Specification of a Communications Infrastructure for Fire Service Mobilising Systems

GD-92/1003A/2.2 Copy 1

List of contents

List of contents

List of contents

List of figures and tables

List of abbreviations

ASCII	(United States of) American Standard Code for Information Interchange
AVLS	Automatic Vehicle Location System
BCC	Block Check Character
CCITT	International Consultative Committee for Telegraphy and Telephony
CE	Communications Entity
Comms	Communications (system)
DCE	Data-Circuit Terminating Equipment
DTE	Data Terminal Equipment
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
GOSIP	Government Open System Interconnection Profile
ISDN	Integrated Services Digital Network
LAN	Local Area Network
LJU	Line Jack Unit
MDT	Mobile Data Terminal
MHS	Message Handling System
MIS	Management Information System
Mobs	Mobilising (system)
MODEM	MOdulator / DEModulator
MTA	Message Transfer Agent
MTBF	Mean Time Between Failures
MTS	Message Transfer System
MTTR	Mean Time To Repair
NMS	Network Management System

List of abbreviations

NTU	Network Termination Unit
OSI	Open Systems Interconnection
PICS	Protocol Implementation Conformance Statement
POCSAG	Post Office Code Standard Advisory Group
PSTN	Public Switched Telephone Network
PW	Private Wire
R	Router
RAS	Resource Availability System
UA	User Agent
USWR	Unique System-Wide Reference
WAN	Wide Area Network

Glossary

Tenderer	A company which has been invited to submit a bid in response to this specification.
Contents	The part of a message that contains user information, eg a turn-out instruction.
Contractor	A company which is selected to offer products under the Standing Offer.
Envelope	The part of a message that contains information relevant only to the transfer of the message across the network.
Frame	The basic unit of information transferred between MTAs, comprising the message itself together with any overheads associated with the MTA-MTA protocol.
Message	The basic unit of communication across the network, comprising two parts, namely envelope and contents.
MTA	Message transfer agents (MTAs) act in pairs to transfer messages across a bearer.
Outstation	Used to mean either a fire station or an appliance.
UA	User agents (UAs) are the ultimate source or destination of all user messages.
<op_map>	Triangular brackets <> are used to denote a defined data type, field type or message type.
“incident”	Quotation marks “” are used to denote a data value or field value.
timeout	Italics are used to denote a defined parameter.

1 Introduction

1.1 General

1.1.1 This document is Volume A of a specification for the communications elements of Fire Service Mobilising Systems. It contains information which is common to more than one element of the sub-system, including definition of the communications protocol to be used.

1.1.2 This volume should be read in conjunction with Volumes B, C and D, which are the specifications for the control room, fire station and appliance equipments respectively.

1.1.3 Background information is provided in a separate document[1] which defines the user requirements and overall approach to replacement of the Fire Service Mobilising Systems.

1.2 Scope of this specification

1.2.1 To ensure that elements procured from different qualified Contractors are interoperable, it is clearly crucial that this specification defines the details of the protocol to be used for communications between the various elements and the minimum functionality of the individual elements.

1.2.2 Subject to this requirement for interoperability, the specification aims to allow Contractors a 'free hand' in the implementation of the standard products offered under the Standing Offer arrangement. The majority of the requirements stated in this specification do not therefore constrain Contractors in terms of implementation.

1.2.3 At the same time, a functional system model has been defined and is described in section 2.3. The purpose of this model is to provide a framework for a clear and unambiguous definition of the required functionality and operation of the various communications elements. Contractors are not obliged to adhere to this system model in their implementation, provided that their chosen solution is compliant with the requirements specified.

1.2.4 Finally, it is worth noting that, for some of the communications bearers (eg analogue private wires, the PSTN), international standards exist for the 'lower level' data link mechanisms used to transfer data across the bearer, eg modulation scheme, data rate etc. In these cases, the definition of the communications protocol in this specification references these standards.

1.2.5 However, for other bearers (eg private or public mobile data, paging schemes), no 'open' standards exist and the mechanisms employed by a given Contractor are generally proprietary to that Contractor. In these cases, this specification does not define the data link mechanisms to be used, but rather defines an industry standard interface that must be presented by Contractors at each end of the link. Many Brigades are currently interested in the use of mobile data for mobilising, and the

approach adopted in this area is discussed further in section 2.5.

1.3 Structure of Tenderer's response

1.3.1 Tenderers shall provide a separate response for each item of equipment offered for inclusion within the Standing Offer. Where appropriate, Tenderers may group information common to more than one response in a separate document. However, there must be no ambiguity as to whether a particular statement in the Tenderer's response refers to all or part of the equipment being proposed.

1.3.2 Tenderers shall describe in full the solution proposed to meet the specifications. The rationale behind solutions shall be described and any assumptions made shall be stated.

1.4 Statement of Compliance

1.4.1 Where the word 'shall' is used in these specifications, the requirement is mandatory and Tenderers must comply in full with the associated requirement. Requirements prefaced by the word 'should' are to be regarded as desirable options. If meeting a desirable requirement will significantly increase the price of the product, Tenderers should offer the additional requirement as a costed option.

1.4.2 Tenders shall include an individual response to each numbered paragraph and list item in this system specification and in the relevant equipment specification(s). In response to paragraphs specifying requirements, this shall include one of the following:

- 'compliant';
- 'partially compliant', in which case additional explanation shall be given;
- 'non-compliant'.

1.4.3 Any failure to comment will be construed as a failure to comply.

1.4.4 Where paragraphs specify that information is to be supplied by the Tenderer, the response to the paragraph shall include the information or a precise reference to its location elsewhere in the tender document.

1.4.5 In response to paragraphs which supply information and do not specify a requirement, the response shall be 'understood'.

1 Introduction

1.5 Contents of this document

1.5.1 Section 2 identifies the key objectives which have impacted upon the system specification and then explains the system model which has been developed.

1.5.2 Section 3 describes the key aspects about the protocol in terms of the convention for addressing messages, the general structure of messages and the general principles for acknowledging messages.

1.5.3 Section 4 defines the functions associated with the router (the central element of a node within the sample system model).

1.5.4 Section 5 defines the functions associated with message transfer agents which are responsible for transferring a message between a pair of nodes.

1.5.5 Section 6 defines the user agents which are associated with interfacing end-user devices to the network (eg printer, alerter etc).

1.5.6 Section 7 defines general requirements applying to design, manufacturing, safety and installation and commissioning.

1.5.7 Section 8 defines the documentation and training requirements.

1.5.8 Appendix A defines the data entities which are used within messages.

1.5.9 Appendix B defines the message set.

1.5.10 Appendix C summarises the parameters associated with each part of the system.

1.5.11 Appendix D summarises the reason codes which are returned within error messages.

1.5.12 Appendix E defines the standard serial port.

1.5.13 Appendix F provides a comparison with the ISO OSI 7 layer model for data communications.

2 System design concepts

2.1 Introduction

2.1.1 This section provides a description of the various design concepts that lie behind the requirements stated in this specification. Accordingly, the section does not itself contain any requirements statements, but is provided for Tenderers' information only.

2.1.2 The objectives behind the approach and system model are discussed in section 2.2.

2.1.3 The proposed system model for the complete communications infrastructure is presented in section 2.3

2.1.4 An analogy with an X.400 message handling system is discussed in section 2.4. It should be emphasised that the analogy relates only to the concept of a message passing system. The mobilising communications infrastructure is required to provide 'near instantaneous' or 'real time' delivery of messages, whereas X.400 systems generally are not.

2.1.5 For economic and operational reasons, many Brigades are currently interested in radio mobilising, ie the use of a mobile data system to mobilise appliances or fire stations via radio. However, the lack of suitable open standards for mobile data precludes the standard definition of such systems in the Standing Offer. The approach adopted for mobile data is discussed in section 2.5.

2.1.5 For various reasons, a direct comparison with the ISO OSI 7 layer model is not particularly illuminating in this case. However, such a comparison is included in appendix F for the sake of completeness.

2.2 Objectives

2.2.1 Before considering the model in detail, it is worth defining the objectives which had to be met. These objectives were:

- that the specification be suitable for products to be procured under a Standing Offer;
- that products had to be interoperable with other products from the same, or different, Contractors;
- that the Standing Offer should attempt to meet the need for provision of equipment with the differing capacities and performance required by different Brigades without imposing a cost premium on any one Brigade;
- that Contractors and Brigades should be able to enhance the basic products without affecting interoperability;

- that the specification should also be useful to Brigades intending to procure equipment outside the Standing Offer.

2.3 System model

2.3.1 In considering a complete Fire Service mobilising communications infrastructure, three separate elements may be identified:

- control room communications sub-systems;
- fire station equipments;
- appliance equipments.

These elements are illustrated graphically in figures 5, 6 and 7 of reference 1.

2.3.2 In terms of communications functionality, these three elements are very largely identical, and may all be regarded as nodes in a communications network. Within the proposed model they are regarded as peers.

2.3.3 The simplest arrangement of nodes will be a star centred upon the communications processor (see figure 2-1).

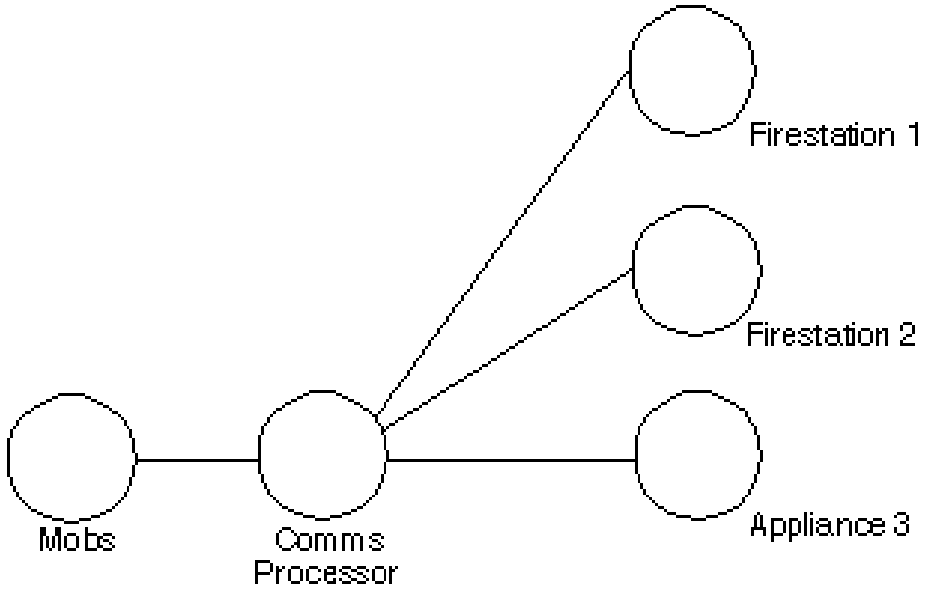

Figure 2-1
Simple network architecture

2.3.4 The architecture is also capable of supporting more complex arrangements. For

example:

- the communications controller may be split into two physical devices to provide resilience;

- the communications controller in one Brigade can be connected to the communications controller in another Brigade to allow the exchange of messages between mobilisation systems or to allow the direct mobilisation of another Brigade's outstations;

- fire stations may be cascaded to minimise the costs of leased lines.

With the more complex networks, the proposed architecture does not guarantee that a message will automatically find all alternative routes in the event of a failure. However, it is still possible to provide resilience if this is required. An example of a more complex network is given in figure 2-2.

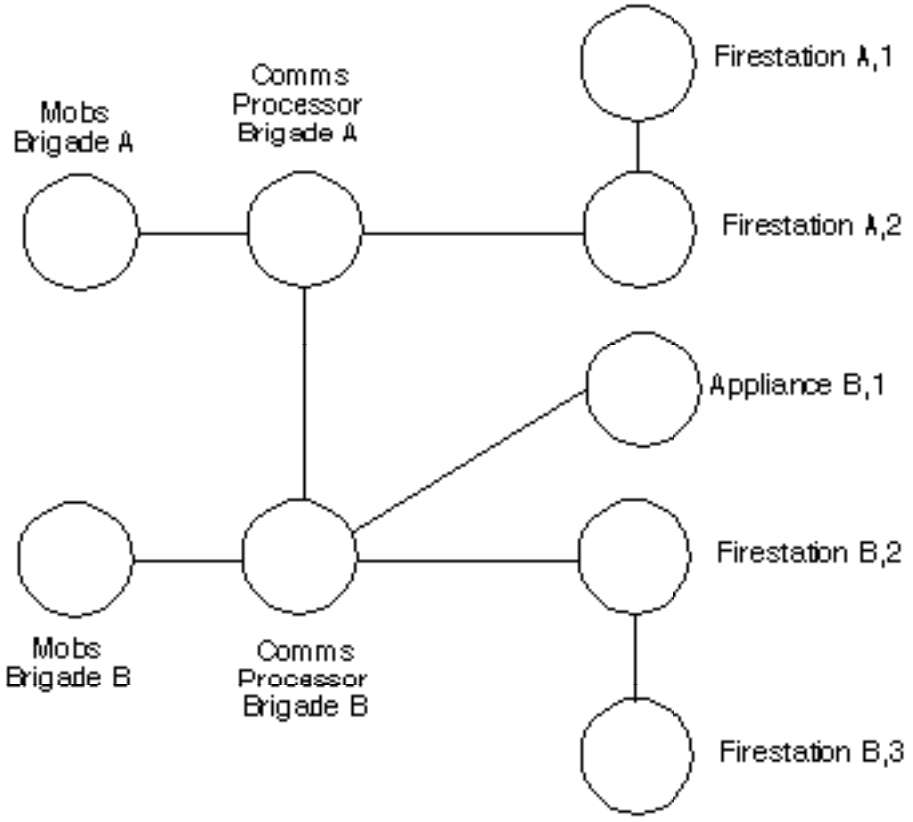

Figure 2-2
An example of a more complex network

2.3.5 Each node may in turn be modelled by a common functional architecture comprising:

- various user agents (UAs) which are concerned with the delivery and receipt of messages to/from the 'end user' application devices connected to the communications network, eg a mobs system or a fire station turn-out printer;

- various message transfer agents (MTAs) which operate in pairs to reliably transfer messages across a given communications bearer, eg across an analogue private wire;

- a router, which is concerned with the orderly switching of messages between MTAs and UAs.

2.3.6 This functional model of a communications node is best understood by reference to the complete system. Figure 2-3 illustrates a small part of a typical Brigade arrangement. From the diagram it can be seen that:

- the basic model of each of the three items of equipment is identical, and comprises a router (R) linking various UAs and MTAs;

- all items of equipment behave as peers;

- MTAs always talk to MTAs;

- UAs and MTAs can be either internal or external to the comms system.

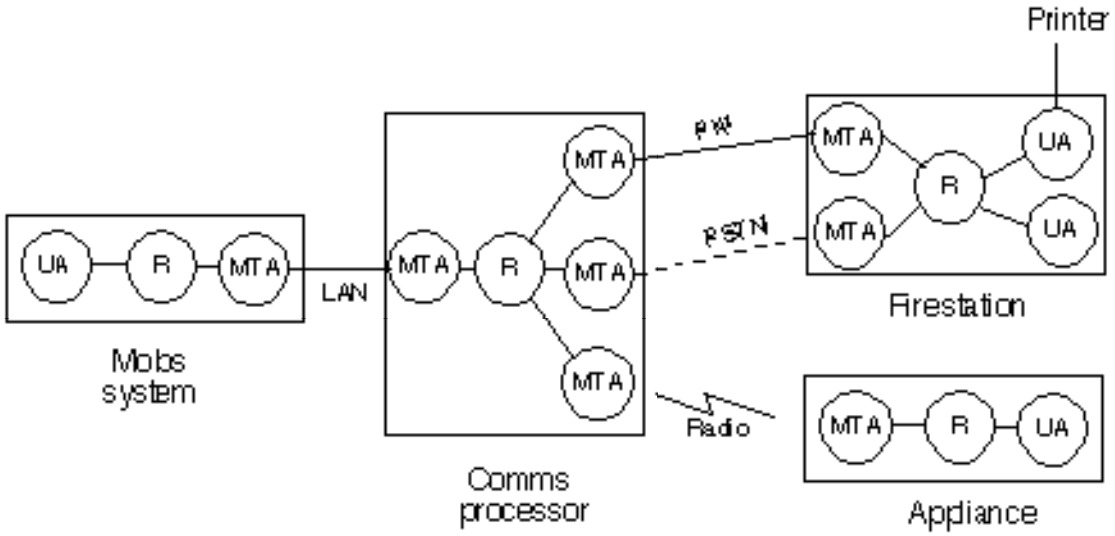

Figure 2-3
Functional system model

2.3.7 A complete Brigade communications system may therefore be modelled as an ensemble of functionally identical message handling nodes operating a 'store and forward' message transfer system (MTS). The elements described above may then be divided into two logical types of component:

- The routers and MTAs are communications entities (CEs) and are primarily concerned with the transfer of messages across the communications network, and are not concerned with the content (or semantics) of messages (with the exception of certain messages required for network management purposes).

- UAs (eg mobs systems, turn-out printers, admin terminals, mobile data units, etc) are not concerned with the mechanisms used to transfer information across the network but are concerned solely with the content (or semantics) of messages (with the exception that they shall be able to verify the source, destination and sequence of messages).

2.3.8 It should also be noted that because the protocol between UAs is independent of which bearer is being used, there are no constraints on the choice of bearer(s) between, for example, the mobilisation system and the communications system.

2.3.9 It is therefore possible to divide each message into an 'envelope' and a 'contents', where the envelope contains only that information pertinent to the transfer of a message across the network, eg the identity of the sender and of the recipient, the message priority, etc. Thus, CEs transfer messages on the basis of the envelope only and pass the contents 'transparently'.

2.3.10 The definition of message contents in this specification is intended primarily to support operational mobilising messages and the various administrative messages relevant to mobilisation, eg duty manning updates, incident log updates etc. However, the specification provides a framework within which future, as yet undefined, message content formats may be accommodated. By virtue of these messages being carried within a common envelope, it will be possible to transport these messages across existing communications systems without them being upgraded.

2.3.11 In specifying the operation of the communications protocol to be used, therefore, it is helpful to focus on the concept of a message as comprising an 'envelope' and a 'contents', and to examine the operations carried out by the various CEs and UAs in processing these separate components. This split is reflected within the remainder of this document.

2.4 Comparison with an X.400 message handling system

2.4.1 The 'store and forward' message transfer model described above is analogous to a 'stripped down' version of the internationally standardised X.400 message handling system (MHS) protocol.

2.4.2 While this analogy is a useful one, it should be emphasised that it relates only to the concept of a message passing system. The mobilising communications infrastructure is required to provide 'near instantaneous' or 'real time' delivery of messages, whereas X.400 systems generally are not.

2.4.3 Under this analogy, the following equivalences may be identified:

- the router and MTAs are effectively equivalent to X.400 message transfer agents (MTAs), and the ensemble of CEs is equivalent to the X.400 message transfer system (MTS);

- UAs are effectively equivalent to X.400 user agents (UAs).

2.4.4 As in the system model described in section 2.3, an important concept in X.400 is the division of each message into an 'envelope' and a 'contents', whereby MTAs transfer messages on the basis of the envelope only and pass the contents 'transparently'.

2.4.5 Similarly, the division of protocol into application protocol and message transfer protocol described above is analogous to the distinction between the P2 (IPMS) and P1 protocols in X.400.

