelay IEs.

The uncertainty value in a TimingRelay IE shall be the maximum difference between network time as reported by the unit whose identifier it includes and network time as reported by the unit whose identifier is in the following TimingRelay IE, or by the unit sending the TimeSyncInfo message if there are no more TimingRelay IEs.

NOTE 1: In TimeSyncInfo messages the uncertainty is a measure of the accuracy of network time as a measure of absolute time. Provided the frequency of the reference is stable, it indirectly provides an upper limit to jitter and wander.

NOTE 2: A different definition of uncertainty is used for frame alignment, see clause 7.3.3.2.

In the case of a local timing reference (i.e. a unit that does not have an upstream link), the time source shall be coded as a TimingSource IE reporting the source of the unit's network time.

NOTE 3: The unit's network time can be derived from an external source such as a GPS receiver. It can also be a fixed offset from local time, which in turn is derived from a free-running local oscillator, in which case the clockClass value will be 248.

Otherwise, it shall be constructed from the time source of the upstream neighbour (i.e. of the link partner on the upstream link, as reported in the most recent downstream message received from it) by either:

a) adding the uncertainty value for the upstream link to the uncertainty value in the last TimingRelay IE, or in the TimingSource IE if there is no TimingRelay IE; or

b) adding a TimingRelay IE containing the upstream neighbour's identifier and the uncertainty value for the upstream link.

Option a) should be taken in most cases; option b) should be taken if the upstream neighbour's message does not contain a TimingRelay IE and the clockClass value in the TimingSource IE is not shown in AES67 [16], Table A.2 as using the PTP timescale.

NOTE 4: Taking option b) allows different sources of the same class (e.g. different "house sync" signals) to be distinguished.

8.2.4 Protocols

The links over which network time is distributed do not need to form a spanning tree, though they do need to avoid forming loops. As with frame alignment messages, links can be classified as upstream, downstream, or off-tree, but these classifications have less effect on the protocol; in particular, the same messages are sent on an upstream link as on the other types.

When a link is connected the "timing" field in Flexilink frames shall be coded as all-ones until a TimeSyncInfo message has been sent to the link partner and the acknowledgement received.

NOTE 1: A unit can implement more than one instance of network time, e.g. one tracking TAI and another tracking house sync, and use different instances on different links. The present document does not specify how to negotiate the use of a particular instance.

If the source of network time is lost, including when the upstream link is disconnected, also when changing to a different source, the "timing" field in transmitted frames on each link shall be coded as all-ones until a new source is selected and a TimeSyncInfo message has been sent to the link partner and the acknowledgement received.

In the case where timing information on the upstream link is lost and information from the same source is available on another link but with a greater uncertainty, before selecting the other link the unit shall delay for a period which shall be one second or such other time (appropriate to the particular network) as may be configured.

NOTE 2: The delay allows the loss of timing information to propagate to other units that are affected by it, in case that includes the proposed new upstream unit.

TimeSyncInfo messages may also be sent when there has been no change in the timing source, at intervals of not less than 30 s.

8.2.5 Accuracy of tracking network time

Each unit shall include the facility for its control plane entity to provide an offset to the network time received from its upstream link, to allow for the latency on the link and the time between the point in a frame where the sender's time is sampled and the point where the recipient's time is sampled. In the absence of any better information, this offset should be set to the mean of the value reported in messages on the link and the value from its own measurements. The "uncertainty" value needs to take account of how similar the delays in the two directions are likely to be.

Annex A (normative): 1 Gb/s Ethernet physical links

A.1 Link negotiation using Ethernet packets

A.1.1 General

For interfaces that support the standard Ethernet MAC layer, link establishment occurs in three stages. First the procedures specified in IEEE 802.3 [17] result in the PHY reporting establishment of the Ethernet link. Then negotiation packets are exchanged, as a result of which the units at the two ends of the link agree to switch to the Flexilink physical link format. Finally, the mechanism specified in clause 7 is used to align frames so that guaranteed service flows can be supported.