2.5 Approach to mobile data

2.5.1 For economic and operational reasons, many Brigades are currently interested in using mobile data to mobilise appliances or fire stations via radio. However, no suitable open standards for mobile data exist at present and the data link mechanisms employed by a given Contractor are invariably proprietary to that Contractor. Consequently:

- such systems cannot be included as a standard product under the Standing Offer (and this specification does not define the 'on-air' data link mechanisms to be used);

- where Brigades decide to employ mobile data, the goal of full on-air interoperability between different Contractors will not in general be achieved.

2.5.2 At the same time, it is important that those Brigades wishing to employ mobile data are still able to exploit the advantages offered by the procurement of systems under the Standing Offer. Accordingly, the approach has been to define an industry standard interface that must be presented by Contractors at each end of the mobile data link, and this interface is fully defined in this specification.

2.5.3 The chief motivation for this approach is that it allows Contractors who are capable of offering mobile data-based equipment to put forward this equipment for qualification under the Standing Offer. Indeed, such Contractors are encouraged to offer a mobile data system as a separately costed option in their response to this

specification.

2.5.4 In relation to mobilisation by radio, it is noted that, according to current regulations, the mobile data system must ensure that a readily-understandable channel identification callsign of 2 alphanumeric characters is transmitted periodically.

3 General protocol issues

3.1 Introduction

3.1.1 This section defines the top-level issues associated with the protocol. These are:

- the addressing scheme used to identify the different elements of the system;
- the message envelope which defines the parameters used by the message transfer protocol;
- the circumstances under which the message transfer system provides an acknowledgement to the UA;
- the general principles for message acknowledgement between UAs;
- message size.

3.2 Addressing conventions

3.2.1 To form an effective means of interoperation between Brigades, it is imperative that all systems use a standard address format. Furthermore, the address of each communications element must be unique within the UK.

3.2.2 The format of an address shall therefore include three parts, as follows:

- a Brigade or agency number in the range 0 to 255;
- a node number in the range 0 to 1023;
- a port number in the range 0 to 63.

The format and encoding is defined formally as <comms_address> in appendix A.

3.2.3 Each Brigade or agency will be issued with a unique number. These numbers will be allocated by the Home Office. This number will form the first part of the address of all nodes within a Brigade.

3.2.4 Each node within a Brigade (ie each fire station or appliance) will also have a unique node number which will be allocated by the Brigade and notified to the Contractor. External systems (such as a mobs system) connected to the communications network will also be allocated a unique node number.

3.2.5 Finally, each port (ie MTA, UA or router) will have a unique number within a node which shall be allocated by the Contractor when configuring equipment for a particular Brigade. The use of these port numbers will be as follows:

- the vast majority of messages will be addressed to UAs, eg a mobilisation text message sent to a printer UA, a mobilise command message sent to a peripheral UA;

- various network management messages will be addressed to any port (ie UA, MTA or router), eg to set the parameters associated with that port;

- by convention, the router (see section 4) shall always be port address 0.

3.3 Message envelope definition

3.3.1 Messages shall be passed around the system with a standard envelope which includes all of, and only, the information which is needed to maintain the integrity of messages and route them around the system.

3.3.2 The envelope shall comprise:

- <source>, which defines the unique address of the source of the message;

- <dest_count>, which specifies the number of destination addresses which are included in the envelope (minimum 1, maximum 63);

- <destination>, which defines <dest_count> unique addresses;

- <priority>, which defines the priority of the message (minimum 1, maximum 9) with priority 1 being the highest priority;

- <ack_req>, which indicates whether the UA sending the message expects an acknowledgement (which may come from the destination UA or from the MTS);

- <seq>, which is a sequence number used to identify messages (see paragraph 3.3.4);

- <message_type>, which defines the type of the content of the message;

- <length>, which is the number of bytes in the <message> field;

- <message>, which is the message or contents of the envelope;

- <prot_vers>, which defines the protocol version being used by the source UA. Version 1 , except Page Officer message which is version 2.

- <bcc>, which is a block check character which is used to confirm the end-to-end integrity of the message. It is calculated by exclusive ORing of all of the preceding bytes within the envelope.

3.3.3 The format and encoding is defined formally in appendix A as <envelope>.

3.3.4 The combination of source address, destination address and sequence number will provide a unique system-wide reference (USWR) for each message. For messages where an acknowledgement is expected, the sequence number shall be used by the originator of a message to uniquely identify each unacknowledged message (ie after having transmitted one message/frame with a particular sequence number, it shall not transmit another message/frame to any destination with the same sequence number until the first message has been fully acknowledged).

3.3.5 Long messages are transmitted in multiple blocks. Each block in a multiple-block message shall have a **different sequence number**.

3.3.6 The USWR is then used to identify to the sender of a message:

- that a particular message could not be delivered;
- which message requesting information a particular response refers to;
- that a particular message has been received and actioned.

3.3.7 The implications of this are:

- at any given time, the originator of a message could have up to 32,767 messages (each of which could be to multiple destinations) which had not been fully acknowledged;
- there is no reason why the frames originating from a source address need have successive sequence numbers;
- there is no reason why the frames arriving at a UA will have successive sequence numbers (and it is unlikely that they will);
- the recipient of a message is unable to tell from the envelope whether a message has been missed. Hence, if order of receipt is important, and the message content does not provide the information to allow messages to be re-ordered, the sender shall not transmit subsequent messages in a sequence until the previous message has been acknowledged;
- a message addressed to multiple destinations is equivalent to multiple singly addressed messages in terms of message identification and acknowledgement.

3.3.8 When a node is upgraded to a higher protocol, it shall send only those messages altered by the protocol, with the higher protocol version. All other messages shall be sent with their previous protocol version.

3.4 MTS-UA acknowledgements

3.4.1 The general procedures for message transfer and acknowledgement shall follow the philosophy of an X.400-like store and forward message transfer system (MTS), as follows:

- The message transfer protocol shall operate in a connection-less mode, whereby a message is passed from node to node until it reaches its destination. An end-to-end connection shall not be present at any stage of the message's progress.

- Each node shall handle messages in an autonomous fashion and shall forward messages towards their ultimate destination according to the addressing scheme. The routing algorithm employed is discussed further in section 4.3.

- The acknowledgement scheme used by the MTS (as opposed to the application level acknowledgements between cooperating UAs) shall be based upon the assumption of successful delivery. In other words, a message that reaches its destination shall generate no acknowledgements from the MTS to the source UA. Acknowledgements (indicating failure) shall be generated only when it is not possible to deliver a message for which <ack_req> was set, ie an acknowledgement was expected.

3.4.2 However, it should be emphasised that the mobilising communications infrastructure is required to provide 'near instantaneous' or 'real time' delivery of messages, whereas X.400 systems generally are not.

3.4.3 The acknowledgement that a message cannot be delivered will be generated by the router (see section 4). This in turn will rely on information from MTAs as to whether a message has been successfully transferred (see section 5).

3.5 UA-UA acknowledgements

3.5.1 A simple protocol shall be used between co-operating UAs so that a UA initiating a transaction knows that it has been completed.

3.5.2 The mechanism depends upon the nature of the transaction which can either be:

- a request from one UA for another to do something;

- a request from one UA to get some information from another UA.

3.5.3 In general, the protocol is organised such that, by setting <ack_req> in the initial message, a UA initiating a transaction will always get some form of (automatic) acknowledgement. This acknowledgement will be one of the following (where different forms of <NAK> message are indicated by the reason codes given in appendix D):

- an acknowledgement (<NAK>) from the MTS to indicate that a message could not be delivered;

- an acknowledgement from the remote UA (<NAK>) to indicate that a message has been received but could not be processed;

- an acknowledgement from the remote UA (<NAK>) to indicate that a message has been received, the required action has been initiated, and that a further acknowledgement will follow when it is complete (eg when a message has been manually acknowledged);

- an acknowledgement (<ACK>) from the remote UA to indicate that the requested action has been completed;

- the information requested from the remote UA.

3.5.4 After submission of a message to the MTS, the sending UA shall implement a timeout mechanism as a 'last ditch' safeguard to cater for the event of no acknowledgement being received from the remote UA or the MTS. The timeout period shall be *no_ack_timeout* as defined in the router parameter table in appendix C.5.

3.5.5 If the timer times out, the message shall be sent again with the same envelope as before. After *retries* attempts, the message shall be removed from the unacknowledged message table. Other action is undefined but will generally result in an alarm to an operator.

3.5.6 Note that the option of a message being manual acknowledged is included for certain mobilising messages. The acknowledgements associated with these messages are discussed in paragraph 6.2.4.10.

3.6 Message size

3.6.1 It is necessary to consider the optimum maximum message size (ie the maximum value of the length parameter within the envelope definition) for messages used to convey information of arbitrary length (eg a computer file). The reasons for this are:

- to allow the interruption of long, low priority messages for the transmission of higher priority messages;

- to simplify implementation.

3.6.2 Accordingly, long unformatted messages shall be transmitted as a sequence of shorter messages.

3.6.3 The optimum length of each message is a compromise since:

- a small message size increases the protocol overhead for long messages;

- a large message size implies that a bearer could be occupied for a relatively long period and could cause an unacceptable delay before a higher priority message could be sent.

3.6.4 The optimum value is consequently dependent upon the speed of the bearers used by Brigades and hence is a user defined parameter *max_length*. This is defined in the router parameter table in appendix C.5.

3.6.5 The maximum value of the *max_length* parameter is set to be 1,023 bytes. To ensure interoperability between systems operating with different values of *max_length*, all systems shall be capable of carrying messages of up to 1,023 bytes.

4 Router

4.1 Introduction

4.1.1 Conceptually, each node of the network contains a single router which binds the MTAs together. The functions of this router are described below.

4.1.2 The routing function is internal to a node and 'the router' is primarily a convenient concept for defining a set of functions which are common to each node. Contractors are free to choose whether they implement the router as a separate software module or whether the required functionality is incorporated within the MTAs.

4.1.3 This section describes those functions which are associated with the router at each node. These are:

- message transfer between MTAs and UAs;
- the routing algorithm for selecting the optimum route;
- procedures for access control and the maintenance of node parameters.

4.2 Message transfer

4.2.1 In outline, the router shall:

- receive messages from MTAs and UAs;
- pass each message to the appropriate UA if that UA is present at the local node;
- if the message is destined for a UA which is not present at the local node, then the router shall determine which is the most appropriate MTA to transfer the message;
- in the event that the MTA first selected is unable to transfer the message, the router shall select another MTA and pass the message to it, and so on.

4.2.2 The router shall be activated on receipt of a message from either a UA or a MTA. The router shall check that the frame format and block check character are valid for all messages received.

4.2.3 On receiving a message, the router shall decide if the message is for its node or for another node. If it is for the local node, it shall be passed to the appropriate UA immediately. If it is for another node, then it must be routed according to the constraints given below.

4.2.4 The router shall ensure that a message is never delayed by a message with a lower priority.

4.2.5 The router may achieve this by first checking whether there are any outstanding messages of a higher priority. If there are, the router shall queue the lower priority messages until all the higher priority messages have been sent.

4.2.6 The router shall ensure that messages of the same priority are processed in order of arrival at the node.

4.2.7 The router shall apply the routing algorithm (discussed in section 4.3) to each of the destination addresses attached to the message.

4.2.8 If a message is directed to a number of different destination addresses, which are to be reached by different bearers, then the router shall replicate the message for each bearer. Each destination address shall only appear on one of the messages.

4.2.9 The performance of the router shall be limited by the availability of bearers and shall transmit on multiple bearers simultaneously.

4.2.10 Each message shall then be passed to the appropriate bearer MTA. The router shall keep a copy of the message until the bearer MTA declares that it has been successfully forwarded. If the MTA is unable to send the message, then the router shall re-apply the routing algorithm to find the next best route and shall send the message to the appropriate bearer MTA. (Note that this interaction between the MTA and the router is internal to the node and has nothing to do with application level acknowledgement messages.)

4.2.11 In the event that there is no alternative route, the router shall determine whether the message which cannot be sent requires an acknowledgement (as determined by the <ack_req> field within the envelope). Then:

- the router shall discard the message if it does not require an explicit acknowledgement;

- the router shall send a <NAK> if the message does require an explicit acknowledgement. The format shall be as described in appendices A and D with:

    - the <reason_code_set> set to “general” (appendix A);
    - the <reason_code> set to “no_bearer” (appendix D).

4.3 Routing algorithm

4.3.1 The router shall route messages with minimum delay according to:

- a table which defines the preferred routes and bearers to reach each destination;

- the priority of the message and hence whether it is allowed to use certain

bearers;

- the availability of bearers.

4.3.2 For each (adjacent) node to which the router is connected, the routing table contains the following information:

- <next_node>, which gives the address of the adjacent node;

- <destination_nodes>, which lists a set of destination addresses of the nodes available from the adjacent node. Note that the port number field in the address is irrelevant for the purposes of node - node routing;

- <agent_type>, which defines the type of the bearer to be used;

- <preference>, which defines the order in which the bearers and adjacent nodes offering the same <destination_address> are to be tried. An entry with <preference> equal to "1" is to be tried before an entry with <preference> equal to "2" etc.

Note that there will in general be multiple entries in the routing table for each adjacent node, eg corresponding to different bearer types.

4.3.3 The routing table can most easily be considered as indexed by the set of adjacent nodes (and the bearers available to each), as described above. However, it may be more efficient for the purposes of message routing to re-structure the table so that it is indexed by destination addresses.

4.3.4 Functionally, the algorithm should operate as defined below:

(a) the router shall look up the <destination_address> of the message in the routing table and identify those entries of the table which apply to that address;

(b) it shall then take the entry which is most preferable (as defined by <preference>) and identify the <next_node> to be visited and the required <agent_type>;

(c) if the route frome <next_node> to <destination_address> has been recorded by the router as having been disabled (see section B.32.3), the router shall proceed to stage (g);

(d) it shall then look at each MTA of the required <agent_type> to find those which could carry the message (see paragraph 4.3.5);

(e) if multiple MTAs meet these criteria, the router shall pick the MTA which will deliver the complete message to the next node in the shortest time;

(f) if no MTAs meet the criteria, the router shall proceed to stage (g);

(g) the router shall select the next most preferential router table entry and go back to stage (c);

(h) if there are no more routing table entries to try, the router shall respond as defined in paragraph 4.2.11.

4.3.5 An MTA can carry a message if:

- the <next_nodes> parameter for the MTA includes the required destination node and the destination node is marked as being reachable;
- the status of the MTA is on-line or idle;
- the MTA can carry messages of the required priority.

In the case of PSTN or ISDN MTAs, the list of <next_nodes> is the relevant PSTN table. The preferred MTA will be one which is already on-line to the required <next_node> (assuming priority is acceptable).

4.3.6 If Tenderers consider that the proposed algorithm is not optimum, they may propose an alternative algorithm. However, all Contractors will be required to base their algorithms on the same set of input parameters.

4.3.7 The routing table shall be modifiable using the <set_parameter> messages, although it will not be changed frequently. More frequent changes in routing (eg to cope with bearer failure etc) will be achieved by changing the status of bearers.

4.4 Access control and parameter maintenance

4.4.1 The router shall support access control and parameter maintenance. As explained in appendix C, each parameter has a password level associated with it which determines who may modify that parameter. The complete set of parameters maintained at each node is listed in the tables in appendix C.5 et seq, which show the parameters associated with a router, with MTAs and with UAs.

4.4.2 The process of ‘logging on’ to supply a password needs to be performed once for each node where parameters are to be changed, and is achieved by sending a <set_parameter> message to the router port at that node, as described in paragraph 4.4.4.

4.4.3 The process by which parameters are changed once ‘logged on’ is described in appendix B for the <set_parameter> message. In broad terms, this is achieved by

sending a <set_parameter> message to the relevant port at that node, eg the asynchronous MTA port.

4.4.4 To 'log on' to a node, the remote address attempts to modify the *current_password* parameter in the router current parameter table. On receiving a <set_parameter> message, the router shall verify that the password supplied is the correct password for the level of access being requested. If it is, then an <ACK> shall be returned and the *current_password* will be set according to the fields within the <set_parameter> message. If the password is incorrect, a <NAK> shall be returned with a <reason_code_set> equal to “parameter” and a <reason_code> equal to “invalid_password”.

4.4.5 The current password parameter is also used to 'log off' a node. Setting the *current_password* parameter in the current parameter table with the level set to 0 will prevent any further parameters being modified. In this instance, the router shall ignore the password field and address field within the <set_parameter> command. An <ACK> shall be returned.

4.4.6 The passwords associated with different levels of access can be changed like any other parameter using the <set_parameter> message. Reading a parameter containing a password shall return the password as the string value “PASSWORD”.

4.5 Route disabling

4.5.1 If the router receives an MTA status change message (message 66) indicating that the status of a local MTA, her called the disabled MTA, has changed to “off-line (user initiated)” or “off-line (fault”, the router shall respond according to the following algorithm:

(a) the router shall determine, according to the routing algorithm detailed in section 4.3, all those <destination_nodes> for which there is no longer a viable route;

(b) if there are no such <destination_nodes>, the router shall takne no further action;

(c) if one or more <destination_nodes> are no longer reachable then the router shall respond as defined in paragraph 4.5.2.

4.5.2 If one or more <destination_nodes> become unreachable according to the algorithm outlined in paragraph 4.5.1, the router shall, for each adjacent node that is currently reachable, send a route status message (message 67) to the node, indicating all those <destination_nodes> that are now unreachable.

4.5.3 If the router receives an MTA status change message (message 66) indicating that the status of a local MTA, here called the enabled MTA, has changed to “on-line” or “idle”, the router shall respond according to the following algorithm:

(a) the router shall determine, according to the routing algorithm detailed in section 4.3, all those <destination_nodes> which are reachable via the enabled MTA;

(b) it shall then, for each adjacent node that is currently reachable, send a route status message (message 67) to the node, indicating all those destination nodes that are reachable via the enabled MTA.

4.5.4 When a node adjacent to the router has previously been unreachable, but becomes reachable once again, the router shall respond according to the following algorithm:

(a) the router shall determine, according to the routing algorithm detailed in section 4.3, all those <destination_nodes> for which there is no longer a viable route;

(b) if there ar no such <destination_nodes>, the router shall proceed to stage (d);

(c) if one or more <destination_nodes> are no longer reachable then the router shall send a route status message (message 67) to the newly reachable node, indicating all those <destination_nodes> which have become unreachable whilst the newly reachable node was itself unreachable;

(d) the router shall determine, according to the routing algorithm detailed in section 4.3, all those <destination_nodes> for which there remains a viable route;

(e) if there ar no such <destination_nodes>, the router shall take no further action;

(f) if one ore more <destination_nodes> are reachable then the router shall send a route status message (message 67) to the newly reachable node, indicating all those <destination_nodes> which have become reachable whilst the the newly reachable node was itself unreachable.

5 Message transfer agents

5.1 Introduction

5.1.1 The primary function of message transfer agents (MTAs) is to act in pairs to transfer messages across a bearer. The units of information transferred between MTAs are referred to as frames, and comprise the messages themselves (ie envelope plus contents), together with any overhead information required to co-ordinate the communication between MTAs. This MTA-MTA protocol is required to:

- ensure that the integrity of messages is maintained;
- ensure that the sending MTA knows whether the message has been received by the remote MTA;
- regularly validate the operation of the link.

5.1.2 In general, the MTAs transfer the message independently of the message type. However, for network management functions, MTAs can also be the source or destination of messages.

5.1.3 Each MTA shall be allocated its own port number and hence each port (ie leased line, PSTN line, radio modem etc) is uniquely identifiable (see section 3.2). Conceptually, a node may consequently have multiple instances of the same type of MTA, although it is up to Contractors whether the implementation reflects this conceptual view.