A.1.2 Packet format

```
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   0 |                                                               |  31
     +      destination address      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  32 |                               |                               |  63
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+      source address           +
  64 |                                                               |  95
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  96 |1 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1|0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0| 127
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 128 |1 0 1 0 1 0 0 0|0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0|  frame type   | 159
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 160 |                          timing word                          | 191
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.1.1: Packet header when using Ethernet MAC layer

Link negotiation packets shall be basic Ethernet MAC frames as specified in IEEE 802.3 [17]. They shall consist of the headers illustrated in Figure A.1.1 followed by information elements as specified in clause A.1.5.

The destination address in a LinkRequest packet shall be 01-80-C2-00-00-01. In all other negotiation packets it shall be the individual MAC address of the destination interface.

NOTE 1: Packets with destination address 01-80-C2-00-00-01 are not forwarded by IEEE 802.1D -conformant bridges [i.9].

The source address shall be the individual MAC address of the interface by which the packet is transmitted.

Bits 96 to 111 inclusive are the length/type field and shall be coded with the OUI Extended EtherType value $88B7_{16}$ specified in IEEE 802-2014 [8], clause 9.2.2. Bits 112 to 151 inclusive are the Protocol Identification field specified in IEEE 802-2014 [8], clause 9.2.4 and shall be coded with $0090A80002_{16}$.

The next octet (bits 152 to 159 inclusive) shall be coded with the message type as listed in Table A.1.1. The next four octets (bits 160 to 191 inclusive) shall carry timing information as specified in clause 8. The remainder of the packet shall contain information elements as specified in clause A.1.5.

NOTE 2: Bits 144 to 191 are the Common Header specified in clause B.2.1.

NOTE 3: In most cases timing information will only be implemented for Flexilink frames, so in negotiation packets the timing field will be coded as all-ones.

Table A.1.1: Negotiation message types

Frame type (hex)	Message type	Clause
00 to 7F	other frame types	B.2.1
80	LinkRequest	A.1.4.2
81	LinkAccept	A.1.4.3
82	LinkReject	A.1.4.4
83	reserved	
84	LinkKeepalive	B.4.3
85 to FF	reserved	

A.1.3 Information required to support the negotiation process

A.1.3.1 States of an interface

Equipment containing an interface shall include a management entity which shall monitor the Link Status reported by the physical layer as specified in IEEE 802.3 [17], clause 22.2.4.2.13. The management entity shall control a "reset" signal which affects the state of the interface. The "reset" signal shall be true at power-on and at any time that the physical layer reports that the link is (or has been) "down".

For the purposes of the protocol descriptions in the present document, an interface shall be in one of the following states:

a) passive

b) requesting

c) accepting

d) connecting

e) connected

The interface shall be in the "passive" state at any time that the "reset" signal is true. The management entity shall be able to request the interface to initiate the negotiation procedure, but only when "reset" is false and the interface is in the "passive" state.

In the "connecting" state the interface is sending Flexilink frames but has not detected that it is receiving Flexilink frames.

In the "connected" state the interface is sending and receiving Flexilink frames. This state encompasses the "await sync", "aligning" and "active" states of the link used in clause 7.

A.1.3.2 Timer

An interface shall have a single timer which has states "running" and "stopped".

Where the protocol descriptions in this clause specify that the timer is "set" for a particular time, it shall enter the "running" state for the specified time (±100 ms), at the end of which a "timeout" event shall occur and the timer shall enter the "stopped" state.

If the timer is set again while still in the "running" state, the timeout event related to the first occasion on which it was set shall not occur.

Where the protocol descriptions in this clause specify that the timer is "stopped", it shall enter the "stopped" state without causing a timeout event.

The timer shall be in the "stopped" state at any time that "reset" is true; assertion of "reset" shall not cause a "timeout" event.

A.1.3.3 Configuration

For each interface, there shall be "configuration" parameters. For each parameter, the set of values supported by the interface is the "supported range"; the value actually used when in "connected" state is the "actual value". The "supported configuration" is composed of the supported ranges of all parameters; the "actual configuration" is composed of the actual values of all parameters.

Implementations may provide the management entity with the ability to read and/or write the supported configuration while "reset" is true, and/or to read the actual configuration while the interface is in the "connected" state.

Wherever appropriate, the operation of the interface shall depend on the values of its configuration parameters.

NOTE: The way in which parameters are encoded is intended to allow each side to specify a set of values it supports (see clause A.1.5), so that the outcome of the negotiation process is the intersection of the values supported by both sides.

A.1.4 Negotiation procedure

A.1.4.1 Initiation of procedure

When the physical layer announces establishment of a link on a network port that supports both Ethernet and Flexilink MAC layers, provided the link is full-duplex and at a speed that is supported by the Flexilink MAC layer, the management entity shall request initiation of the negotiation procedure. Other protocols, such as DHCP, should also be attempted, to establish whether the link is to a legacy network.

When the management entity requests initiation of the negotiation procedure, no action shall be taken if "reset" is true or the interface is not in "passive" state. Otherwise, the supported configuration shall be transmitted in a LinkRequest message, the timer shall be set for 500 ms, and "requesting" state shall be entered.

A.1.4.2 Reception of LinkRequest message

When a LinkRequest message is received in "passive" or "requesting" state, the timer shall be stopped and an "actual value" of each parameter shall be chosen as specified in clause A.1.5.

If the identifier in the SenderIdentification IE is the same as the recipient's, indicating a looped-back connection, or there is at least one parameter for which an "actual value" cannot be chosen, the supported configuration shall be transmitted in a LinkReject message and the "passive" state shall be entered.

Otherwise, the resulting actual configuration shall be transmitted in a LinkAccept message and the "accepting" state shall be entered.

A.1.4.3 Reception of LinkAccept message

LinkAccept messages shall be ignored when in the "passive" state.

When a LinkAccept message is received in the "requesting" or "accepting" state, the timer shall be stopped and the actual values in the message shall be checked to see whether they specify a configuration that is supported by the recipient. If so, the actual configuration shall be loaded from the message, and the "connecting" state shall be entered. Otherwise, the supported configuration shall be transmitted in a LinkReject message and "passive" state shall be entered.

Any parameters not included in the LinkAccept message should take the value specified in clause A.1.5 for the case where the parameter is not included in a LinkRequest message.

When a LinkAccept message is received in "connecting" state, if all parameters in the message have values compatible with the actual configuration then no further action shall be taken. Otherwise, the actual configuration shall be transmitted in a LinkReject message and "passive" state shall be entered.

NOTE: Normally, a LinkAccept will only be received in the "connected" state if both sides sent a LinkRequest when the link came up; the process for deriving the actual configuration is intended to ensure that the configurations derived by both sides are identical.

A.1.4.4 Reception of LinkReject message

On reception of a LinkReject message, the timer shall be stopped and the interface shall enter the "passive" state.

A.1.4.5 Timeout

Timeout should only occur in "requesting" state.

On timeout, the LinkRequest message may be retransmitted and the timer set for at least 500 ms. Otherwise, "passive" state shall be entered. The management entity may try a fixed number of times and then give up, or it may continue trying forever. It may set the timer for a longer interval on each attempt.

A.1.4.6 Reception of Flexilink frames

If the interface detects that it is receiving valid Flexilink frames in any state other than "connected", it shall stop the timer, enter the "connected" state, and apply the "link up" procedure specified in clause 7.2.2.3.

If it was in "passive" or "requesting" state, it shall apply an appropriate default configuration. It may also apply further protocols, such as the Simple Control Protocol specified in Annex C, to exchange information with the link partner.

A.1.5 Information Elements (IEs) in negotiation packets

A.1.5.1 IE format

```
0             7 8          15 16
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ...  +-+-+-+-+-+-+-+-+-+
|      tag      |   length     |        information              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ...  +-+-+-+-+-+-+-+-+-+
```

Figure A.1.2: IE format where the tag value is nonzero

The first octet of an IE shall be the tag, which defines the type of information as specified below. If the tag value is zero, the remainder of the IE is padding which extends to the end of the message, as shown in Figure A.1.3. All tag values not specified in clause A.1.5 are reserved for future standardization.

If the tag value is non-zero, the second octet shall be coded with the number of octets of information, and the information shall begin in the third octet, as shown in Figure A.1.2.

Future revisions of the present document may specify additional "information" octets for particular information element types. Therefore, information elements shall be transmitted with the length specified in clauses A.1.5.3 to A.1.5.6 but on reception longer information elements shall be accepted, with the additional "information" octets being ignored.

The coding of each IE type is specified in the clause indicated in Table A.1.2.

Table A.1.2: Information elements in negotiation packets

Tag (hex)	IE type	Clause
00	Padding	A.1.5.2
01 to 81	reserved	
82	SenderIdentification	A.1.5.3
83	SignallingFlow	A.1.5.4
84	RecipientIdentification	A.1.5.5
85	LinkType	A.1.5.6
86 to FF	reserved	

A.1.5.2 Padding IE