5.1.4 Section 5.2 first describes those parameters which are common to all MTAs. The following sections describe the functions which are supported by different types of MTA:

- private wire MTA (section 5.3);
- PSTN MTA (section 5.4);
- WAN MTA (section 5.5);
- asynchronous MTA (section 5.6);
- LAN MTA (section 5.7);
- ISDN MTA (section 5.8).

5.1.5 Note that the asynchronous MTA (and to a lesser extent the other MTAs such as the LAN MTA or the WAN MTA) provides a general purpose mechanism for interfacing to other equipment or communications services, such as:

- a mobile data communications services (whether public or private);

- administration or management information system (MIS) terminals;

- network administrator or engineering terminals.

5.1.6 Public Radio Data Networks (e.g. PAKNET, RAM etc). Where provided, the MTA should operate in an identical manner to a PSTN modem, i.e. initialising and clearing a logical point to point link between two PAD's, all other radio MTA parameters act as their ISDN equivalents. The following have been identified as shown:

- each PAD is identified by a 14 digit NUA (Network User Address), section A;

- the NUA of remote PAD's are stored in the MDT_table and can be accessed as parameter 21, section A and C;

- agent _type, e.g. 19 PAKNET, section A.

5.2 MTA parameters

5.2.1 The parameters common to all MTAs are:

- *port_number,* which is the number of the port (1-63);

- *agent_type,* which defines the type of the port;

- *interface_status,* which defines the operational status of the port and is described below;

- *notify_status_changes,* which defines whether the MTA should send an <MTA_status_change> message when its status changes;

- *frame_tx,* which shall be incremented every time the MTA successfully sends a frame;

- *frame_rx,* which shall be incremented every time the MTA successfully receives a frame;

- *frame_tx_failures,* which shall be incremented every time the MTA fails to send a frame;

- *frame_rx_failures,* which shall be incremented every time the MTA receives an invalid frame;

- *priority,* which defines the priority of messages which may be carried by this MTA. A **single** digit priority defines the lowest level of priority of messages which can be carried by this MTA. If a **two** digit priority is used, the higher order digit defines the lowest level of priority, and the low order digit defines an

additional priority of messages which may be carried by this MTA;

- *next_nodes*, which defines which nodes can be reached directly from this node (see following).

5.2.2 These parameters shall be readable and modifiable by a remote UA (usually the network management station) using the <set_parameter> and <parameter_request> messages.

5.2.3 The *interface_status* parameter can take any of the following values:

- idle;
- on-line;
- off-line (user initiated);
- off-line (fault).

5.2.4 The “idle” state is only used by PSTN or ISDN MTAs which are operational but which do not have any calls established.

5.2.5 “On-line” is the usual status for most MTAs and indicates that the MTA is available to transfer messages. A PSTN or ISDN MTA which has a call established will also be set to “on-line”.

5.2.6 “Off_line (user_initiated)” identifies that a UA (usually the network management station) has set the MTA off-line. With this status, the interface shall not initiate or receive messages. Once set to this state, the MTA shall not change state unless the parameter is manually changed by another UA.

5.2.7 “Off-line (fault)” identifies that the MTA has detected a fault with the interface. When set to this state, the MTA shall not be passed any further messages by its local router. It shall still attempt to receive messages if they arrive from the remote end of the link. If the MTA subsequently establishes that the link is operational, it shall change the state to on_line.

5.2.8 A change from “off_line (fault)” to “on_line” (or vice-versa) shall cause a <MTA_status_change> message to be transmitted if *notify_status_changes* is set to TRUE.

5.2.9 The use of the *next_nodes* parameter varies slightly depending upon the nature of the MTA. For PW and radio, it is a static parameter. For PSTN and ISDN, it will identify which node the bearer is currently connected to.

5.3 Private wire MTA

5.3.1 Introduction

5.3.1.1 A pair of private wire MTAs shall support a point to point asynchronous connection between two nodes. A separate pair of dedicated modems shall be provided for each private wire.

5.3.1.2 The required functionality shall be provided with either:

- a modem which is an integral part of the node;
- a modem which is connected to the main part of the node through a serial link.

5.3.1.3 The modem should be within the confines of the node to minimise the chance of EMI. If the modem is external to the node, it shall be installed to minimise the chance of EMI corrupting data.

5.3.2 Private wire MTA functions

5.3.2.1 The private wire MTA shall support the following parameters in addition to those supported by all MTAs:

- *verify_period*, which is the maximum period which is allowed without artificially creating activity on the link (see paragraph 5.3.4.14);
- *verify_timeout*, which is the maximum period of inactivity which is allowed on the link before deciding that it is has failed;
- *frame_duration*, which is the maximum period between sending/receiving the first character of a frame and receiving the last character of a frame;
- *frame_timeout*, which is the maximum period from the sending of the last character of a frame to deciding that a frame acknowledgement is overdue;
- *retries_allowed*, which is the maximum number of times that an MTA shall attempt to send a message before deciding it has failed.

5.3.2.2 The MTA should support the <test> message to allow V.54 loop back tests to be performed remotely. It should support Loop 1, Loop 2 and Loop 3 tests.

5.3.3 Physical and electrical specification

5.3.3.1 Each private wire MTA shall include all components necessary to provide an

interface to a 2-wire leased line. The MTA shall include all wiring and equipment up to the customer side of the NTP.

5.3.3.2 The line modulation shall conform to CCITT V.32 recommendation with V.42 error correction.

5.3.3.3 The modems shall support CCITT V.54 recommendations.

5.3.4 MTA-MTA Protocol

5.3.4.1 A simple MTA-MTA protocol is required to guarantee frame level synchronisation between cooperating MTAs. The protocol between MTAs shall use the following frame structure and protocol messages:

- <SOH><envelope><EOT>;
- <ENQ>;
- <ACK>;

where:

- SOH is ASCII character 01_{hex};
- <envelope> is as defined in appendix A;
- <EOT> is ASCII character 04_{hex};
- <ENQ> is ASCII character 05_{hex};
- <ACK> is ASCII character 06_{hex}.

Receiving frames

5.3.4.2 When the MTA is waiting for a frame to arrive, it shall scan the incoming characters until an <SOH> or <ENQ> is detected.

5.3.4.3 If an <ENQ> is detected, it shall return an <ACK>.

5.3.4.4 If <SOH> is detected, it shall count the expected number of characters (as determined within the envelope) and verify that the next character is <EOT>. If so, it shall:

- pass the frame to the router;

- send back an <ACK> as soon as any frame which is already being transmitted on the return path is complete;

- increment the *frames_rx* counter.

5.3.4.5 If the frame is not valid, it shall be discarded and the *frame_rx_failures* counter shall be incremented.

5.3.4.6 The MTA shall start a timer on receiving the <SOH>. If this timer reaches *frame_duration* before the end of the frame is received, the MTA shall discard the characters received so far and wait for a new message. The *frame_rx_failures* counter shall be incremented.

5.3.4.7 The link shall operate full-duplex.

5.3.4.8 The MTA should verify the message has been received correctly by use of <bcc>. If an error has occurred the MTS should not return an <ACK> and the message should be retransmitted.

Transmitting frames

5.3.4.9 The MTA shall maintain a local variable which is the number of retries. This will initially be set to 0.

5.3.4.10 The MTA shall start a timer on transmitting the <SOH>. If this timer reaches *frame_duration* before the end of the frame is transmitted, it shall increment the *frame_tx_failures* counter and reset the link. The local retries counter shall then be incremented and if it is less than or equal to the *retries_allowed* parameter, this process shall be started again. If it is more than the permitted counter, the MTA shall fail as indicated in paragraph 5.3.4.12.

5.3.4.11 Having successfully transmitted all characters in a frame, the MTA shall wait for an <ACK> from the remote end. If it does not receive an <ACK> within *frame_timeout* seconds, it shall increment the *frame_tx_failures* counter and the local retries counter. If this is still less than or equal to the permitted number, the frame shall be retransmitted. If is more than the permitted counter, the MTA shall fail as indicated in paragraph 5.3.4.12.

5.3.4.12 If the frame is transmitted successfully and acknowledged, the *frame_tx_counter* shall be incremented.

5.3.4.13 If the MTA is still not successful, it shall discard the frame and inform the router that it was unable to complete the transmission.

Link verification

5.3.4.14 At all times, the MTA shall monitor the status of the line (CCITT V.24 circuit 109). If the line fails / recovers, the state of *interface_status* shall be changed appropriately. If the line fails, the MTA shall discard any partial frames which it is receiving / sending. If sending a frame, it shall inform the router that it was unable to complete the transmission.

5.3.4.15 To verify the end-to-end status of the line, the MTA shall maintain a timer which measures the interval since the last valid message was received. If this timer reaches the value of *verify_period* seconds, the MTA shall send an <ENQ> to the remote MTA. If the timer reaches *verify_timeout*, the MTA shall set the status of the interface to "off-line (fault)". If verify_timeout > 2 (verify_period) the MTA should poll each verify_period until verify_timeout.

5.3.4.16 If the MTA receives an <ENQ> frame, it shall send an <ACK> frame back to the remote MTA and reset its inactivity timer.

5.3.4.17 If the status of the MTA changes from "online"or "idle" to "off-line(user initiated)" or "off-line(fault)" the MTA shall immediately inform its local router of its current status using the MTA status change message (message 66).

5.3.4.18 If the status of the MTA changes from "off-line(user initiated)" or "off-line(fault)" to "online" or "idle" the MTA shall immediately inform its local router of its current status using the MTA status change message (message 66).

5.4 PSTN MTA

5.4.1 Introduction

5.4.1.1 A pair of PSTN MTAs shall support dialled connections across the PSTN using CCITT V.22 bis modems and V.42 error correction.

5.4.1.2 The required functionality shall be provided with either:

- a modem which is an integral part of the node;
- a modem which is connected to the main part of the node through a serial link.

5.4.1.3 The modem should be within the confines of the node to minimise the chance of EMI. If the modem is external to the node, it shall be installed to minimise the chance of EMI corrupting data.

5.4.2 PSTN MTA functions

5.4.2.1 The PSTN MTA shall support the following parameters in addition to those

supported by all MTAs:

- *frame_duration*, which is the maximum period between sending/receiving the first character of a frame and receiving the last character of a frame;
- *frame_timeout*, which is the maximum period from the sending of the last character of a frame to deciding that a frame acknowledgement is overdue;
- *retries_allowed*, which is the maximum number of times that an MTA shall attempt to send a message before deciding it has failed;
- *dial_tones*, which indicates whether the MTA should use tone dialling or loop disconnect dialling;
- *my_tel_no*, which is the telephone number of this MTA;
- *connect_stats*, which is a list of all calls made.

5.4.2.2 The MTA should support the <test> message to allow V.54 loop back tests to be performed remotely. It should support Loop 1, Loop 2 and Loop 3 tests.

5.4.3 Physical and electrical specification

5.4.3.1 Each PSTN MTA shall include all components necessary to provide an interface to a direct exchange line or an extension off a PABX. The connection point shall be a LJU.

5.4.3.2 The line modulation shall conform to CCITT V.22 bis recommendation with V.42 error correction, LAP-M.

5.4.3.3 The modems shall support CCITT V.54 recommendations.

5.4.3.4 Automatic calling and answering control should follow the CCITT V.25 bis recommendations.

5.4.3.5 Redial attempts shall be as defined in Pattern B of BS6789, part 3.1.

5.4.3.6 The modem shall be certified to fall back to lower speeds when line quality dictates.

5.4.4 MTA-MTA Protocol

5.4.4.1 The protocol between communicating PSTN MTAs shall be the same as for the PW link, except that on establishing a link, the calling party shall transmit the following message:

- <SOH> <telephone number> <EOT>.

This is discussed under link establishment.

Link establishment

5.4.4.2 When told to establish a link to a remote site, the MTA shall dial the appropriate telephone number.

5.4.4.3 On establishing the link, the MTA shall send an <SOH> <telephone number> <EOT> message containing the telephone number of the calling party. The called party shall look up the telephone number to find the node address of the calling party. If this operation is successful, the called party shall:

- return an <ACK> to the calling party;
- set the *next_nodes* parameter to address the range of ports at the calling node;
- set the *interface_status* to “online”.

5.4.4.4 On receipt of the <ACK>, the calling party shall set its *next_nodes* parameter to address the range of ports at the called node and set its *interface_status* to “online”.

5.4.4.5 If the operation is not successful or if the called party refuses the connection, the calling party shall clear down the link. Similarly, if the calling party does not receive an <ACK> within *frame_timeout* seconds, it shall terminate the link and notify the router that it could not forward the message.

5.4.4.6 The calling party shall record in *connect_stats* the telephone number dialled and the connect time for each call. It should record the time at which the call was made.

Receiving frames

5.4.4.7 The process of receiving a frame is the same as for a private wire MTA.

Transmitting frames

5.4.4.8 The process for transmitting a frame is the same as for a private wire MTA.

5.4.4.9 When the MTA which establishes the connection transmits a frame, it shall start a timer. Each time it transmits further frames or acknowledgements, it shall reset the counter.

5.4.4.10 If the timer reaches the value of *hold_time* (as supplied by the telephone table), the

MTA shall terminate the connection and set its *interface_status* to “idle”.

Link verification

5.4.4.11 While the circuit is established, the MTA shall monitor the status of the line (circuit 109). If the line fails / recovers, the state of *interface_status* shall be changed appropriately. If the line fails, the MTA shall discard any partial frames which it is receiving / sending. If sending a frame, it shall inform the router that it was unable to complete the transmission.

5.4.4.12 If the status of the MTA changes from “online” or “idle” to “off-line(user initiated)” or “off-line(fault)” the MTA shall immediately inform its local router of its current status using the MTA status change message (message 66).

5.4.4.13 If the status of the MTA changes from “off-line(user initiated)” or “off-line(fault)” to “online” or “idle” the MTA shall immediately inform its local router of its current status using the MTA status change message (message 66).

5.5 WAN MTA

5.5.1 Introduction

5.5.1.1 The WAN MTA is used to provide a high speed interface between equipment at the same or different sites. It shall conform to the GOSIP 4 WAN sub-profile.

5.5.1.2 In OSI terminology, the WAN MTA acts as an end-system (DTE). The WAN MTA may be directly connected to another DTE (DTE-DTE) or to an X.25 network (DTE-DCE).

5.5.2 GOSIP WAN Options

5.5.2.1 All mandatory options within the GOSIP 4 WAN subprofile shall be supported. The WAN MTA shall support the following GOSIP options:

- X.21 signalling at 64,000 bps;
- Connection-mode Network Service (CONS) at the Network Layer;
- Class 0 and Class 2 Transport classes;
- be capable of initiating and responding to both Class 0 and Class 2 connection

request Transport Protocol Data Units (TPDUs).

5.5.2.2 Tenderers who are offering a WAN MTA shall supply a GOSIP WAN procurement PICS (Protocol Implementation Conformance Statement) with their response.

5.5.3 WAN MTA functions

5.5.3.1 The WAN MTA shall support the following parameters in addition to those supported by all MTAs:

- *my_WAN_address*, which is the WAN address of this MTA;
- *connect_stats*, which is a list of all calls made.

5.5.3.2 For each node in the *WAN_addresses* table held by the router (see appendix C), the type of connection may be either Permanent Virtual Circuit (PVC) or Switched Virtual Circuit (SVC). Where a PVC is specified, the WAN MTA shall establish the connection as part of its initialisation routine upon power up, and the connection shall normally remain established indefinitely.

5.5.3.3 The WAN MTA shall also set up and tear down PVCs in response to network management messages, as follows:

- upon receipt of a message setting the status to “off-line”, an MTA that was previously “on-line” shall tear down all of the PVCs specified in the table and SVCs that are currently set up;
- upon receipt of a message setting the status to “on-line”, an MTA that was previously “off-line” shall set up all of the PVCs specified in the table.

5.5.3.4 Where an SVC is specified, the WAN MTA shall establish a connection to the node only upon demand, ie when requested to transfer a message to that node. Once the message has been transferred successfully to the destination node, the sending WAN MTA shall clear down the SVC.

5.6 Asynch MTA

5.6.1 Introduction

5.6.1.1 The asynch MTA provides a general purpose interface for connecting a comms system to other local intelligent equipment. This equipment may be a DTE (eg a building management system) or it may be more communications equipment (eg a

mobile data system).

5.6.2 Asynch MTA functions

5.6.2.1 The asynch MTA shall support the following parameters in addition to those supported by all MTAs:

- *verify_period*, which is the maximum period which is allowed without artificially creating activity on the link;
- *verify_timeout*, which is the maximum period of inactivity which is allowed on the link before deciding that it has failed;
- *frame_duration*, which is the maximum period between sending/receiving the first character of a frame and receiving the last character of a frame:
- *frame_timeout*, which is the maximum period from the sending of the last character of a frame to deciding that a frame acknowledgement is overdue;
- *retries_allowed*, which is the maximum number of times that an MTA shall attempt to send a message before deciding it has failed.

5.6.3 Physical and electrical specification

5.6.3.1 The UA shall include all equipment up to the port.

5.6.3.2 The port shall provide a V.24/V.28 interface.

5.6.3.3 The port shall be configured as a DTE.

5.6.3.4 Connectors should conform to ISO 2110 (25 pin 'D' type).

5.6.4 MTA-MTA Protocol

5.6.4.1 A simple MTA-MTA protocol is required to guarantee frame level synchronisation between cooperating MTAs. The protocol beteeen MTAs shall use the following frame structure and protocol messages:

- <SOH><envelope><EOT>;
- <ENQ>;

- <ACK_M>;
- <ACK_S>;

where:

- <SOH> is ASCII character 01_{hex};
- <envelope> is as defined in appendix A;
- <EOT> is ASCII character 04_{hex};
- <ENQ> is ASCII character 05_{hex}.

5.6.4.2 <ACK_M > and <ACK_S> represent acknowledgement characters 06_{hex} and 07_{hex} respectively. Given a particular MTA-MTA link, the MTA on one side of the link shall be allocated the acknowledgement character <ACK_M> for transmission and shall expect to receive <ACK_S>. Converseley, the MTA on the other end of the link shall be allocated <ACK_S> for transmission and shall expect to receive <ACK_M>. The MTA which transmits the <ACK_M> character will be referred to as the 'master' and the MTA which transmits the <ACK_S> character will be referred to as the 'slave'.

5.6.4.3 Which acknowledgement characters a particular MTA expects to transmit and receive shall be indicated by the MTA parameter *ack_type* (see table C.8). This parameter will consist of a single ASCII character as follows:

'M': the MTA (master) transmits the <ACK_M> acknowledgement character;

'S': the MTA (slave) transmits the <ACK_S> acknowledgement character;

'N': the MTA used the previous protocol scheme (as specified in GD-92 / 1003A / 2.1).

For the purpose of the remainder of section 5.6, the acknowledgement characters that a particular MTA expects to transmit and receive will be referred to a *ack_char_tx* and *ack_char_rx* respectively.

Receiving frames

5.6.4.4 When the MTA is awaiting for a frame to arrive, it shall scan the incoming characters until an <SOH> or <ENQ> is detected.

5.6.4.5 If an <ENQ> is detected, the MTA shall return its *ack_char_tx* acknowledgement

character.

5.6.4.6 If <SOH> is detected, it shall count the expected number of characters (as determined within the envelope) and verify that the next character is <EOT>. If so, it shall:

- pass the frame to the router;
- send back its *ack_char_tx* acknowledgement character as soon as any frame which is already being transmitted on the return path is complete;
- increment the *frames_rx* counter.