```
0             7 8
+-+-+-+-+-+-+-+-+-+-+  ...  +-+-+-+-+
|0 0 0 0 0 0 0 0|    padding        |
+-+-+-+-+-+-+-+-+-+-+  ...  +-+-+-+-+
```

Figure A.1.3: Padding IE format (tag value is zero)

Information element type 00_{16} shall be interpreted as "end of message". If present, it shall be the last information element. The "padding" field may consist of any number of octets (including zero), and each octet may contain any value.

A Padding IE shall be present if required to meet the minimum packet size (miFrameSize) specified in IEEE 802.3 [17] clause 4.4.2, and may also be used (for example) to round the packet length up to a whole number of words.

A.1.5.3 SenderIdentification IE

The information field in a SenderIdentification IE shall consist of 8 octets coded with the sender's identifier as specified in clause 4.2.

Absence of a SenderIdentification IE shall indicate that the sender does not have an identifier.

A.1.5.4 SignallingFlow IE

The SignallingFlow IE shall be used to specify the flow label to be used on signalling messages to the sender.

If the information field is not the same number of bits as the "flow label" field in forwarding plane packets, the value shall be truncated or zero-extended at the most significant end.

NOTE: In the case of the format specified in ETSI GS NGP 013 [1], clause 8.2.2.2, a flow label is 13 bits; values 0 to 255 can be coded in one octet, and all values can be coded in two octets by adding three zero bits at the most significant end. The CRC is not included and needs to be calculated by the recipient of the message.

If there is no SignallingFlow IE, signalling messages shall use flow label zero.

A.1.5.5 RecipientIdentification IE

The information field in a RecipientIdentification IE shall consist of 8 octets coded with an identifier as specified in clause 4.2.

In the case of a LinkAccept message that is a reply to a LinkRequest that did not include a SenderIdentification IE, a RecipientIdentification IE shall be included, coded with an identifier that the recipient should adopt. The identifier shall have a 1 in bit 6 as specified in clause 4.2.

Any other message may include a RecipientIdentification IE to confirm the link partner's identity.

If the recipient of a message containing a RecipientIdentification IE does not have an identifier and the identifier in the RecipientIdentification IE has a 1 in bit 6, it shall adopt the identifier in the message as its identifier for as long as the link is up. If it has an identifier which is the same as the identifier in the RecipientIdentification IE it shall take no further action. Otherwise it shall reply with a LinkReject message.

A.1.5.6 LinkType IE

```
 0 1   3 4     7 8                                       29  31
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|log2m| type  |                reserved                 |T|F|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.1.4: "Link type" word

The information field in a LinkType IE in a LinkAccept message shall consist of 4 octets coded with a "link type" word as illustrated in Figure A.1.4. In other message types it shall consist of a multiple of 4 octets coded with one or more "link type" words, in decreasing order of preference, showing the link types the sender can support.

Within the "link type" word:

- bits 1 to 3 shall be coded with $\log_2 m$, where *m* is as specified in ETSI GS NGP 013 [1], clause 6.3.1 as shown in Table A.1.3;
- bits 4 to 7 shall be coded as shown in Table A.1.4;
- bit 30 shall be coded with 1 if the sender supports timing information, 0 otherwise;
- bit 31 shall be coded with 1 if the sender is able to process FindRoute requests, 0 otherwise; and
- bits 0 and 8 to 29 are reserved and shall be coded as all-zero on transmission and ignored on reception.