5.6.4.7 If the frame is not valid, it shall be discarded and the *frame_rx_failures* counter shall be incremented.

5.6.4.8 The MTA shall start a timer on receiving the <SOH>. If this timer reached *frame_duration* before the end of the frame is received, the MTA shall discard the characters received so far and wait for a new message. The *frame_rx_failures* counter shall be incremented.

5.6.4.9 The link shall operate full-duplex.

5.6.4.10 The MTA should verifty that the message has been received correctly by use of <bcc>. If an error has occurred, the MTA should not return its *ack_char_tx* acknowledgement character and the message should be retransmitted.

5.6.4.11 The MTA should compare any message which had been received correctly against those messages which it has recently transmitted but which have not yet been acknowledged. If the received message is identical to such a transmitted message, it should be discarded and an *ack_char_tx* acknowledgement character should not be returned.

Transmitting frames

5.6.4.12 The MTA shall maintain a local variable which is the number of retries. This will initially be set to 0.

5.6.4.13 The MTA shall start a timer on transmitting the <SOH>. If this timer reaches *frame_duration* before the end of the frame is transmitted, it shall increment the *frame_tx_failures* counter and reset the link. The local retries counter shall then be incremented and it it is less than or equal to the *retries_allowed* parameter, this process shall be started again. If it is more than the permitted counter, the MTA shall fail as indicated in paragraph 5.6.4.16.

5.6.4.14 Having successfully transmitted all characters in a frame, the transmitting MTA shall

wait for receipt of its *ack_char_rx* acknowledgement character from the remote end. If the MTA does not receive this character within *frame_timeout* seconds, it shall increment the *frame_tx_failures* counter and the local retries counter. If the local retries counter is still less than or equal to the permitted number, the frame shall be retransmitted. If it is more than the permitted counter, the MTA shall fail as indicated in paragraph 5.6.4.16.

5.6.4.15 If the frame is transmitted successfully and acknowledged, the *frame_tx_counter* shall be incremented.

5.6.4.16 If the MTA is still not successful, it shall discard the frame and inform the router that it was unable to complete the transmission.

Link verification

5.6.4.17 At all times, the MTA shall monitor the status of the line (CCITT V.24 circuit 109). If the line fails / recovers, the state of *interface_status* shall be changed appropriately. If the line fails, the MTA shall discard any partial frames which it is receiving / sending. If sending a frame, it shall inform the router that it was unable to complete the transmission.

5.6.4.18 To verify the end-to-end status of the line, the MTA shall maintain a timer which measures the interval since the last valid message was received. If this timer reaches the value of *verify_period* seconds, the MTA shall send an <ENQ> to the remote MTA and wait for receipt of its *ack_char_rx* acknowledgement character. If the timer reaches *verify_timeout*, the MTA shall set the status of the interfact to "off-line (fault)". If *verify_timeout* > 2 (*verify_period)* the MTA whould poll each *verify_period* until *verify_timeout.*

5.6.4.19 If the MTA receives an <ENQ> frame, it shall sent its *ack_char_tx* acknowledgement character back to the remote MTA and reset its inactivity timer.

5.7 LAN MTA

5.7.1 Introduction

5.7.1.1 The LAN MTA is used to provide a high speed interface between equipment at the same site. It shall conform to the GOSIP 4 LAN (CO) subprofile.

5.7.2 GOSIP LAN options

5.7.2.1 All mandatory options within the GOSIP 4 LAN subprofile shall be supported. The LAN MTA shall support the following GOSIP options:

- Connection-mode (CO) mode of operation;
- Carrier Sense Multiple Access with Collision Detection (CSMA / CD) 10base2;
- Class 0 and Class 2 Transport classes;
- be capable of initiating and responding to both Class 0 and Class 2 connection request TPDUs.

5.7.2.2 The selection of CO mode of operation implies the use of:

- Logical Link Control 2;
- Connection-mode Network Service (CONS) at the Network Layer.

5.7.2.3 Tenderers who are offering a LAN MTA shall supply a GOSIP LAN procurement PICS with their response. Contractors shall also indicate what other options they are able to support. In particular, there is likely to be interest in:

- other types of CSMA / CD media, eg 10base5, 10baseT;
- Token Ring.

5.7.3 LAN MTA functions

5.7.3.1 The LAN MTA shall support the following parameters in addition to those supported by all MTAs:

- *my_LAN_address*, which is the LAN address of this MTA.

5.8 ISDN MTA

5.8.1 Introduction

5.8.1.1 The ISDN MTA shall provide an ‘S’ interface to a basic rate ‘2B+D’ ISDN (Integrated Services Digital Network) service (eg BT’s ISDN2).

5.8.2 Conformance with CCITT recommendations

5.8.2.1 The interface to ISDN shall be in accordance with CCITT recommendation I.420,

which refers to the following detailed specifications:

- Basic User-Network Interface - Layer 1 Specification (CCITT I.430);
- ISDN User-Network Interface - Data Link Layer Specification (CCITT I.441 / Q.921);
- ISDN User-Network Interface - Layer 3 Specification for Basic Call Control (CCITT I.451 / Q.931).

5.8.2.2 All terminals, interface cards etc shall be approved for connection to the public network via standard RJ45 sockets (as in BT's Network Terminating Equipment 6C (NTE6C) ISDN2 interface).

5.8.2.3 The system shall not require the use of ISDN supplementary services to provide mobilising communications facilities.

5.8.3 ISDN MTA functions

5.8.3.1 The ISDN MTA shall operate in essentially the same fashion as the PSTN MTA.

6 User agents and the application protocol

6.1 Introduction

6.1.1 User agents (UAs) are the ultimate source or destination of each message.

6.1.2 Conceptually, each UA has a single unique address that corresponds to either:

- a single printer;
- an alerter or paging system;
- a terminal network;
- a set of peripherals;
- a fall-back mobilising system;
- a network management terminal;
- systems external to the Standing Offer, such as:
    - a mobs system;
    - a wall chart;
    - other future UAs which are yet to be specified.

6.1.3 Note that systems external to the Standing Offer will appear to the communications network as a combined MTA and UA(s). Thus, while the UA address(es) of the external system will be known to the network, the interface from the network to the external system will be provided by one of the MTAs described in section 5, and will operate in accordance with the MTA-MTA protocols defined therein. (By contrast, the MTA-UA protocol is internal to a node and is not defined by this specification.)

6.1.4 Note also that where communications bearers external to the Standing Offer are employed (eg in the case of public or private mobile data systems), these external systems do not represent a source or destination of messages. Hence, no UA or UA address is associated with such systems.

6.1.5 A pair of UAs shall communicate using the application protocol. In each case, the communication shall take place using a common protocol, but not all of the protocol will be relevant to all UAs.

6.1.6 For network management purposes, every UA in the system shall support two special UA addresses, relating to the network manager and an alternative network manager. Where a dedicated network management terminal is procured (as an option under the Standing Offer), it is likely that this will be configured as the primary network manager. However, this is not mandatory, and some Brigades may

prefer to designate the mobs system, for example, as the primary network manager.

6.2 Common elements of UAs

6.2.1 Each UA has a minimum set of messages which are recognised as being valid and which can be generated for transmission to the network management UA or to other UAs. These messages are concerned with network maintenance and management.

6.2.2 The following messages shall be accepted as valid by all UAs:

- <ACK>;
- <NAK>;
- <set_parameter>;
- <parameter_request>;
- <param_req_multiple>.

6.2.3 The following messages shall be generated by all UAs:

- <ACK>;
- <NAK>;
- <parameter>.

6.2.4 Receiving messages

6.2.4.1 On receiving a message, the UA shall verify the length and the BCC. If these are incorrect, it shall attempt to send a <NAK> message with a <reason_code_set> of “general” and a <reason_code> of “check_error”.

6.2.4.2 A UA will accept messages of the current protocol version it is loaded with or lower. Messages with a higher protocol version will result in the node returning a <NAK> with a <reason_code_set> of “general” and a <reason_code> of “inv_prot”.

6.2.4.3 If a UA receives a message which it does not understand and the <ack_req> flag is set, the UA shall return a <NAK> as described in appendix B with:

- <reason_code_set> of “general”;
- <reason_code> of “inv_mess”.

6.2.4.4 If <ack_req> is not set, the message shall be discarded.

6.2.4.5 If the message is understood, the MTA will respond as described for the appropriate message in appendix B.

6.2.4.6 If the message is a <NAK>, the UA will examine the reason code.

6.2.4.7 If the reason code is that the message cannot be delivered, then:

- if the message was to the network manager, it should be re-sent to the alternative network manager;

- otherwise, the action is undefined, but will generally result in an alarm being sent to the network manager or operator in the form of a text message.

6.2.4.8 If the <reason_code> is “check_error”, the message shall be re-sent with the same envelope as before. After *retries* attempts, the action is undefined, but will generally result in an alarm to an operator. A request to re-transmit a message which is not in the list of unacknowledged messages shall be ignored.

6.2.4.9 If the <reason_code> indicates that the message was delivered and is being processed, but that a further acknowledgement will be sent in due course, the UA shall start an appropriate timer (eg appropriate for a manual acknowledgement). If this timer expires a local alarm will be generated (eg to warn an operator) but the message will then be removed from the queue of unacknowledged messages.

6.2.4.10 Similarly, the UA which is waiting for a manual acknowledgement so that it can send an <ACK> shall also operate a timer (*man_ack_timeout* in the router table). If this timer expires the UA shall forget the messages which it was waiting to <ACK>. To ensure that this system works correctly, the man_ack_timeout parameter must be set for a shorter duration than the timer referred to within 6.2.4.9.

6.2.4.11 The parameters common to all UAs are:

- port_number, which is the number of the port (1-63);

- agent_type, which defines the type of the port.

6.2.4.12 These parameters shall be readable and modifiable by another UA (usually the network management UA) using the <set_parameter> and <parameter_request> messages.

6.2.4.13 If a <NAK> is returned with a reason code other than <*wait_ack*> then the message should be discarded.

6.3 Alerter UA (option 1)

6.3.1 Introduction

6.3.1.1 The UA shall provide a connection to the interface unit on an MG-4 alerter system which conforms to the standards defined in reference 2.

6.3.2 External interface

6.3.2.1 The interface to the alerter shall be through a V.24 interface. The characteristics of this interface are defined in reference 2.

6.3.2.2 The alerter UA shall include all hardware and software up to the port on the communication system.

6.3.3 Messages understood

6.3.3.1 In addition to the messages common to all UAs, the alerter UA shall recognise the following message types:

- <page_officer>;
- <alert_crew>;
- <alert_eng>.

The appropriate response to each of these messages is as defined in appendix B.

6.3.4 Messages sent

6.3.4.1 In addition to the messages common to all UAs, the alerter UA shall also generate the message type <alert_status>.

6.4 Alerter UA (option 2)

6.4.1 Introduction

6.4.1.1 This UA shall:

- replace the functionality of the interface unit on an alerter system which conforms to the standards defined in reference 2;

- provide the interface to the encoder unit.

6.4.1.2 This UA is optional since the required interfaces are not standard. Tenderers should indicate whether they could provide such a UA, and indicate the features and constraints of their proposed solution.

6.4.2 External interface

6.4.2.1 The interface to the alerter encoder shall be through a V.24 interface. The characteristics of this interface are undefined.

6.4.2.2 The alerter UA shall include all hardware and software up to the encoder including:

- the functionality of the interface unit;
- the master local control panel.

6.4.3 Messages understood

6.4.3.1 The messages which will be understood are the same as for the alerter UA (option 1).

6.4.4 Messages sent

6.4.4.1 The messages which will be generated are the same as for the alerter UA (option 1).

6.5 Paging UA

6.5.1 Introduction

6.5.1.1 This UA shall provide an interface between the communications system and a public or private paging system.

6.5.2 External interface

6.5.2.1 The protocol used to access the paging service is dependent upon the Contractor of the service or system and hence can not be defined within this specification. Tenderers shall:

- state which paging services or systems they are able to support;

- state the protocols and interfaces used to connect to the paging service.

6.5.3 Messages understood

6.5.3.1 In addition to the messages understood by all UAs, the paging UA shall understand the <page_officer> message.

6.5.4 Messages sent

6.5.4.1 It is not envisaged that the UA will generate any messages in addition to the messages which are common to all UAs.

6.6 Peripheral UA

6.6.1 Introduction

6.6.1.1 The purpose of the peripheral UA is to:

- allow output peripherals (eg sounders, lights etc) to be activated remotely;
- allow the status of input peripherals to be monitored remotely;
- provide an interface to local controls (eg acknowledge printer message, reprint last message).

6.6.2 External interface

6.6.2.1 Each peripheral UA shall provide:

- 16 input channels;
- 16 output channels.

6.6.2.2 Tenderers shall identify alternative options which they could supply with fewer circuits and shall identify the costs for these alternative options.

6.6.2.3 It should be possible to provide more than 16 input and output lines by providing additional UAs.

6.6.2.4 Each input channel shall be presented on two wires. The input state shall be indicated by terminating the lines with either a high or low impedance.

6.6.2.5 The inputs should be protected against accidental connection to standard 240V ac mains supplies.

6.6.2.6 Input circuits shall be debounced.

6.6.2.7 Each output circuit should be presented on three terminals:

- a common terminal;
- a normally open terminal (which is connected to the common terminal when the output is asserted);
- a normally closed terminal (which is disconnected from the common terminal when the output is asserted).

6.6.2.8 The output circuits shall be voltage free and shall be rated to switch 10A at 250V ac.

6.6.2.9 The UA shall include all hardware and software up to the termination point on the equipment.

6.6.2.10 The installation option shall include the supply and installation of cables and other ancillaries necessary to connect the equipment to a distribution rack.

6.6.3 Messages understood

6.6.3.1 In addition to the messages common to all UAs, the peripheral UA shall recognise the following message types:

- <mobilise_command>;
- <activate_peripheral>;
- <deactivate_peripheral>;
- <peripheral_status_request>.
- <alert_crew>.

6.6.3.2 Each of the commands is described in appendix B. The difference between the output commands is that:

- <mobilise_command> and <alert_crew> activates a number of outputs for a preset period and, where possible, confirms that the command has been successful before sending a return;
- <activate_peripheral> asserts a number of outputs and then sends an

acknowledgement. The outputs remain asserted for a preset period or until a <deactivate_peripheral> command is received.

6.6.3.3 All of the above commands use the following additional parameters which are supported by the peripheral UA:

- parameter type *op_map* which defines:
  - how the functions within the <op_peripherals> field map to physical output lines;
  - how long an output should be asserted for;
- parameter type *ip_map* which defines:
  - how functions within the <ip_peripherals> field map to physical input lines;
  - whether an input is asserted when the input state is closed or open;
  - whether an input being asserted should cause an unsolicited command to be sent to the network manager;
  - whether an input being changed should cause an unsolicited command;
- parameter *mobilising_printer* which defines the address of the local printer which is used for mobilising.

6.6.4 Messages sent

6.6.4.1 In addition to the messages common to all UAs, the peripheral UA shall also be capable of generating a <peripheral_status> message.

6.6.4.2 If an input changes to the asserted state, the peripheral UA shall look at the appropriate entry in the *ip_map* to see if it should send an unsolicited <peripheral_status> message to the network manager UA. If so, the peripheral UA shall send a <peripheral_status> message as described in appendix B.

6.6.5 Other functions

6.6.5.1 The <physical_bit> data entity indicates which peripherals are to be activated / deactivated. Whether the peripheral UA has the particular peripheral device is indicated in the parameters associated with that UA (see table C.13, p121). The ip_map or op_map is set to 255 indicating that the function is not provided.

6.6.5.2 If the *ip_map* allocates a physical input to the function of 'reprint last message' and this input is asserted, the peripheral UA shall command the printer UA as defined by *mobilising_printer* to reprint the last message it printed as a result of receiving a <mobilise_message>.

6.6.5.3 If the *ip_map* allocates a physical input to the function of 'manual acknowledge' and this input is asserted, the node shall return full <ACK> messages in response to any messages which have been sent to any local UAs and which require a manual acknowledgement. If there are any such messages outstanding, the peripheral UA shall not send an unsolicited <peripheral_status> as required by paragraph 6.6.4.2.

6.7 Printer UA

6.7.1 Introduction

6.7.1.1 The function of the printer UA is to accept a serial stream of data from the router, to strip off the addressing information from the message and pass the data to the destination printer for printing.

6.7.2 External interface

6.7.2.1 The printer port shall be either:

- a Centronics 8 bit parallel port;
- a V.24 interface as defined in appendix E.

6.7.2.2 The Centronics connector on the port to the comms equipment should be a female 25 way D type (clearly marked `Parallel').

6.7.2.3 Printer control shall be achieved by implementing Epson FX compatible code sequences.

6.7.2.4 Printer control shall be implemented so as not to delay the printing of priority messages.

6.7.2.5 'Printer out-of-paper (PE)' and 'Paper low' shall be implemented either through the Centronics interface or through external switching contacts.

6.7.2.6 The printer shall have Auto Line Feed (LF) with Carriage Return (CR) enabled.

6.7.3 Messages understood

6.7.3.1 In addition to the messages common to all UAs, the printer UA shall recognise the following message types:

- <mobilise_message>, which is a structured message containing a turnout instruction;
- <text_message>, which is an unstructured message.

6.7.3.2 The <mobilise_message> shall be printed and processed as described in appendix B.

6.7.3.3 If a printer UA receives a priority 1 <mobilise_message> whilst printing a lower priority message then, unless the lower priority message will finish printing within 10 seconds, the UA shall:

- issue a form feed or other means of clearing the lower priority message;
- print the <mobilise_message> with its final form feed;
- start printing the lower priority message from the beginning or from the start of the previous page.

6.7.3.4 If a mobilise_message arrives at a printer UA, and a manual acknowledgement is still outstanding and reprint_message is true then a clearly annotated station copy of the previous mobilise_message should be printed before the current message.

6.7.3.5 If a mobilise_message is received which does not require a manual acknowledgement and reprint_message is true then a clearly annotated station copy of the mobilise_message should be printed immediately after the master copy.

6.7.3.6 If the printer UA is commanded to reprint the last message whilst it is part way through printing a message it should suspend printing the current message and reprint the last message fully printed. The current message should then be reprinted from the start.

6.7.4 Messages sent

6.7.4.1 In addition to the standard parameters associated with UAs, the printer UA shall support a flag *notify_printer_available*. If this flag is set and the printer UA detects that the printer has gone offline, the printer UA shall send a <printer_status> message to the network management UA.

6.7.5 Other functions

6.7.5.1 The printer UA shall store the last <mobilise_message> it received. If the printer UA receives a request from the peripheral UA to reprint the last message, it shall reprint the message in the same format as the first time.

6.8 Resource UA

6.8.1 Introduction

6.8.1.1 The function of the resource UA is to maintain a table of resources.

6.8.1.2 Each UA shall be able to store the details of up to 20 resources using a limited <status_code> set to 0 (no status data), 4 (available at base), 5 (not available) and 6 (available contact by radio).

6.8.2 Messages understood

6.8.2.1 In addition to the messages common to all UAs, the resource UA shall recognise the following message types:

- <resource_status_request> which is used to ask the UA to return its resource table;
- <resource_status> which is used to update the table held by the UA.

6.8.2.2 On receipt of a <resource_status_request> the UA shall respond with a <resource_status> message. See Section 6.8.3.

6.8.2.3 On receipt of a <resource_status> message, the UA shall update its resource table for each of the resources contained in the message. If new resources are included within the message and there is insufficient space in the table to store the details of those resources, then the UA shall return a <NAK> with a <reason_code_set> of “general” and a <reason_code> of “inv_call”.