Table A.1.3: Allocation period

Code	*m*	Allocation period (ms)	Frames per allocation period	Slots per allocation period
0	1	0,5	8	976
1	2	1	16	1 952
2	4	2	32	3 904
3	8	4	64	7 808
4	16	8	128	15 616
5	32	16	256	31 232
6	64	32	512	62 464
7	reserved			

Table A.1.4: Link type codes

Code (decimal)	Link type
0	reserved
1	virtual link as specified in Annex B
2	physical link using the Flexilink MAC layer with normal frame alignment (see clause 7.1.5)
3	physical link using the Flexilink MAC layer with simple loopback timing (see clause 7.1.5)
4 to 15	reserved

On a link within an island, bits 30 and 31 shall both be set. On an "external" link, they show what functions the client implements.

A.2 Link negotiation using Flexilink signalling messages

This is for further study.

A.3 Data transfer

The format on the link shall be as specified in ETSI GS NGP 013 [1], clause 8.2.2.2 except that there may be up to 18 gap octets between frames. The first frame of each framing period (see clause 7.1.2) shall have its frame type coded as binary 0101 0000.

NOTE 1: There are 512 frames in a framing period.

The default event at which the value coded in the 32-bit "timing information" field is sampled (see clause 8.1.3) shall be when the "frame type" octet is presented to the PHY via its Media Independent Interface (see Figure 22-1 of IEEE 802.3 [17]).

NOTE 2: Ideally the event would be related to transmission on the medium; if events other than the default are defined in future revisions of the present document their use can be signalled in the LinkType IE using some of the reserved bits in the link type word.

A.4 Link-specific IE formats

A.4.1 SlotAlloc IE on tightly synchronized links

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          anchor slot          |        number of gaps         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               e               |  table ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...                    .
```

Figure A.4.1: Start of SlotAlloc IE on tightly synchronized links

The fixed part of a SlotAlloc IE on a tightly synchronized link shall be coded as specified in clause 5.7.14.2, with the initial three integers occupying 16 bits each (as illustrated in Figure A.4.1) and being interpreted as unsigned.

In the case of an allocation of a single slot, when the number of gaps will be zero and the table will be empty, the third integer (the value for *e*) may be omitted.

The first slot in a frame with its type coded as binary 0101 0000 shall be the first in an allocation period. The "trailing" background octets, together with the framing octets up to and including the timing information in the following frame, shall be treated as a slot which cannot be allocated to any flow.

NOTE: This slot, which is included in the counts in Table A.1.3, will be 64 to 70 octets in length. Including it makes routing of guaranteed service flows easier because the length of a gap of a specific number of slots can be assumed to be independent of whether it spans a frame boundary.

The variable part shall be empty.

A.4.2 SlotAlloc IE on loosely synchronized links

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        slots per second                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.4.2: SlotAlloc IE on loosely synchronized links

The fixed part of a SlotAlloc IE on a loosely synchronized link shall consist of four octets containing an unsigned 32-bit integer containing the number of slots per second required for the flow, as specified in clause 5.7.14.3 and illustrated in Figure A.4.2.

The variable part shall be empty.

A.4.3 BasicAlloc IE

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                        label                      |    CRC    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.4.3: Fixed part of BasicAlloc IE

The fixed part of a BasicAlloc IE shall consist of two octets containing the value to be used in the "label" field of packets transmitted on the link in the direction towards the sender of the message (including the three CRC bits) as specified in ETSI GS NGP 013 [1], clause 8.2.2.2 and illustrated in Figure A.4.2.

The variable part shall be empty.

Annex B (normative): Virtual links

B.1 Encapsulation

B.1.1 General

A "virtual link" is an association between two interfaces that are attached to a legacy network; it shall be set up using the negotiation packets specified in Annex A. Multiple virtual links can be connected on a single interface.

There are two kinds of virtual link:

- external: links to terminal devices such as PCs; and
- internal: links between islands within a cloud.

The "external" case is asymmetrical, with the terminal device being referred to as the "client" and the other as the "switch"; the client is not expected to be able to implement any of Flexilink's time-related features. The "internal" case is symmetrical.

Encapsulation in Ethernet (layer 2) and in UDP (layer 3) are specified in the present document. Packets may also be encapsulated in other technologies.

B.1.2 Encapsulation in Ethernet

Encapsulation in Ethernet packets shall be as specified in clause A.1.2 and illustrated in Figure A.1.1. Bits 144 to 191 are the common header which is further specified (with the bits renumbered) in clause B.2.1.

Timing information should be included on internal links except in negotiation packets.

NOTE: External links, and the negotiation process on internal links, can be implemented using software processes that are not able to insert accurate timing information.

A link is identified by the source and destination MAC addresses (bits 0 to 95). Thus there can only ever be one link between a pair of interfaces using Ethernet encapsulation, though there may be multiple links between different interfaces on the same two units.

Internal links should use Ethernet encapsulation where possible, to reduce overheads.

B.1.3 Encapsulation in UDP

When implemented over UDP, a link is identified by the IP addresses and UDP port numbers. Thus there can be several separate links between a pair of interfaces. External links will typically use UDP; this allows each client process running in a computer to have its own link, and avoids the need to inject timing information that is not supported by the hardware.

In the UDP header, the port numbers shall be as specified in clause B.3.1. The checksum may be set to zero, to avoid needing to buffer the packet while the checksum is being calculated.

The data shall be as specified in clause B.2.

B.2 Payload formats

B.2.1 Common header