6.8.2.4 If the UA receives information about a resource with the <status_code> set to “No status data” it shall remove that resource from its table (assuming it existed).

6.8.3 Messages sent

6.8.3.1 In addition to the message common to all UAs, the UA shall generate <resource_status> messages as described in Appendix B.10.

7 General equipment requirements

7.1 Design and construction standards

7.1.1 All equipment shall be designed and built to good commercial standards consistent with the required design life of at least ten years.

7.1.2 To ensure future support and supply over these timescales, all equipment shall (wherever possible) be based on 'open' or established industry standard hardware platforms and software products. Tenderers shall identify explicitly any items of proposed equipment that are proprietary or single sourced.

7.2 Software design standards

7.2.1 Software should be written in a well established, high-level language.

7.2.2 Where a low-level language is employed, the Tenderer shall state:

- the extent to which the low-level language is used;
- the reason for not employing a high-level language;
- what aspects of functionality are provided by low-level software.

7.2.3 All software shall be modular in design.

7.2.4 Tenderers shall state:

- the design methodology that will be used;
- the standards to which software will be:
  - developed;
  - tested;
  - documented;
- whether they hold registration to BS 5750 Part 1;
- what procedures will be adopted for change control.

7.3 Approvals

7.3.1 Contractors shall ensure that the equipment being supplied has the required type approvals from an accredited approvals house.

7.3.2 Where standards are called up within this specification, Contractors shall provide evidence to confirm that the equipment being supplied has been independently

tested for conformance with the required standard.

7.4 Labelling

7.4.1 All communications interfaces must be labelled to show:

- to what the interface is normally connected;
- how the interface is configured.

7.4.2 Indicators of the operational status of communications equipment must be labelled.

7.4.3 Switch positions and configuration controls must be identified with labelling indicating the function of the switch or control.

7.5 Physical characteristics

7.5.1 Tenderers shall state the dimensions of the space required for each piece of equipment. These figures shall include any space required for access purposes.

7.5.2 Tenderers shall state the weight of their equipment.

7.5.3 All equipment for station use (not including batteries) should be capable of either wall, floor or rack mounting.

7.6 Installation and commissioning

7.6.1 Tenderers shall quote for installation as a separate item.

7.6.2 Installation shall include all parts and labour which are required to:

- deliver the supplied equipment to site;
- where necessary, decommission the superseded equipment and remove it to an agreed location;
- install the newly supplied equipment;
- test and commission the supplied equipment.

7.6.3 At completion of installation, all fixed equipment shall be:

- properly earthed by connection to an earthing bus bar which will be provided by the Brigade;

- securely mounted by means of fixings to the floor or similar;
- installed so as to ensure easy access for maintenance or replacement of units.

7.6.4 Installation of mobile equipment in appliances and other vehicles shall include:

- connection of the equipment to the vehicle power supply;
- secure mounting of the equipment within the vehicle cabin;
- interfacing of any peripherals to the equipment.

7.6.5 The Contractor shall agree detailed installation arrangements with the Brigade not less than 2 months in advance.

7.6.6 Installation will be deemed to be complete when the equipment has passed a set of acceptance tests which demonstrate it working as an integral part of a complete communications system.

7.6.7 A general statement of the acceptance tests to be run will be defined as part of the Standing Offer arrangement. The Contractor shall provide the Brigade with a detailed test script at least 1 month in advance of the start of the tests, and shall not commence testing without written approval from the Brigade.

7.6.8 Installation shall include, but not be limited to, the supply and installation of:

- all cables required to interconnect the different elements of equipment which have been supplied for one site;
- any plugs / sockets or other items required to terminate cables.

7.6.9 For the purposes of the quotation, it shall be assumed that all cable runs for fixed equipment are less than or equal to 10 m in length. Tenderers shall state any additional charge that will apply for supplying and installing cables of longer lengths, and shall also state the maximum recommended length for each cable.

7.6.10 Installation shall include the provision of labour for on-site configuration of the system to meet the Brigade's requirements.

7.7 Safety and general installation standards

7.7.1 The installation of the system shall satisfy the provisions of current health and safety legislation, including the Health and Safety at Work Act (1974), the Electricity at Work Regulations (1989), and the IEE Wiring Regulations (current edition).

7.7.2 The system shall be designed and installed so as to avoid all risk of injury to operators and maintenance staff. Any unavoidable risks, whether mechanical, electrical, chemical or of any other kind, shall be clearly identified by means of warning notices or similar.

7.7.3 Exposure to hazardous voltages shall not be possible without the use of special tools.

7.7.4 All cables carrying data signals shall be protected by an earthed shield.

7.7.5 The length of all cable runs carrying data signals shall be minimised.

7.8 Maintenance

7.8.1 The requirements for maintenance are defined in the ITT.

8 Documentation and training

8.1 Documentation

8.1.1 Tenderers shall itemise and cost separately all user and technical documentation described in this sub-section.

8.1.2 A system design document, including a description of the system configuration, shall be provided. This shall be at a level appropriate to allow full management of the system, including the future incorporation of other equipment purchased from other manufacturers.

8.1.3 Technical documentation shall also be provided in the form of a maintenance handbook on all items of hardware supplied, using the following sub-headings or equivalent:

- Performance Specification: A detailed performance specification shall be provided for each unit including environmental and power supply limits.

- Test Specification: A test specification shall be provided for each item, including details of the test procedure, test equipment, etc.

- Setting to Work Instructions: These shall be provided for each item and shall cover any special handling requirements of the equipment.

- Routine Maintenance Instructions: These shall include details of routine tests together with limiting values. Any waveforms which are important in testing, particularly timing diagrams, should be represented graphically, with reference to test points in the circuits.

- Fault Tracing Instructions: These shall be comprehensive, enabling faults to be traced to sub-units and components in sub-units or on printed circuit boards. Relevant waveforms should be represented graphically.

- Circuit and Layout Drawings: Diagrams shall be provided for each sub-unit and circuit board as follows:

    - circuit diagrams, including component values;

    - component layout;

    - interface connection and sub-system inter-connection diagrams.

- Component Schedule: A list of all components shall be provided, giving manufacturers' part numbers, quantities used, Contractor second source, values, ratings, tolerances, etc.

8.1.4 Full documentation shall also be provided for all software which forms part of the system and shall include the following:

- a functional description of all modules and their interactions;
- fully documented source code for all modules;
- memory maps for all memory mapped devices;
- listings of all object code;
- details of any compilers which are required.

Tenderers should also supply object code in soft copy form, together with appropriate instructions such that any PROMs etc could be re-created by the Brigade as might be required in the future.

8.1.5 Draft documentation shall be provided not later than 2 months in advance of the delivery date for the equipment to which it applies. Final documentation (that accurately describes the final configuration of the installed and fully working system) shall be provided not later than 2 months after final system acceptance.

8.2 Training

8.2.1 Tenderers shall offer, and cost separately, the provision of training courses for a number of Brigade staff. Courses shall be provided for system managers and maintenance staff. The course for system managers shall include:

- a general description of the equipment;
- instructions on setting up and configuring the equipment;
- instructions on operating the equipment;
- instruction on diagnosing which item of equipment has failed.

8.2.2 The course for maintenance staff shall include:

- all aspects covered within the system managers course;
- procedures for diagnosis of faults at a module level within a piece of equipment;
- instructions on replacing modules within equipment;
- instructions on installing and commissioning equipment.

8.2.3 Tenderers shall state the cost of such courses on a per individual basis or a per course basis, together with the maximum number of attendees. Tenderers shall state the duration and location of such courses.

8.2.4 Appropriate documentation shall accompany all training courses and shall cover all aspects of the course material. Draft documentation shall be provided at least 3 months prior to the start of the training course.

9 References

1 *'Fire Service Mobilising Systems User Requirements and Functional Analysis'*, GD-92 / 1002 / 2.0, 10 July 1992.

2 *'Replacement firemans alerter system - draft technical specification MG-4'*, Home Office Radio Frequency and Communications Planning Unit, 10 September 1990.

A Data entities

A.1 Definition language

A.1.1 The nomenclature used in this appendix to describe message data entities and in appendix B to describe message contents is to be interpreted as follows:

<...>	Represents a data entity.
(<..><..>)	The entity groups enclosed in parentheses may be repeated.

A.2 Encoding of data entities

A.2.1 In defining data entity formats, the following guidelines are applied:

- A few short fields are fixed length, typically a byte or a word.
- The vast majority of fields are of variable length.
- Most fields that can be of variable length are prefixed by a byte which indicates the length.
- Very long fields of variable length are prefixed by a word which indicates the length.
- A desirable aim is for brevity of messages, so where data is binary by nature, then binary representations are used in the message.
- Where byte and bit numbers are referred to, the first byte or bit (ie the most significant bit / byte) is numbered zero.
- The basic data comprises an 8 bit byte. All binary representations of numbers in data structures are to be considered unsigned, ie a single byte may take values from 0 - 255.

A.2.2 In building up the structure of the entities as shown in the data dictionary, a number of basic data types are used. The hierarchy of these byte-oriented data types are:

- <bit_mapped_binary>: the status of each individual bit (ie “clear” or “set”) is coded separately to carry a single element of binary information.
- <binary>: the direct representation of an unsigned number in the range 0 - 255 encoded in binary form. In multi-byte fields, the first byte is the most significant.
- <ASCII_numeric>: the set of 7-bit ASCII codes representing the numerals 0 - 9; the most significant bit is set to “0”.

- <ASCII_alphanumeric>: the set of 7-bit ASCII codes representing A - Z in upper and lower cases, numerals 0 - 9 and space; the most significant bit is set to “0”.

- <ASCII_alphabet>: the set of 7-bit ASCII codes representing A - Z in upper and lower cases; the most significant bit is set to “0”.

- <ASCII>: the set of all 7-bit ASCII codes in the range 0 - 127; the most significant bit is set to “0”.

- <string>: a data type which has a variable length. The first byte of such a string is a binary byte <count>, showing the length, in bytes, of the string which follows it (this length does *not* include the <count> byte). The remainder of the data type is a series of ASCII codes. The <string> data type is used for data fields which have a variable length, but whose maximum allowed length is fairly short, and so no compression is used on the ASCII data.

- <compressed_string>: a data type formed in a similar way to the <string> type above. Again, the first byte of such a <compressed_string> is a binary count of the byte length of the remainder. This data type is used for fields which have a variable length and whose maximum size is fairly large, and so the data compression technique described below is applied to the initial ASCII data. The byte length contained in <count> is the length of the compressed string.

- <long_comp_string>: a data type formed in an identical fashion to the <compressed_string> above, except that the length of the string is encoded in a two byte field <long_count>.

A.3 Data compression

A.3.1 A number of messages use the data type <compressed_string>. The composition of this type of data is described below.

A.3.2 UAs shall recognise characters which are repeated more than three times in succession in an outgoing messages and shall compress these to a string of the form:

<ESC><char>N

where <ESC> is the ASCII escape character, and N is of type <binary> and indicates the number of occurrences.

A.3.3 For example, the string:

<sp><sp><sp><sp><sp>HIGH<sp>STREET

in an outgoing message would be transmitted as:

<ESC><sp><05_{hex}>HIGH<sp>STREET

A.3.4 Note that this processing shall only take place on the data type compressed string. No other data type will be compressed.

A.3.5 To ensure correct translation at the receiving end and maintain consistency in the implementation of coding, any single <ESC> character in the <compressed_string> contents before compression will be replaced by <ESC><ESC>001.

A.3.6 Similarly, the receiving end shall recognise the <ESC> character in the contents of incoming messages and perform character decompression using the next two characters.

A.3.7 Any length limitation applies to the length of the *compressed* string. The protocol does not artificially constrain the length of an uncompressed string.

A.4 Data entity definitions

A.4.1 The detailed message structure given in appendix B refers to a number of data entities which are here defined essentially in alphabetic order. Where natural, the definition of data sub-types upon which the definition of one of the data types listed in the message entity tables relies are grouped immediately beneath the data type to which they are relevant.

A.4.2 When maximum lengths of the data types <string> and <compressed_string> are given, these refer to the length as held in the <count> field of the data type, and thus do not include the one byte of the <count> field itself. A string of length 0 is a valid type used to indicate that a field is not being supplied.

A.4.3 In the following, bytes and bits are numbered from 0 (which is the most significant). Note that since 8 bit ASCII is used, normal printable characters have the most significant bit set to 0.

<active_state> <word8> encoded as follows:

0	active when contacts closed
1	active when contacts open

<address> <address_text><house_number><street><sub-district><district><town><county><postcode>

with:

<address_text> <compressed_string> of length 0 - 120, giving the premises (zone) and name

<house_number> <string> of length 0 - 10

<street> <compressed_string> of length 0 - 40

<sub-district> <compressed_string> of length 0 - 30

<district> <compressed_string> of length 0 - 30

<town> <compressed_string> of length 0 - 30

<county> <compressed_string> of length 0 -20

<postcode> <string> of length 0 - 10

<address_range> <comms_address><comms_address>

<address_string> <compressed_string> of length 0-30, giving the alternative address printed at the top of the message

<address_table> (<address_range>, <address_string>)

<agent_type> <word8> encoded as follows:

0 ISDN (basic rate)

1 PSTN

2 Private wire

3 Radio

4 Printer

5 Alerter UA (Option 1)

6 Alerter UA (Option 2)

7 Paging UA

	8	Peripheral UA
	9	WAN MTA
	10	LAN MTA
	11	Asynch MTA
	12	Network Management UA
	13	MOBS UA
	14	Standby MOBS UA
	15	Resource UA
	16	MDT UA
	17	AVLS UA
	18	Clarion
	19	Paknet
	20	RAM
	21	Radio (ETSI 300/230) MTA TBD
	22	User Defined
	23	User Defined
	24	User Defined
	25	User Defined
<alarm_ref>	<word16>	
<alarm_serial>	<string> of length 0 -12	
<alarm_type>	<string> of length 0 - 10	
<alerter_engineering>	<word8> ASCII alphabet encoded as follows:	
	A	Lock system to Tx A
	B	Lock system to Tx B

	Z	Restore alternate message transmit keying
	J	Continuous Tx test, carrier only
	K	Cease continuous carrier
	L	Continuous Tx test, repeat engineering code address
	M	Cease engineering codeword transmission
	C	User defined parameter
	D	User defined parameter
	E	User defined parameter
<alerter_status>	<word16> ASCII alphabet encoded as follows:	
	ba	Local operation of firecall team “A” buttons
	bb	Local operation of firecall team “B” buttons
	bc	Local operation of firecall team “C” buttons
	bd	Local operation of firecall team “All Teams” buttons
	ca	Local operation of testcall team “A” buttons
	cb	Local operation of testcall team “B” buttons
	cc	Local operation of testcall team “C” buttons
	cd	Local operation of testcall “All Teams” buttons
	xa	Fault, Tx A, low forward power
	xb	Fault, Tx A, high VSWR
	xc	Fault, Tx A, other
	xd	Fault, Tx A, other
	xe	Fault, Tx B, low forward power
	xf	Fault, Tx B, high VSWR

xg	Fault, Tx B, other
xh	Fault, Tx B, other
xz	Total failure both transmitters
ea	Fault, encoder
eb	Fault, power supply mains fail
ec	Fault, low battery voltage
ed	Fault, unspecified
la	System local fault, locked to Tx A
lb	System local fault, locked to Tx B
xj	Fault cleared, Tx A, low forward power
xk	Fault cleared, Tx A, high VSWR
xl	Fault cleared, Tx A, other
xm	Fault cleared, Tx A, other
xn	Fault cleared, Tx B, low forward power
xp	Fault cleared, Tx B, high VSWR
xq	Fault cleared, Tx B, other
xr	Fault cleared, Tx B, other
xs	Fault cleared total failure both transmitters
ee	Fault cleared, encoder
ef	Fault cleared, power supply mains fail
eg	Fault cleared, low battery voltage
eh	Fault cleared, unspecified
lc	Fault cleared, system local fail, locked to Tx A
ld	Fault cleared, system local fail, locked to Tx B

	rd	Fault cleared, no reply from monitor receiver
<alert_group>	<word16> ASCII alphabet encoded as follows:	
	FA	Firecall team A
	FB	Firecall team B
	FC	Firecall team C
	FD	Firecall teams A, B and C
	FE	Firecall teams A and B
	FF	Firecall teams B and C
	FG	Firecall teams A and C
	TA	Testcall team A
	TB	Testcall team B
	TC	Testcall team C
	TD	Testcall teams A, B and C
	TE	Testcall teams A and B
	TF	Testcall teams B and C
	TG	Testcall teams A and C
<appl_quantity>	<word8> binary	
<appl_type>	3 bytes ASCII alphanumeric, left justified and filled with spaces encoded as follows:	
	HP	Hydraulic Platform
	FEV	Foam Equipment Vehicle
	L4R	Light Four Wheel Drive
	etc for all codes defined in the "Yellow Book"	
<available>	boolean	

<AVL_data>	<string> of length 0 - 40. If <AVL_type> = 0 then <AVL_data> has length 0
<AVL_type>	<word8> encoded as follows: 0 no AVL data system present 1 - 255 reserved for assignment to AVL systems
<block>	<word8>
<boolean>	<word8> encoded as follows: 0 FALSE 1 TRUE
<call_agency>	<string> of length 0 - 10
<callsign>	<string> of length 0 - 6 ASCII alphanumeric
<callsign_list>	<count>(<callsign>), where <count> denotes the number of callsigns in the list
<comms_address>	3 bytes (24 bits), binary encoded as follows: bits 0-7 brigade_identifier (0-255) bits 8-17 node_identifier (0-1023) bits 18-23 port_identifier (0-63)
<connect_time>	<word16> (seconds)
<connect_type>	<word8> encoded as follows: 0 Permanent Virtual Circuit 1 Switched Virtual Circuit
<count>	<word8>
<destination_nodes>	<count>(<address_range>)
<dial_tones>	<word8> encoded as follows: 0 pulse dialling

1 tone dialling

<envelope> <source><count&length><destination><prot&priority><ack&seq><message_type><message> <bcc>

with:

<source> <comms_address>

<count&length> <word16> encoded as follows:

bits 0 - 9 <length>

bits 10 - 15 <dest_count>

<length> 10 binary bits (0 - 1,023) denoting the message length

<dest_count> 6 binary bits (1 - 63) denoting the number of destinations

<destination> (<comms_address>)

<prot&priority> <word8> encoded as follows:

bits 0 - 3 <priority>

bits 4 - 7 <prot_vers>

<priority> 4 binary bits (1 - 9) denoting the message priority

<prot_vers> 4 binary bits (1 - 15) denoting the protocol version operated by the message originator

<ack&seq> <word16> encoded as follows:

bits 0 - 14 <seq>

bit 15 <ack_req>

<seq> 15 binary bits (0 - 32767)

<ack_req> 1 bit indicating whether an acknowledgement is required to the message. Encoded as follows:

0 no ack_required

1 ack_required

<message_type> <word8> indicating the number of the message type

<message> see appendix B

<bcc> <word8> which is calculated by exclusive-ORing all of the preceding bytes in the envelope and message together

<first_entry> <index>

<format_type> <word8> encode as follows:

1 text table

2 address information

<generate_alarm> <word8> encode as follows;

0 do not generate an alarm message;

1 generate alarm message on assertion;

2 generate alarm message on de-assertion,

3 generate alarm message on both;

4 generate alarm if asserted longer than set <regen_time>;

5 generate alarm for each period longer than set <regen_time>;

6 generate alarm for each period longer than set <regen_time>;

7 genereate alarm for each period longer than set <regen_time>.