```
                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                             0 |0 0 0 0 0 0 1 0|  frame type   | 15
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
16 |                         timing word                       | 47
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B.2.1: Common header

The data for all types of packet or frame shall begin with the six octets illustrated in Figure B.2.1.

The first octet shall contain the value 2; other code points for this octet are reserved. The second octet shall contain a frame type code as listed in Table B.2.1. The next four octets shall contain a timing word coded as specified in clause 8.1.3.

The content of the remainder shall depend on the frame type code, and is the same as on the other transports.

NOTE: The number of octets in an Ethernet header is an odd multiple of 2, so the alignment illustrated applies when encapsulation is directly in Ethernet (as in Figure A.1.1) and also for UDP/IP over Ethernet. The value in the first octet is inherited from AES51 [i.7]; this octet is retained to preserve the alignment.

Table B.2.1: "Type-and-format" coding

Code (hex)	Content	Clause
00 to 25	reserved	
26	Basic service packet	B.2.3
27	Link timing information	B.2.4
28 to 3F	reserved	
40 to 50	Physical link frame	ETSI GS NGP 013 [1], clause 8.2.2.1
51 to 7F	reserved	
80 to FF	Negotiation message	A.1.2 and B.2.2

B.2.2 Negotiation packets

The data for a negotiation packet shall consist of a common header with a frame type listed in Table A.1.1, followed by IEs as specified in clause A.1 as modified by clause B.3.

B.2.3 Basic service frames

```
                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                             0 |0 0 0 0 0 0 1 0|0 0 1 0 0 1 1 0| 15
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
16 |                         timing word                       | 47
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
48 |         length        | CRC |         label         | CRC | 79
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B.2.2: Basic service packet header

Each basic service frame shall carry a single basic service packet, as illustrated in Figure B.2.2. The frame type shall be coded as 26_{16} and the format of the basic service packet shall be as specified in ETSI GS NGP 013 [1], clause 8.2.2.2.

If padding is added after the packet payload to meet the minimum length requirement of the underlying network, it shall be coded as all-ones.

NOTE: The label for a flow is assigned by the recipient of the flow (see note 1 to clause 6.1.1) so where there are several virtual links connected to the same interface the unit can allocate flow labels for them from a common number space, in which case the forwarding plane will not need to examine the legacy network headers to establish which link the packet arrived on.

B.2.4 Link timing frames

```
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                              0 |0 0 0 0 0 0 1 0|0 0 1 0 0 1 1 1|  15
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  16 |                          timing word                        |  47
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  48 |resvd|   period    |     frame     |         octet           |  79
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   80 |                                                              |
     |                            padding                          |
     |                                                             | 143
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 144 |          reserved            |        label         |resvd| 175
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B.2.3: Link timing information

The data for a link timing frame shall consist of 22 octets as illustrated in Figure B.2.3.

Bits 48 to 79 contain phase information showing which octet was being transmitted on physical links over 1 Gb/s Ethernet when the network time coded in the timing word was sampled. Bits 48 to 50 are reserved. Bits 51 to 57 count the framing periods specified in clause 7.1.2; this is a free-running counter and its value in different link timing frames can be used to detect jitter or wander that exceeds 32 ms. Bits 58 to 66 code the frame number within the framing period. Bits 67 to 79 code the octet number within the frame.

Bits 80 to 143 are available for the receiving interface to insert its own timing word and phase information.

Bits 160 to 172 contain the flow label used for signalling messages; this serves to identify the link without needing to examine the legacy network headers.

B.3 Establishment of a link

B.3.1 General

Each virtual link shall have state and information as specified in clause A.1.3 except that there shall be no "passive" or "connecting" state.

NOTE 1: Where several virtual links exist on the same interface, each virtual link has its own state.

The negotiation procedure shall be initiated by transmitting the supported configuration in a LinkRequest message, setting the timer for 500 ms, and entering "requesting" state.

Link Request packets may be broadcast, multicast or unicast. Appropriate multicast addresses and UDP port numbers are for further study.

NOTE 2: Link Request packets can be unicast if an interface is configured to establish a link with a specific interface on another unit.

All packets except the Link Request shall be unicast. The address and, if required, port number shall be the sender's address and port number from the Link Request or Link Accept message.

B.3.2 Reception of LinkRequest message

When a LinkRequest message is received for a virtual link that is in the "requesting" state, its timer shall be stopped. When a LinkRequest message is received for a virtual link that does not currently exist, the link shall be created. In either case, an "actual value" of each parameter shall be chosen as specified in clause A.1.5.

If the identifier in the SenderIdentification IE is different from the recipient's and an "actual value" can be chosen for every parameter, the resulting actual configuration shall be transmitted in a LinkAccept message and the "connected" state shall be entered.

NOTE: If the identifier in the SenderIdentification IE is the same as the recipient's this indicates a looped-back connection to another interface on the same unit.

Otherwise, the supported configuration shall be transmitted in a LinkReject message unless the LinkRequest was to a broadcast or multicast address; in any case the virtual link shall be deleted.

B.3.3 Reception of LinkAccept message

When a LinkAccept message is received for a virtual link that is in the "requesting" state, the timer shall be stopped and the actual values in the message shall be checked to see whether they specify a configuration that is supported by the recipient. If so, the actual configuration shall be loaded from the message, and the "connecting" state shall be entered. Otherwise, the supported configuration shall be transmitted in a LinkReject message and the virtual link shall be deleted.

Any parameters not included in the LinkAccept message should take the value specified in clause A.1.5 for the case where the parameter is not included in a LinkRequest message.

B.3.4 Reception of LinkReject message

On reception of a LinkReject message, the timer shall be stopped and the virtual link shall be deleted.

B.3.5 Timeout

Timeout should only occur in "requesting" state.

On timeout, the LinkRequest message may be retransmitted and the timer set for at least 500 ms. Otherwise, the virtual link shall be deleted. The management entity may try a fixed number of times and then give up, or it may continue trying forever. It may set the timer for a longer interval on each attempt.

B.3.6 Link termination

Either party may terminate a link at any time by sending a Link Reject; the other party should confirm termination by sending a Link Reject of its own.

B.3.7 Address translation

Virtual links that pass through middleboxes implementing Network Address Translation are for further study.

B.4 Data transfer

B.4.1 Basic service packets

Basic service packets shall be conveyed in basic service frames as specified in clause B.2.3.

The flow number used for signalling messages shall be as signalled in the SignallingFlow IE in LinkRequest and LinkAccept packets. Other flows shall be connected and cleared down using the same signalling messages as on other links.

The recipient should check that the packet header in bits 48-79 is valid, i.e. that bits 48 and 49 are zero and both CRCs are correct, and ignore the packet if not. The payload length shall be as signalled in bits 50 to 60; if this is less than the length indicated by the encapsulation (e.g. UDP length or length of Ethernet packet) the additional data shall be ignored, if greater the missing data is undefined and the packet may be treated as errored.

NOTE: Support for fragmentation of basic service packets that are longer than the MTU of the legacy network is for further study.

On internal links, the timing field should contain the sender's network time at the time the packet was transmitted, in the same way as on tightly-synchronized links. On external links, the client is not expected to support processing of the timing information, and the timing field may be coded as all-ones.

B.4.2 Guaranteed service packets

Guaranteed service packets shall be encapsulated in basic service packets as specified in ETSI GS NGP 013 [1], clause 7.1.