<hold_time> <word8> (in seconds)

<incident_number> <word32>

<index>	<word16>
<ip_map>	<physical_bit><active_state><generate_alarm>
<ip_peripherals>	<word16>. Each bit is mapped to a peripheral function; bit set indicates:
	0 Manual acknowledge pressed
	1 Power failed, batteries on
	2 Power failed, standby generator on
	3 Repeat last message pressed
	4 Batteries low
	5 Paper low
	6 Spare
	7 Spare
	8 Running-Call-Telephone door opened
	9 Paper out
	10 - 15 Spare
<ISDN_table>	(<index><used><next_node><tel_number><hold_time><available>)
<LAN_address>	<string> of length 0 - 16
<LAN_table>	(<index><used><next_node><LAN_address>)
<last_entry>	<index>
<level>	<word8> (0 - 4)
<long_count>	<word16>
<man_ack_req>	<boolean>
<map_ref>	<string> of length 0 - 16
<MDT_table>	(<index><used><next_node><network_user_address>

	<hold_time><available>)
<mobilisation_type>	<word8> encoded as follows:
	0 Pre-alert
	1 Incident
	2 Non-incident
	3 Batch mobilise address
	4 Standby
	5 Demobilise
	6 Test
	7 - 255 unassigned
<more_values>	<boolean>
<MTA_status>	<word8> encoded as follows:
	0 idle
	1 on-line
	2 off-line (user initiated)
	3 off-line (fault)
<network_user_address>	<string> of length 0 - 14
<next_node>	<comms address> of the router at a node
<node_name>	<string> of length 0 - 20
<number_types>	<word8> (0 - 12)
<of_blocks>	<word8>
<OIC>	<string> of length 0 - 20
<op_map>	<physical_bit><active_state><pulse_length>
<op_peripherals>	<word16>. Each bit is mapped to a peripheral function;

	bit set stimulates:
	0 Station sounders
	1 Station lights
	2 Spare
	3 Spare
	4 Spare
	5 Station doors
	6 Standby sounder
	7 spare
	8 - 15 Appliance indicators 1 - 8
<pager_number>	<tel_number><pager_type>
<pager_priority>	<word8> ASCII alphabet encoded as follows:
	E emergency
	P priority
	R routine
	A administrative
<pager_text>	<compressed_string> of length 0 - 200
<pager_type>	<word8> ASCII alphabet encoded as follows;
	A alphanumeric paging call;
	N numeric paging call;
	T tones only.
<parameter_no>	<word8> (0 - 255). For assignment see appendix C.
<parameter_table>	<word8> (0 - 2) encoded as follows:
	0 permanent table

	1 non-volatile table
	2 current table
<parameter_value>	For valid <parameter_value> data types, see appendix C.
<pass_param>	<level><password><comms_address>
<password>	<string> of length 0 - 10
<physical_bit>	<word8> (0 - 15)
<port_number>	<word8> (0 - 63)
<preference>	<word8>
<printer_status>	<word8> with:
	0 off line
	1 paper out
	2 on line
<PSTN_table>	(<index><used><next_node><tel_number><hold_time><available>)
<pulse_length>	<word16> (time in seconds)
<query_type>	<word8> ASCII alphabet encoded as follows:
	to be defined later.
<reason_code>	<word8> (0 - 255). For assignments see appendix D
<reason_code_set>	<word8> (0 - 4) encoded as follows:
	0 printer
	1 general
	2 alerter
	3 peripheral
	4 parameter

<regen_time>	<word16> in seconds
<remarks>	<compressed_string> of length 0 - 200
<request_code>	<word8> ASCII alphabet encoded as follows:
	S Request to speak
	E Emergency
	C Request to speak confidential
<reset_reason>	<word8> encoded as follows:
	0 Requested by <Reset_request> message
	1 Software failure
	2 Power on
	3-255 Reserved for future expansion
<reset_type>	<word8> encoded as follows:
	0 software reset
	1 hardware reset
	2 hardware reset and re-load parameters from permanent tables
<riders>	<word8> (1 - 15)
<routing_table>	(<index><used><next_node><destination_nodes> <agent_type> <preference>)
<status_code>	<word8> encoded as follows:
	0 No status data
	1 Mobile to Incident
	2 In Attendance at Incident
	3 Available at Incident

4	Available at Base
5	Not Available
6	Available Contact by Radio
7	Available Home Address
8	Available Guesting
9	Available Pager
10	Mobile to Stand-by Location
11	Available by Telephone
12	Incident closed
13	Stop (status)
14	Mobile unavailable
15	Defective unavailable
16	Insufficient crew available
17	Available other telephone
18	Out of area (eg O.B)
19	Pending (between committed and MI)
20	Available delayed turn-out
21	Next message
22	Keep informed
23	Redirect to another incident
24	Mobile in standby station’s area
25	At standby station
26	Mobile to guest at another station
27	Mobile guesting in another station’s area

	28 Delayed response - off station
	29 Mobile - return to station if required
	30 Mobile - depleted crew
	31 Mobile - BA deficiency
	32 Available alerter
	33 On duty - not available
	34 Available base (officer)
	35 Mobile available
	36 Available - mobile to base
	37 Off Duty
	38 Batched
<stop_code>	5 bytes ASCII alphanumeric
<table>	<compressed_string> of length (0-255)
<tel_number>	<string> of length 0 - 16; left justified, ASCII numeric or ASCII space
<tel_stats>	(<index><time_and_date><tel_number> <connect_time>)
<test_type>	<word8> (0 - 255)
<text>	<long_comp_string>
<time_and_date>	<string> in format "DDMMMYYHHMMSS" encoded as follows: **(No byte count)**
	DD 01-31 MMM first 3 characters of the month name in uppercase ASCII YY 00-99 HH 00-23 MM 00-59 SS 00-59

A Data entities

<update>	<compressed_string> of length (0 - 255)
<used>	<boolean>
<WAN_address>	<string> of length 0 - 16
<WAN_stats>	(<index><time_and_date><WAN_address> <connect_time>)
<WAN_table>	(<index> <used> <next_node><WAN_address> <connect_type>)
<word8>	1 byte binary (0 - 255)
<word16>	2 bytes binary (0 - 65535)
<word32>	4 bytes binary

B Message content definitions

B.1 List of all messages

B.1.1 Tables B-1 to B-6 below give a list of all the message types which are defined in the protocol, using the nomenclature defined in appendix A.

B.1.2 In defining message content formats, the following guidelines are applied:

- Mandatory fields are placed at the front of the message.
- Optional fields are placed at the end of the message, with the fields least likely to be used placed at the end.
- Optional fields may be omitted from the message entirely if and only if all of the following optional fields are also omitted.
- Messages may include a set of repeating fields, denoted by (<...>). Any repeating set must be placed at the end of the message (since otherwise the message cannot be unambiguously decoded).
- From the two definitions above, it follows that a message may be defined to have optional fields or repeating fields but not both.
- Note that any fields of variable length, whether optional or mandatory, may be reduced to length 0 to indicate that no information is being supplied.

B.1.3 In the tables, the entry under the heading “Ack req” refers to the contents of the <ack_req> field in the envelope. This field is used by the message transfer system to determine whether it is required to inform the sender about a message which could not be delivered.

B.1.4 Similarly, the entry in the column headed “Priority” indicates the normal setting of the <priority> field in the envelope.

B.1.5 While the table shows the expected values of the <ack_req> and <priority> fields, the correct operation of systems shall not be dependent upon the priority fields being set as shown.

B.1.6 Note that in this appendix, message types have not been denoted with triangular brackets, in order to distinguish the messages from data types or field types, eg message number 01 is referred to as Mobilise_command rather than <Mobilise_command>.

B.1.7 Message numbers run from 01 to 66, with associated messages generally (but not always) grouped together. The message numbering scheme has developed over time and the following points should be noted:

- message numbers 4, 6, 11, 26 and 29 have existed in previous versions of the messaging scheme, but have subsequently been deleted;

- message numbers 12 - 19, 32 - 39, 44 - 49 and 52 - 59 have never been defined.

Message Number	Message Name
01	Mobilise_command
02	Mobilise_message
03	Page_officer
04	Area_page_message*
05	Resource_status_request
06	Mobilise_message*
07	Activate_peripheral
08	Deactivate_peripheral
09	Peripheral_status_request
10	Reset_request
11	DELETED
20	Resource_status
21	Duty_staffing_update
22	Log_update
23	Stop
24	Make-up
25	Interrupt_request

Table B-1
Numerical list of messages

Message Number	Message Name
26	DELETED
27	Text_message
28	Peripheral_status
29	DELETED
30	Reset
31	Incident_notification
40	Alert_crew
42	Alert_status
43	Alert_eng
50	<ACK>
51	<NAK> (Negative acknowledge)
60	Set_parameter
61	Parameter_request
62	Parameter
63	Param_req_multiple
64	Test
65	Printer_status
66	MTA_status_change
67	Route_status

Table B-1 (continued)
Numerical list of messages

B Message content definitions

Message Number	Message Name
100	Supplier_message
101	Brigade_message
102	Data_base_query
103	Formatted_text
104	Proforma_definition_query
105	Proforma_definition

Table B-1 (continued)
Numerical list of messages

B Message content definitions

Msg No	Message name	Ack req	Priority	Structure
01	Mobilise_command	Yes	1	<op_peripherals><man_ack_req>
02	Mobilise_message	Yes	1	<block><of_blocks><man_ack_req><time_and_date><call sign_list> (<incident_number><mobilisation_type><address> <map_ref><tel_number><text>)
03	Page_officer	Yes	varies	<pager_priority><pager_number><pager_text>
04*	Area_page_message			See Table B-7
06*	Mobilise_message			See Table B-7
27	Text_message	Yes	varies	<block><of_blocks><text>
40	Alert_crew	Yes	1	<alert_group><man_ack_req><op_peripherals>

Table B-2
Mobilisation messages

Msg No	Message name	Ack req	Priority	Structure
05	Resource_status_ request	Yes	3	(<callsign>)
20	Resource_status	Varies	3	(<callsign><AVL_type><AVL_data><status_code> <remarks>)
21	Duty_staffing_update	Varies	3	(<callsign><OIC><riders><status_code><remarks>)
22	Log_update	Yes	4	<callsign><incident_number><update>
23	Stop	Yes	4	<callsign><incident_number><stop_code>
24	Make-up	Yes	2	<callsign><incident_number><number_types> (<appl_type> <appl_quantity>)
25	Interrupt_request	Yes	2	<callsign><request_code><text>
31	Incident_notification	Yes	2	<alarm_type><call_agency><tel_number><alarm_ref> <alarm_serial><address><text>

B Message content definitions

Table B-3
Resource or incident messages

Msg No	Message name	Ack req	Priority	Structure
07	Activate_peripheral	Yes	1	<op_peripherals>
08	Deactivate_peripheral	Yes	1	<op_peripherals>
09	Peripheral_status_ request	Yes	3	<NULL>
28	Peripheral_status	See text	3	<ip_peripherals><op_peripherals>

Table B-4
Peripheral messages

Msg No	Message name	Ack req	Priority	Structure
50	<ACK>	No	varies	<NULL>
51	<NAK>	No	varies	<count><destination><reason_code_set><reason_code>

Table B-5
Message protocol messages

Msg No	Message name	Ack req	Priority	Structure
10	Reset_request	Yes	3	<reset_type>
30	Reset	No	3	<reset_reason>
42	Alert_status	See text	3	<alerter_status>
43	Alert_eng	Yes	3	<alerter_engineering>
60	Set_parameter	Yes	varies	<parameter_table><parameter_no>(<index><parameter_value>)
61	Parameter_request	Yes	varies	<parameter_table><parameter_no>
62	Parameter	No	varies	<more_values><parameter>
63	Param_req_multiple	Yes	varies	<parameter_table><parameter_no> <first_entry><last_entry>
64	Test	Yes	varies	<test_type>
65	Printer_status	No	3	<printer_status>
66	MTA_status_change	Yes	3	<MTA_status>
67	Route_status	Yes	3	<boolean><destination_nodes>

Table B-6
Network management messages

Msg No	Message name	Ack req	Priority	Structure
04	Area_page_message	No	varies	<pager_priority><pager_number><pager_text>
06	Mobilise_message	yes	1	As defined by Hampshire and Kent Brigades
100	Supplier_message	varies	4	(<binary>)
101	Brigade_message	varies	3	<text>
102	Data_base_query	yes	3	<query_type><text>
103	Formatted_text	yes	varies	<format_type><table>
104	Proforma_definition_query	yes	varies	<format_type>
105	Proforma_definition	yes	varies	<format_type><table>

Table B-7
Non-Mandatory messages

B.1.8 The following sections contain descriptions of all the message types given in the tables above. For each message type the format is:

- the name of the message;
- the format of the message, showing the data fields of which it is composed;
- the way in which the message is used;
- the response to be made to the successful receipt of the message.

B Message content definitions

B.2 Message 01: Mobilise_command

B.2.1 Format

<op_peripherals><man_ack_req>

B.2.2 Usage

B.2.2.1 This is the general outstation mobilisation command, and prepares the station for the receipt of a mobilisation message by sounding the station sounders etc. The <op_peripherals> is a bit-orientated message indicating which (if any) of the appliance indicators, station sounders etc are required to be turned on. Setting a bit implies that the device is to be activated.

B.2.2.2 If <man_ack_req> is true then a manual acknowledgement is required.

B.2.3 Action on receiving message

B.2.3.1 On receiving a mobilise command message, the UA shall verify that it has peripherals corresponding to each of the functions to be activated. If it does not, it shall return a <NAK> for each function which it does not support. The fields of the <NAK> shall be set with a <reason_code_set> of “peripherals” and the appropriate <reason_code> for each function which cannot be performed.

B.2.3.2 The UA shall then activate each peripheral according to the peripheral *op_map* table which defines for each function how long the output is to be asserted for. If an output is marked to be asserted for a pulse length of 0, the relevant output shall be toggled (ie change state) rather than being pulsed.

B.2.3.3 Where an output has a corresponding input circuit it shall be monitored to confirm that the required output is functioning.

B.2.3.4 If the relevant feedback is a delayed response (eg it requires a manual intervention), a <NAK> shall be returned to inform the sender of the message that it has been received and is being processed. The fields of the <NAK> shall be set with a <reason_code_set> of “general” and <reason_code> of “wait_ack”.

B.2.3.5 If the relevant feedback fails, the UA shall return a <NAK> for each output which fails. If the output has a delay response and no feedback is received, the UA should wait an appropriate period before returning a <NAK>. The fields of the <NAK> shall be set with a <reason_code_set> of “peripherals” and the appropriate <reason_code> for the function which fails to operate.

B.2.3.6 If some of the peripherals have operated correctly and the message requires a manual acknowledgement, the UA shall send back a <NAK>. The <reason_code_set> shall be “general” and the <reason_code> shall be “wait_ack” (if it has not already sent a “wait_ack”).

B.2.3.7 Finally, when all functions associated with the message have finished (either successfully or unsuccessfully), the UA shall return an <ACK>. A given function is regarded as finished when either:

- it has been successfully completed;
- it has failed to complete according to an instantaneous feedback mechanism;
- it has been manually acknowledged.

B.3 Message 02: Mobilise_message

B.3.1 Format

<block><of_blocks><man_ack_req><time_and_date><callsign_list>(<incident_ number> <mobilisation_type> <address><map_ref> <tel_number> <text >)

B.3.2 Usage

B.3.2.1 This message is used to send the text of a mobilising message.

B.3.2.2 The fields of this message are used in the following way:

- <block> and <of_blocks> are used to allow long mobilise messages (eg for batch mobilisation) to be split into blocks for transmission;
- <man_ack_req> is true if a manual acknowledgement is required;
- <time_and_date> contains the time and date that the message was submitted for transmission;
- <callsign_list> is used to identify a number of individual appliances where the mobilise message is sent to (multiple) fire stations hosting multiple appliances;
- <incident_number> is used to uniquely identify each incident;
- <mobilisation_type> defines the type of the incident (eg test, pre-alert etc);

- <address>, <map_ref> and <tel_number> define the address, location and contact telephone number of the incident;

- <text> supplies optional additional information.

The definition of a repeating set of fields for this message allows it to be used for batch mobilisations.

B.3.2.3 Blocks shall be sent sequentially and no more than three blocks shall ever be unacknowledged.

B.3.2.4 Multi-block messages should be treated as separate messages for the purposes of acknowledgements. For multi-block messages requiring manual acknowledgement, the last block shall have <man_ack_req> set to true. All other blocks will not have <man_ack_req> set.

B.3.3 Action on receiving message

B.3.3.1 The UA shall identify each field within the message and print appropriate headers and prompts so that the output is clear and easy to read. The printed information shall include the date and time of submission of the message in a suitable format (extracted from the message contents).

B.3.3.2 The UA shall check that the source address of the message matches the *default_src* parameter stored by the UA. If this is not the case, the UA shall convert the message source address to a suitable ASCII string by reference to the *alternative_src* table, and shall print the relevant ASCII string as part of the message header information. This function is intended to ensure that the recipients of the message are aware that it has come from a source other than the normal mobilising system.

B.3.3.3 If the message cannot be printed or displayed for some reason, the UA shall respond by sending a <NAK>. If the UA is a printer UA, the <reason_code_set> will be “printer” and the <reason_code> will be "no_paper" or "off_line".

B.3.3.4 If the message requires a manual acknowledgement, the UA shall send back a <NAK>. The <reason_code_set> shall be “general” and the <reason_code> shall be “wait_ack”.

B.3.3.5 If the message does not require a manual acknowledgement, or when the UA receives a manual acknowledgement, it shall return an <ACK>.

B.3.3.6 If the UA receives further blocks after sending a <NAK> indicating a message has not been displayed/printed it may send a <NAK> with a <reason_code_set> of “general” and a <reason_code> of “abort”. The source UA will not transmit any additional blocks after receiving an abort (although there may still be further blocks which have been sent but not yet received).

B Message content definitions

B.4 Message 03: Page_officer

B.4.1 Format

<pager_priority><pager_number><pager_text>

B.4.2 Usage

B.4.2.1 This command activates the paging of an individual officer. The fields of this message provide flexibility to support tone or alphanumeric pagers.

B.4.2.2 <pager_priority> is intended for use by Home Office alerter systems. This field shall be ignored for other types of pager UA.

B.4.2.3 <pager_number> identifies the pager. For some public paging systems, this will be the telephone number to be dialled.

B.4.2.4 <pager_text> is a text message to be transmitted to a pager.

B.4.2.5 Not all fields will be usable by all UAs associated with paging.

B.4.3 Action on receiving message

B.4.3.1 On receiving this message, the UA shall verify that the parameters received are valid for that particular UA. For example, if the UA provides an interface to a tone paging system, then it should verify that no <pager_text> has been provided. If the parameters are not valid, it shall return a <NAK>. The fields of the <NAK> shall be set with a <reason_code_set> of "general" and a <reason_code> of “invalid_param”.

B.4.3.2 If the paging system cannot action the command and obtain an acknowledgement immediately, it shall return a <NAK>. The fields of the <NAK> shall be set as described in appendix D with a <reason_code_set> of "alerter" and a <reason_code> of "wait_ack".

B.4.3.3 The alerter unit shall then proceed to initiate the paging call and if possible verify that it has been transmitted. If this operation succeeds, it shall send an <ACK>.

B.4.3.4 If the paging call is not successful, the UA shall send a <NAK> with a <reason_code_set> of "alerter" and the relevant <reason_code>.

B.4.3.5 The codes which may be generated thus are: “tx_a_fail”, “tx_b_fail”, “tx_fail”, “enc_fail”, “power_fail”, “other_fail”. The codes for these messages are given in the tables of reason codes. In the case of systems with two transmitters, “tx_fail” shall only be sent if both transmitters a and b have failed and need not be accompanied

by <NAK>s carrying “tx_a_fail” and “tx_b_fail” codes. The UAs for other types of paging/alerter systems may not be able to generate the full set of codes; in particular, UAs sending data to be broadcast by a public system are only likely to be able to return the “other_fail” code. The codes for these messages are given in the tables of reason codes.

B.5 Message 05: Resource_status_request

B.5.1 Format

(<callsign>)

B.5.2 Usage

B.5.2.1 This message is used either to ask a resource to send its current status or to interrogate a resource database or other system. In the latter case, the inclusion of call sign as a repeating field is to allow the message to be sent to a single address which knows about multiple call signs. Alternatively, a call sign string with length 0 shall cause the receiving UA to return information about all of the resources known to it.

B.5.2.2 If the UA responding to the message is associated with only a single call sign, it may ignore the call sign parameter.

B.5.3 Action on receiving message

B.5.3.1 If the UA receiving the message is associated with only a single call sign and is able to respond, it shall do so with a Resource_status message. If it is not able to respond, it shall reply with a <NAK>. The <reason_code_set> shall be "general" with a <reason_code> of “inv_call”, “avl_fail” or “status_fail”.

B.5.3.2 If the UA receiving the message is associated with multiple call signs and the message requests the status of multiple resources, the Resource_status message sent by the UA in response shall contain a status entry for each of the resources listed in the request message (even if the entry is set to “No status data”).

B.6 Message 07: Activate_peripheral

B.6.1 Format

<op_peripherals>

B.6.2 Usage

B.6.2.1 The <op_peripherals> is a bit-orientated message indicating which (if any) of the appliance indicators, station sounders etc are required to be turned on. Setting a bit implies that the device is to be activated. The outputs are left activated for a preset period or until a deactivate command is received.

B.6.3 Action on receiving message

B.6.3.1 The UA shall verify that it has peripherals corresponding to each of the functions to be activated. If it does not, it shall return a <NAK> for each function which it does not support. The fields of the <NAK> shall be set with a <reason_code_set> of "peripherals" and the appropriate <reason_code> for each function which cannot be performed.

B.6.3.2 The UA shall then activate each peripheral according to the peripheral op_map table which defines for each function which output line is to be asserted and for how long. It shall not verify that the operation has been performed. If an output is marked to be asserted for a pulse length of 0, the relevant output shall be toggled (ie change state) rather than being pulsed.

B.6.3.3 The UA shall then return an <ACK>.

B.7 Message 08: Deactivate_peripheral

B.7.1 Format

<op_peripherals>

B.7.2 Usage

B.7.2.1 This message requests the remote station to deactivate the contacts specified in <op_peripherals>.

B.7.3 Action on receiving message

B.7.3.1 The UA shall verify that it has peripherals corresponding to each of the functions to be activated. If it does not, it shall return a <NAK> for each function which it does not support. The fields of the <NAK> shall be set with a <reason_code_set> of "peripherals" and the appropriate <reason_code> for each function which cannot be performed.

B.7.3.2 The UA shall then deactivate each peripheral according to the peripheral *op_map* table which defines for each function which output line is to be deasserted. It shall not verify that the operation has been performed.

B.7.3.3 The UA shall then return an <ACK>.

B.8 Message 09: Peripheral_status_request

B.8.1 Format

<NULL>

B.8.2 Usage

B.8.2.1 This message requests the remote station to indicate the current status of the peripherals which it is monitoring.

B.8.3 Action on receiving message

B.8.3.1 If this message is understood the UA shall respond with a Peripheral_status message.

B.9 Message 10: Reset_request

B.9.1 Format

<reset_type>

B.9.2 Usage

B.9.2.1 This message is a request for a remote node to perform a reset, and would be addressed to the router at that node. The <reset_type> indicates the 'severity' of the requested reset.

B.9.3 Action on receiving message

B.9.3.1 The remote comms processor should respond to the successful receipt of a Reset_request message which it is able to carry out by sending an <ACK>. The actual implementation of the reset is dependent upon the node receiving the

message.

B.10 Message 20: Resource_status

B.10.1 Format

(<callsign><AVL_type><AVL_data><status_code><remarks>)

B.10.2 Usage

B.10.2.1 This message enables a resource or UA to inform a remote UA about its status.

B.10.2.2 The message may be sent either in response to a <resource_status_request> message or unsolicited from a resource. If the message is being sent in response to a <resource_status_request>, it shall be sent with <ack_req> cleared. If it is being sent unsolicited, then it is optional whether the <ack_req> bit is set.

B.10.2.3 As detailed in the data dictionary, the <AVL_type> and <AVL_data> fields are included for compatibility with future automatic vehicle location systems, although coding for the <AVL_data> field cannot be defined at present.

B.10.2.4 The status of multiple resources may be returned within a single message (limited by the maximum size of the message).

B.10.3 Action on receiving message

B.10.3.1 If the <ack_req> bit is not set, then the UA shall not provide any response.

B.10.3.2 If the <ack_req> bit is set, then:

- the UA shall return a <NAK> with a <reason_code_set> of "general" and a <reason_code> of "inv_mess" if it does not understand the message;

- the UA shall return an <ACK> if it does understand the message.

B.11 Message 21: Duty_staffing_update

B.11.1 Format

(<callsign><OIC><riders><status_code><remarks>)

B.11.2 Usage

B.11.2.1 This unsolicited message allows the manning details and status of mobile resource(s) (identified by callsign) to be entered remotely.

B.11.2.2 It is optional whether the message is sent with <ack_req> set.

B.11.3 Action on receiving message

B.11.3.1 If the <ack_req> bit is not set, then the UA shall not provide any response.

B.11.3.2 If the <ack_req> bit is set, then:

- the UA shall return a <NAK> with a <reason_code_set> of "general" and a <reason_code> of "inv_mess" if it does not understand the message;

- the UA shall return an <ACK> if it does understand the message.

B.12 Message 22: Log_update

B.12.1 Format

<callsign><incident_number><update>

B.12.2 Usage

B.12.2.1 This message is used to update an incident log. The callsign indicates who is updating the log and the <incident_number> indicates which incident is to be updated.

B.12.3 Action on receiving message

B.12.3.1 The recipient shall acknowledge the successful receipt of a valid Log_update message from a resource by returning an <ACK>.

B.12.3.2 The precise interpretation of this message is left to the UA. However, it is likely that it will add the text message with the callsign of the resource sending it and the time of receipt to the incident log.

B.12.3.3 The UA receiving the Log_update message may decline to accept it. In this case, it shall return a <NAK> with a <reason_code_set> of “general” and a <reason_code> of “not accepted”.

B.13 Message 23: Stop

B.13.1 Format

<callsign><incident_number><stop_code>

B.13.2 Usage

B.13.2.1 This message is sent to control by a resource when the incident is stopped, to prevent the deployment of more resources. This message is sent without the textual Stop message (which can follow as a Log_update message) in order to expedite issuance of recall messages to other resources, so preventing any further deployment of resources to an incident.

B.13.3 Action on receiving message

B.13.3.1 The successful receipt of a valid Stop message shall be acknowledged with an <ACK> message.

B.13.3.2 The UA receiving the Stop message may decline to accept it (eg because the callsign of the sender of the Stop message is not assigned to the incident). In this case, the UA shall return a <NAK> message with a <reason_code_set> of "general" and a <reason_code> of "not_accepted”.

B.14 Message 24: Make-up

B.14.1 Format

<callsign><incident_number><number_types>(<appl_type><appl_quantity>)

B.14.2 Usage

B.14.2.1 This message allows a user, identified by his callsign, to request a number of additional types of appliances to Make-up for the incident to which he is currently assigned. The message allows up to 16 different types of appliance to be specified, each with its own quantity.

B.14.3 Action on receiving message

B.14.3.1 The receipt of this message shall be signalled to the sender by returning an <ACK> if the message is being actioned.

B.14.3.2 The UA receiving the Make_up message may decline to accept it (eg because the callsign of the sender of the message is not assigned to the incident). In this case, the UA shall return a <NAK> message with a <reason_code_set> of "general" and a <reason_code> of "not_accepted".

B.15 Message 25: Interrupt_request

B.15.1 Format

<callsign><request_code><text>

B.15.2 Usage

B.15.2.1 This message indicates either a request to speak or an emergency.

B.15.3 Action on receiving message

B.15.3.1 Upon receipt of this message, the recipient shall acknowledge the request with an <ACK>.

B.16 Message 27: Text_message

B.16.1 Format

<block><of_blocks><text>

B.16.2 Usage

B.16.2.1 This message is used to send text between nodes (eg to a printer). Blocks shall be sent sequentially and no more than three blocks shall ever be unacknowledged.

B.16.3 Action on receiving message

B.16.3.1 If the message is correctly received, the UA shall return an <ACK>. Messages shall

be printed in order according to the block number. Messages shall be displayed / printed as soon as possible. The UA shall indicate when the last block has been printed.

B.16.3.2 If the message cannot be printed or displayed, the UA shall respond by sending a <NAK>. If the UA is a printer UA , the <reason_code_set> will be “printer” and the <reason_code> will be “no_paper” or “off_line”.

B.16.3.3 If the UA receives further blocks after sending a <NAK> indicating a message has not been displayed / printed it may send a <NAK> with a <reason_code_set> of “general” and a <reason_code> of “abort”. The source UA will not transmit any additional blocks after receiving an abort (although there may still be further blocks which have been sent but not yet received).

B.17 Message 28: Peripheral_status

B.17.1 Format

<ip_peripherals><op_peripherals>

B.17.2 Usage

B.17.2.1 This message is sent to inform another UA about the state of a set of peripherals. This message is either sent asynchronously in reply to a Peripheral_status_request message from the control or may be sent unsolicited.

B.17.2.2 <ip_peripherals> defines the status of the functional input lines. A bit will be set if the input is asserted. A bit will be undefined if the input function is not assigned.

B.17.2.3 <op_peripherals> defines the status of the functional output lines. A bit will be set if the UA considers that the output is asserted. A bit will be undefined if the output function is not assigned.

B.17.2.4 If the message is being sent in response to a previous message, then the application dependent fields of the envelope shall be set as follows:

- <seq> shall have the same sequence number as in the original message;
- the <ack_req> bit shall be cleared;
- <priority> shall be set to the same value as in the incoming message.

B.17.2.5 If the message is being sent unsolicited, then the application dependent fields of the envelope shall be set as follows:

- <seq> shall be set according to the standard rules;
- <ack_req> shall be set;
- <priority> shall be set to 3.

B.17.3 Action on receiving message

B.17.3.1 The receipt of this message shall be acknowledged with an <ACK> message if <ack_req> was set.

B.17.3.2 The extent of any further action is unspecified.

B.18 Message 30: Reset

B.18.1 Format

<reset_reason>

B.18.2 Usage

B.18.2.1 This message indicates that the UA has performed a reset, either in response to a Reset_request message, after a reset caused by a software failure, or after the unit has been powered up.

B.18.3 Action on receiving message

B.18.3.1 The successful receipt of this message at the control does not need to be acknowledged with an <ACK>. The extent of any further action is unspecified.

B.19 Message 31: Incident_notification

B.19.1 Format

<alarm_type><call_agency><tel_number><alarm_ref><alarm_serial><address><text>

<alarm_type> indicates the type of incident.

<call_agency> identifies the company forwarding the call.

<tel_number> identifies the telephone number of the person reporting the call or alternatively a contact number for the incident.

<alarm_ref> can give a unique reference for the alarm installation.

<alarm_serial> will be a message reference generated by the node sending the incident. If this node is another mobs system, it will be its incident number for the incident. If this node is an alarm agency, it will be a reference number by which the incident can be identified.

<address> identifies the address of the incident.

<text> provides any additional information associated with an incident, eg time.

B.19.2 Usage

B.19.2.1 This message is provided in order to facilitate the connection of automatic fire alarm collecting centres to Fire Brigade systems. Alternatively, it may be used to pass messages between mobs systems. This message will be sent from an alarm company's premises directly to the mobilising system.

B.19.3 Action on receiving message

B.19.3.1 If the message is received and is being considered, the UA shall return a <NAK> with the <reason_code_set> to "general" and the <reason_code> set to "wait_ack". This response shall be followed by a further <NAK> or <ACK> as described below.

B.19.3.2 If the message is received and is accepted, the UA shall return an <ACK>.

B.19.3.3 If the message is received but is not accepted, the UA shall return a <NAK> with the <reason_code_set> to "general" and the <reason_code> set to "not_accepted".

B.19.3.4 It is beyond the scope of the protocol to determine how UAs shall respond to these different circumstances.

B.20 Message 40: Alert_crew

B.20.1 Format

<alert_group><man_ack_req><op_peripherals>

B.20.2 Usage

B.20.2.1 This command activates the alerting of one or more groups of retained or wholetime firemen in readiness for a mobilisation. The alert group specifies which group(s) should be alerted. It may also be used to activate peripherals (eg lights) in conjunction with the alert message. Note that the message must be addressed to both the Alerter UA and the peripheral UA if the peripherals are to be activated as part of the Alert_crew message.

B.20.2.2 If <man_ack_req> is true then a manual acknowledgement is required.

B.20.3 Action on receiving message

Action of Alerter UA

B.20.3.1 If the alerting system cannot action the command and obtain an acknowledgement immediately (ie because it requires a manual acknowledgement), it shall return a <NAK>. The fields of the <NAK> shall be set with a <reason_code_set> of “general” and a <reason_code> of “wait_ack”.

B.20.3.2 The alerter unit shall then proceed to initiate the alerter call and, if possible, verify that it has been transmitted. If this operation succeeds then:

- if it requires a manual acknowledgement it shall return an <ACK> when this acknowledgement is received;

- if it does not require a manual acknowledgement it shall return an <ACK> immediately.

B.20.3.3 The codes which may be thus generated are the same as those given above for the Page_officer message, with the addition of the codes “gr_a_fail”, “gr_b_fail”, “gr_c_fail”. If multiple group transmissions fail, the “other_fail” code shall be sent, not three <NAK>s carrying three “gr_fail” codes. The codes for these messages are given in the tables of reason codes.

Action of Peripheral UA

B.20.3.4 On receiving an Alert_crew message the UA shall verify that it has peripherals corresponding to each of the functions to be activated. If it does not, it shall return a <NAK> for each function which it does not support. The fields of the <NAK> shall be set with a <reason_code_set> of “peripherals” and the appropriate <reason_code> for each function that cannot be performed.

B.20.3.5 The UA shall then activate each peripheral according to the peripheral *op_map* table which defines for each function how long the output is to be asserted for. If an output is marked to be asserted for a pulse length of 0, the relevant output shall be toggled (ie change state) rather than being asserted.

B.20.3.6 Where a output has a corresponding input circuit it shall be monitored to conform that the required output is functioning.

B20.3.7 If the relevant feedback is instantaneous but fails, the UA shall return a <NAK> for each output which fails. The fields of the <NAK> shall be set with a <reason_code_set> of "peripherals" and the appropriate <reason_code> for the function which fails to operate.

B20.3.8 Finally, when all the functions associated with the message have finished (either successfully or unsuccessfully), the UA shall return an <ACK> . A given function is regarded as finished when either:

- it has been successfully completed,
- it has failed to complete according to an instantaneous feedback mechanism.

B.21 Message 42: Alert_status

B.21.1 Format

<alerter_status>

B.21.2 Usage

B.21.2.1 <alerter_status> indicates the status of the alerter. The codes used by <alerter_status> are defined in the data dictionary.

B.21.2.2 The Alert_status message shall be returned if the UA becomes aware of a change in the status conditions.

B.21.2.3 The application-dependent fields of the envelope shall be set as follows:

- <seq> shall be set according to the standard rules;
- <ack_req> shall be set;
- <priority> shall be set to 3.

B.21.3 Action on receiving message

B.21.3.1 The receipt of this message shall be acknowledged with an <ACK> message if

<ack_req> was set.

B.21.3.2 The action to be taken on receipt of this message with regard to updating the databases and informing operators at the control is not specified.

B.22 Message 43: Alert_eng

B.22.1 Format

<alerter_engineering>

B.22.2 Usage

B.22.2.1 This command allows engineering commands to be remotely sent to an alerting system.

B.22.2.2 The range of the commands which may be accepted by other systems is system dependent and cannot be specified here. <alerter_engineering> specifies the command to be applied.

B.22.3 Action on receiving message

B.22.3.1 If the requested command is not available, the UA receiving a command which it cannot support shall return a <NAK> with a <reason_code-set> of "alerter" and a <reason_code> of “no_fac”.

B.22.3.2 A UA which can handle the requested function shall issue the required command to the alerter. If this is successful, it shall return an <ACK>.

B.22.3.3 If it is not successful, it shall issue a <NAK> with a <reason_code_set> of "alerter" and a <reason_code> from the alerter table.

B.23 Message 50: <ACK>

B.23.1 Format

<NULL>

B.23.2 Usage

B.23.2.1 This message is a solicited response sent to indicate that a message has been received and processed.

B.23.2.2 The fields within the envelope of the message are the same as for any other message. However, for clarity:

- <source> defines the unique address of the MTA or UA which received and processed a message;
- <dest_count> specifies the number of destination addresses which are included in the envelope (which will always be 1 for <ACK>s);
- <destination> is set to be the same as the source of the original message;
- <length> will always be 0.

B.23.2.3 The application dependent fields of the envelope shall be set as follows:

- <seq> shall have the same sequence number as the in the original message;
- the <ack_req> bit shall be cleared;
- <priority> shall be set to the same value as in the incoming message.

B.23.3 Action on receiving message

B.23.3.1 On receiving an <ACK>, the UA shall remove the message being acknowledged (as determined by examining the source of the <ACK> and the sequence number) from the queue of unacknowledged messages.

B.23.3.2 A message sent to multiple destinations shall only be considered as fully acknowledged when all of the destinations have acknowledged the message. Similarly, if the UA times out and re-transmits a message, it shall only be re-transmitted to those destinations which have not acknowledged

B.23.3.3 The action to be taken on receiving an <ACK> for a message which does not appear in the list of unacknowledged messages is undefined.

B Message content definitions

B.24 Message 51: <NAK> (Negative acknowledge)

B.24.1 Format

<count><destination><reason_code_set><reason_code>

B.24.2 Usage

B.24.2.1 This message can be sent by either a router or a UA to indicate that a particular message could not be delivered or acted upon or that it has been delivered but an <ACK> could not be sent back immediately. It is sent back to the original sender of a message.

B.24.2.2 The fields within the envelope of the message are the same as for any other message. However, for clarity:

- <source> defines the unique address of the router or UA which is generating the <NAK>;
- <dest_count> specifies the number of destination addresses which are included in the envelope (which will always be 1 for a <NAK>);
- <destination> is set to be the same as the source of the original message;
- <length> is the number of bytes in the following message string.

B.24.2.3 The application dependent fields of the envelope shall be set as follows:

- <seq> will have the same sequence number as in the original message;
- the <ack_req> bit will be cleared;
- <priority> will be set to the same value as in the incoming message.

B.24.2.4 Within the body of the message, the fields will be set as follows:

- <count> will be set to the number of destination addresses in the message which caused the <NAK>;
- <destination> will list each of the destination addresses from the original message;

- the <reason_code_set> will define the class of reasons why the <NAK> has been generated (eg is was a router problem);

- <reason_code> defines the specific reason within the set.