The FindRoute request message shall include an AsyncAlloc IE specifying the label to be used in the basic service packet header, and the fixed part of the SyncAlloc IE in the response message shall consist of the number of guaranteed service packets per allocation period coded as a 32-bit integer.

B.4.3 Keepalives

Link timing frames as specified in clause B.2.4 shall be transmitted on internal links, and may be transmitted on external links, at intervals of between 2 and 2,5 seconds. They show the phase relationship between Flexilink frames at the two ends of the link as well as showing that the link is still active. This allows the frame structure to be loosely phase-locked as specified in clause 7.1.6.

On any link on which link timing frames are not used, LinkKeepalive packets (see Table A.1.1 and clause B.2.2) shall be transmitted, either at intervals of between 2 and 2,5 seconds or whenever no frame has been transmitted for approximately 2 seconds. LinkKeepalive packets do not include any IEs, and thus consist only of the legacy protocol header and the six-octet common header illustrated in Figure B.2.1. A link on which no incoming frame has been seen for 8 seconds should be cleared down.

Annex C (normative): Simple Control Protocol

C.1 General

The Simple Control Protocol provides a mechanism for identifying a device that is plugged into a network interface without requiring it to implement signalling or management protocols.

NOTE: This is similar to the way the 2-wire serial interface on SFP modules is used.

It allows the control plane entity of the unit that contains the network interface (the "client") to identify the device, and to configure the interface appropriately, for instance if the device is a "simple" device that transmits a defined information stream but does not implement signalling.

C.2 Request message

```
                                   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                 0 | control |     read length     |  15
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
16 |                             address                           |  47
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
48 |                             address                           |
   //                          data to write                       //
   |                                                               | nnn
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C.2.1: Request message format

Request messages are transmitted by the client to the device.

Each request message shall be a basic service packet with an all-ones label. The payload is illustrated in Figure C.2.1 and consists of the following fields:

- Bits 0 to 4 may be used for control functions within the device; when coded as 00000 there shall be no effect on the operation of the device, nor of the protocol.
- Bits 5 to 15 shall code the number of data octets to be included in the reply.
- Bits 16 to 47 shall code an address which selects the information in the device to be read and/or written. The address shall be aligned as a bit address and the device may ignore the least significant bits.

EXAMPLE: If the address selects byte-addressable memory bits 45 to 47 will be ignored; if the memory is 32 bits wide bits 43 to 47 will be ignored.

- Bits 48 onwards shall contain data to be written at the address.

The length of the data to be written shall be deduced from the length of the message; if the message is 6 octets long nothing shall be written. Similarly, if bits 5 to 15 are coded with zero nothing shall be read.

C.3 Reply message

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  0 |                           address                           |  31
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 32 |                                                             |
//                            data                            //
    |                                                             | nnn
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C.3.1: Reply message format

Reply messages are transmitted by the device to the client.

Each reply message shall be a basic service packet with label value 1. The payload is illustrated in Figure C.3.1 and begins with four octets containing an address. Any further octets are data.

Any bits in the address that the device ignores shall be coded as zero. If there is any data present, the address shall be the address of the first data bit.

C.4 Protocol

The client may send request messages at any time, but the device may ignore any that it is unable to process, for instance if it has not finished processing a previous request, or if writing to a read-only location is requested.

After processing a request the device should send a reply message with address and data length corresponding to the request. It may adjust the address and length, for instance to align the data on a word boundary. In the case of a request with non-empty data and non-zero read length, the reply shall show the value after completing the write operation.

NOTE 1: The reply therefore shows whether the data have been written successfully.

The first 32 bits of the address space shall be read-only; the first 24 bits shall contain an OUI, and the next 8 bits shall contain an identification of the device family, defined by the owner of the OUI. The interpretation of all other addresses, and of non-zero values in the control bits, shall be dependent on the device family.

NOTE 2: Thus the client can discover the device family be reading four (or more) octets from address zero.

The device may also send unsolicited reply messages, for instance to report a change in its state, but only if enabled by a request.

NOTE 3: Thus the client does not need to reserve label value 1 unless it implements the Simple Control Protocol.

History

Document history		
V1.1.1	November 2022	Publication