B.24.2.5 The actual values to be used for <reason_code_set> and <reason_code> are defined in appendix D.

B.24.3 Action on receiving message

B.24.3.1 On receiving a <NAK> with a <reason code_set> of "general" and a <reason_code> of "wait_ack", the UA shall cancel any timer which is set up to detect a communications timeout. It may, depending upon the application, start a timer to generate a timeout if a further acknowledgement is not received.

B.24.3.2 The action to take on receiving a <NAK> with a different code is largely application dependent.

B.24.3.3 Contractors shall propose their own functionality.

B.25 Message 60: Set_parameter

B.25.1 Format

<parameter_table><parameter_no>(<index><parameter_value>)

B.25.2 Usage

B.25.2.1 This message is used whenever a UA wishes to change a parameter in another node.

B.25.2.2 The <parameter_table> field indicates which of the parameter tables is to be modified (see appendix C);

B.25.2.3 The <parameter_no> field indicates which of the parameters within that table is to be modified.

B.25.2.4 The <parameter_value> indicates the value to which the parameter should be set. The type of this field is dependent upon the type of the parameter being changed (as defined in the tables in appendix C). If the parameter being changed is a table then the entries in the table to be changed shall be indicated by <index>. Although

theoretically possible, it is assumed that there will never be a requirement to set a single parameter longer than 1023 characters.

B.25.3 Action on receiving message

B.25.3.1 On receiving such a message, the UA shall verify, in the following sequence:

- the node address;
- the password level;
- the format of the parameter.

B.25.3.2 The source address of the message attempting to modify a parameter shall be compared to the address stored within the current password field. If the address does not match, the MTA shall send back a <NAK> as defined below.

B.25.3.3 The current password level as defined by the current password parameter shall be compared to the password level required to set the parameter. If the current password level is not sufficient to allow the password to be changed, the MTA shall send back a <NAK> as defined below.

B.25.3.4 The final check is that the parameter value is appropriate for the parameter being set. The MTA shall check that:

- the syntax of the data is appropriate (eg if the data value is meant to be a string, it should contain only valid characters and the first byte should give the length);
- the value of the data is within the permitted range.

B.25.3.5 If the data is not valid, the MTA shall send back a <NAK> with the appropriate <reason_code>.

B.25.3.6 If all of the previous tests are completed successfully, the parameter shall be changed to the new value.

Termination

B.25.3.7 This message shall be completed as follows:

- if either an error occurred in the access protection, or the specified <parameter_table> field is invalid, or the parameter syntax is wrong, then the MTA shall return a <NAK> with the appropriate <reason_code>;
- if the parameter was successfully changed, the MTA shall send back an <ACK>

message.

B.26 Message 61: Parameter_request

B.26.1 Format

<parameter_table><parameter_no>

B.26.2 Usage

B.26.2.1 This message is used to read the value of a specific parameter held in a node.

B.26.2.2 The <parameter_table> indicates which of the parameter tables is to be used.

B.26.2.3 The <parameter_no> field indicates which of the parameters within that table is to be returned.

B.26.3 Action on receiving message

B.26.3.1 On receiving such a message, the UA shall verify that the <parameter_no> is valid. If the parameter does not exist, the UA shall send back a <NAK> as defined below.

Termination

B.26.3.2 An acknowledgement shall always be sent in response to this message. The acknowledgement shall take the following form:

- if an error occurred, the MTA shall return a <NAK> with the appropriate codes from the parameter table;
- if the request to see the parameter is valid, the MTA shall send a Parameter message.

B.27 Message 62: Parameter

B.27.1 Format

<more_values><parameter_value>

B.27.2 Usage

B.27.2.1 This message is used to return a parameter value in response to a Parameter_request message.

B.27.2.2 The application dependent fields of the envelope shall be set as follows:

- <ack_req> bit shall be cleared;
- <priority> shall be set to the priority of the message requesting the information;
- <seq> shall be set to the same value as in the incoming message.

B.27.2.3 The <parameter_value> is the value of the requested parameter. Its form is dependent upon the type of the parameter as defined in appendix C.

B.27.2.4 <more_values> shall be set to true if the UA cannot fit all of the requested parameters in the returned message (ie in response to a param_req_multiple message).

B.27.3 Action on receiving message

B.27.3.1 This message should only be received in response to a Parameter_request message. The UA receiving the message can establish which message (and hence which parameter) it has received by examining the <source> and <seq> of the received message. The receipt of this message shall remove the corresponding Parameter_request message from the queue of unacknowledged messages.

B.27.3.2 The manner in which the information is then used is application dependent.

B.28 Message 63: Param_req_multiple

B.28.1 Format

<parameter_table><parameter_no><first_entry><last_entry>

B.28.2 Usage

B.28.2.1 The purpose of this message is to access multiple data items which are stored under the same parameter (eg statistics) (see appendix C).

B.28.2.2 <parameter_no> defines the parameter within the current table.

B.28.2.3 <first_entry> defines the first entry to be retrieved. If this parameter is set to zero, then the first entry to be retrieved is the most recent one (or if the parameter table being accessed does not contain date and time then the first entries in the table).

B.28.2.4 <last_entry> defines the last entry to be retrieved. If <first_entry> was set to zero, then <last_entry> defines the number of parameters to be retrieved.

B.28.2.5 The <parameter_table> field indicates which of the parameter tables is to be used.

B.28.3 Action on receiving message

B.28.3.1 If the requested parameters exist, then a parameter message shall be returned which will contain as many of the requested parameters as possible, limited by the size of the parameter message.

B.28.3.2 If it is not possible to fit all of the requested parameters in the returned parameter message, then the requesting node may send a further request for additional parameters.

B.28.3.3 If the requested parameters do not exist, then the UA shall return a <NAK> with a <reason_set_code> of "param" and a <reason_code> of “inv_entry”.

B.28.3.4 The <parameter_table> field indicates which of the parameter tables is to be used.

B.29 Message 64: Test

B.29.1 Format

<test_type>

B.29.2 Usage

B.29.2.1 The purpose of this message is to enable remote tests to be carried out.

B.29.2.2 The <test_type> field specifies which test should be carried out.

B.29.3 Action on receiving message

B.29.3.1 If the test is supported by the MTA receiving the message, it shall return a <NAK>. The <reason_code_set> shall be “general” and the <reason_code> shall be "wait_ack".

B.29.3.2 The MTA shall then automatically carry out the test and establish the result. This will typically involve applying the loop and generating a test sequence. It shall then return an <ACK> if the test was successful or a <NAK> if the test failed. In the latter case, the <reason_code_set> shall be "general" and the <reason_code> shall be "test_fail".

B.29.3.3 If the test is not supported, the MTA shall return a <NAK>. The <reason_code_set> shall be "general" and the <reason_code> shall be "inv_test".

B.29.3.4 If the test cannot be carried out (eg the MTA is transmitting real frames) the MTA should return a <NAK>. The <reason_code_set> shall be “general” and the <reason_code> shall be “not accepted”.

B.30 Message 65: Printer_status

B.30.1 Format

<printer_status>

B.30.2 Usage

B.30.2.1 Where the notify_printer_available flag is set, this message shall be sent by a printer UA to inform the network management UA that the printer has gone off-line.

B.30.2.2 If the printer returns to an on-line state with notify_printer_available flag set then a printer_status message should be sent with <printer_status> set to on-line.

B.30.3 Action on receiving message

B.30.3.1 The receipt of this message shall be acknowledged with an <ACK> message if <ack_req> was set. Further action to be taken on receipt of this message is not specified.

B Message content definitions

B.31 Message 66: MTA_status_change

B.31.1 Format

<MTA_status>

B.31.2 Usage

B.31.2.1 This command indicates the status of an MTA has changed from being “off-line (fault)” to “on-line” (or vice versa).

B.31.3 Action on receiving message

B.31.3.1 The receipt of this message shall be acknowledged with an <ACK> message. The action to be taken on receipt of this message with regard to informing operators is not specified.

B.32 Message 67: Route_status

B.32.1 Format

<boolean><destination_nodes>

B.32.2 Usage

B.32.2.1 This message signifies to a receiving router that the routes to <destination_nodes>, via the node from which the message was received, are to be enabled or disabled.

B.32.2.2 If <boolean> is TRUE, the routes are to be enabled; otherwise; if <boolean> is FALSE, the routes are to be disabled.

B.32.2.3 <destination_nodes> indicates all those nodes to which routes via the source node are to be enabled or disabled.

B.32.3 Action on receiving message

B.32.3.1 On receiving this message, the router shall update its routing table to signify the current status of routes to the <destination_nodes> via the source node of the

message. This will be achieved as outlined in the paragraphs which follow.

B.32.3.2 If the routes are to be disabled, ie <boolean> is FALSE, the router shall examine its routing table and for each <next_node> corresponding to the source node of the message shall:

(a) identify those destination nodes associated with <next_node> that also belong to the <destination_nodes> parameter of the received message;

(b) for each such node the router shall record (either in the router table itself, or in a temporary table), that the route to this node, via the source node, has been disabled.

B32.3.3 If the routes are to be enabled, ie <boolean> is TRUE, the router shall examine its routing table and for each <next_node> corresponding to the source node of the message shall:

(a) identify those destination nodes associated with <next_node> that also belong to the <destination_nodes> parameter of the received message;

(b) for each such node, if the router has recorded that the route to this node, via the source node, has been disabled (see paragraph B.32.3.2), it shall remove this record (ie re-enable the route).

B32.3.4 Upon completion of the above actions, the Router shall return an ACK.

B.33 Message 04: Area_page_message

B.33.1 Format

<pager_priority><pager_number><pager_text>

B.33.2 Usage

B.33.2.1 This command activates the paging of an individual officer using the **brigade's private radio scheme as the primary bearer**. The fields of this message provide flexibility to support tone or alphanumeric pagers.

B.33.2.2 <pager_priority> is intended for use by Home Office alerter systems. This field shall be ignored for other types of pager UA.

B.33.2.3 <pager_number> identifies the pager.

B.33.2.4 <paper_text> is a text message to be tradsmitted to a pager.

B33.2.5 Not all fields will be usable by all UAs associted with paging.

B.33.3 Action on receiving message

Primary Radio bearer

B.33.3.1 On receiving this message, the UA shall verify that the parameters received are valid for that particular UA. For example, if the UA provides an interface to a tone paging system, then it should verify that no <pager_text> has been provided. If the parameters are not valid, it shall return a <NAK>. The fields of the <NAK> shall be set with a <reason_code_set> of "general" and a <reason_code> of "invalid_param".

B.33.3.2 If parameters are valid, the alerter unit shall then proceed to initiate the paging call and if possible verify that it has been transmitted. If this operation succeeds, **it shall not send an <ACK>**.

B.33.3.3 If the paging call is not successful, the UA shall send a <NAK> with a <reason_code_set> of "alerter" and the relevant <reason_code>.

B.33.3.4 The codes which may be generated thus are: "tx_a_fail", "tx_b_fail", "tx_fail", "enc_fail", "power_fail", "other_fail". The codes for these messages are given in the tables of reason codes. In the case of systems with two transmitters, "tx_fail" shall only be sent if both transmitters a and b have failed and need not be accompanied by <NAK>s carrying "tx_a_fail" and "tx_b_fail" codes.

Secondary (PSTN) bearer

B33.3.5 As defined in message 03 Page_officer.

B.34 Message 06: Mobilise_message

B.34.1 Format

Not defined

B.34.2 Usage

B.34.2.1 This message is used by the Hampshire and Kent brigades to send the text of a mobilising messages that are structured to their requirements.

B.34.2.2 Blocks shall be sent sequentially and no more than three blocks shall ever be

unacknowledged.

B.34.2.3 Multi-block messages should be treated as separate messages for the purposes of acknowledgements. For mult-block messages requiring manual acknowledgement, the last block shall have <man_ack_req> set to true. All other blocks will not have <man_ack_req> set.

B.34.3 Action on receiving message

B.34.3.1 As defined in message 02 Mobilise_message.

B.35 Message 100: Supplier_message

B.35.1 Format

(<binary>)

B.35.2 Usage

B.35.2.1 This message is used to send supplier data between nodes (e.g. to a router).

B.35.3 Action on receiving message

B.35.3.1 The receipt of this message shall be acknowledged with an <ACK> message if <ack_req> was set.

B.35.3.2 The action to be taken on receipt of this message is not specified.

B.36 Message 101: Brigade_message

B36.1 Format

<text>

B.36.2 Usage

B.36.2.1 This message is used to send non-operational text between nodes (e.g. to a printer). This message will primarily be used over private radio bearers.

B.36.3 Action on receiving message

B.36.3.1 The receipt of this message by the UA shall **not** be acknowledged with an <ACK> even if <ack_req> was set.

B.36.3.2 If the message cannot be printed or displayed, the UA shall respond by sending a <NAK>. If the UA is a printer UA, the <reason_code_set> shall be “printer” and the <reason_code> will be “no-paper” or “off-line”.

B.37 Message 102: Data_base_query

B.37.1 Format

<query_type><text>

B.37.2 Usage

B.37.2.1 This message is used to remotely query data bases.

B.37.3 Action on receiving message

B.37.3.1 The receipt of this message shall be acknowledged with an <ACK> message.

B.37.3.2 If the required data cannot be found, the UA receiving the request shall return a <NAK> with a <reason_code_set> of “general” and a <reason_code> of “data_fail”.

B.38 Message 103: Formatted_text

B.38.1 Format

<format_type><table>

B.38.2 Usage

B.38.2.1 This message allows user-data within locally defined pro-formas (or tables) to be sent, without headings, in response to a SQL message.

B.38.3 Action on receiving message

B.38.3.1 If the mesage is correctly received, the UA shall return an <ACK>. Messages shall be displayed / printed as soon as possible.

B.38.3.2 If the message cannot be printed or displayed, the UA shall respond by sending a <NAK>. If the UA is a printer UA, the <reason_code_set> will be “printer” and the <reason_code> will be “no_paper” or “off_line”.

B.39 Message 104: Proforma_definition_enquiry

B.39.1 Format

<format_type>

B.39.2 Usage

B.39.2.1 This message allows a UA, e.g. printer UA, to enquire about the layout of a new (unknown) pro-forma.

B.39.3 Action on receiving message

B.39.3.1 The receipt of this message shall be acknowledged with an <ACK> message.

B.39.3.2 If the requested pro-forma layout exists, then a pro-forma definition message shall be returned which will contain all the requested details.

B.39.3.3 If the requested pro-forma layout does not exist, then the UA shall return a <NAK> with a <reason_code_set> of “general” and a <reason_code> of “proform_fail”.

B.40 Message 105: Proforma_definition

B.40.1 <format_type><table>

B.40.2 Usage

B.40.2.1 This message conveys the details of a pro-forma’s layout on the receipt of a request from a UA, e.g. printer UA.

B.40.3 Action on receiving message

B.40.3.1 The receipt of this message shall be acknowledged with an <ACK>.

B.40.3.2 The action to be taken on receipt of this message is not specified.

C Parameter values

C.1 Introduction

C.1.1 This appendix provides details of the parameters within the system.

C.1.2 There are three versions of each parameter as defined in section C.2 and each parameter is protected by a password as described in section C.3. Section C.4 provides further general rules for handling parameters. Subsequent sections within the appendix list the individual parameters. The meaning and use of each parameter is described within the relevant section of the specification.

C.2 Parameter tables

C.2.1 There are three types of parameter table which are:

- the permanent table: which is a set of parameters which cannot be modified through software;
- the non-volatile table: which is a set of parameters which can be modified through software but which are not lost when the equipment is powered down;
- the current table: which is the set of parameters which are currently being used. The values of these parameters can be changed through software but the values will be lost when the equipment is powered down.

C.2.2 The permanent table provides the basic set of parameters which are set when the equipment is installed. These parameters are subsequently used as a fall-back set in the event that the other sets become corrupted.

C.2.3 The equipment shall be provided with a means of powering up which copies these default parameters to both the non-volatile and the current table.

C.2.4 On a normal power up, or following a software reset, the non-volatile set of parameters is copied to the current table and is used by the equipment. It is therefore possible to modify this table subsequently but not to have the parameters take effect until the equipment is reset.

C.2.5 The current set of parameters are those which are in use by a node. It is possible to modify these parameters without affecting the non-volatile tables.

C Parameter values

C.3 Password level

C.3.1 Each parameter is given a password level which defines who may modify a parameter.

C.3.2 The different levels of authority are:

- the Contractor (level 4);
- the maintainer / manager (level 3);
- the network supervisor (level 2);
- users (level 1).

C.3.3 These levels of authority are referred to as password levels.

C.3.4 Before modifying a parameter, it is necessary to set the current password (the process of logging on is defined in section 4.4). The address that was supplied may then modify any parameter at or below the level corresponding to the password supplied (eg the Contractor can modify any parameter).

C.3.5 Having modified a parameter, the network address should log-off the node as defined in section 4.4.

C.3.6 A parameter may be read by anyone (irrespective of password levels) apart from where noted in the parameter tables.

C.4 General information on parameters

C.4.1 Where parameters are acting as counters, upon reaching their maximum value, they shall remain at that value.

C.4.2 Parameters acting as counters may be reset using the <set_parameter> message.

C.4.3 Where an arbitrary number of instances of a parameter are required (eg a parameter stored for each message transferred), they shall all be stored under a common parameter. A further field within that parameter defines which instance of it is to be read (using the <parameter_req> message).

C.4.4 In practice, Contractors shall provide a finite number of slots for such parameters. When all slots are full, further entries shall overwrite the oldest previous entries.

C.5 Parameters associated with a router

No	Name	Password level	Type	Description
1	brigade_number	Level 3	<word8>	The Brigade number
2	node_number	Level 3	<word16>	The node number
3	node_name	Level 3	<node_name>	The name of the node
4	current_password	None	<pass_param>	Current password
5	level1_pass	Level 1	<password>	Access level 1 password
6	level2_pass	Level 2	<password>	Access level 2 password
7	level3_pass	Level 3	<password>	Access level 3 password
8	level4_pass	Level 4	<password>	Access level 4 password
9	max_length	Level 2	<word16>	Max_length
10	NW_mgr_address1	Level 2	<comms_ address>	Address of preferred network manager
11	NW_mgr_address2	Level 2	<comms_ address>	Address of alternative network manager
12	no_ack_timeout	Level 3	<word8>	Application level timeout(s)
13	router_table	Level 2	routing_table	This is a table of up to 200 entries on which the router bases its decisions.
14	PSTN_table	Level 2	PSTN_table	This is a table of up to 200 telephone numbers used for PSTN calls by the node.
15	WAN_table	Level 2	WAN_table	This is a table of up to ~~20~~ 200 WAN addresses used by the node.

No	Name	Password level	Type	Description
16	LAN_table	Level 2	LAN_table	This is a table of up to 200 LAN addresses used by the node.
17	ISDN_table	Level 2	ISDN_table	This is a table of up to 200 telephone numbers used for ISDN calls by the node.
18	man_ack_timeout	Level 3	<word16>	Time in seconds before it is assumed that a manual acknowledgement will not be received.
19	retries	Level 2	<word8>	Number of attempts to send a message allowed.
20	time_and_date	Level 2	<time_and_ date>	The current time and date.
21	MDT_table	Level 2	MDT_table	This is a table of up to 200 user addresses used by the node.
100 - 149	Spare	User definable	User definable	User definable

C.6 Parameters associated with all MTAs

No	Name	Password level	Type	Description
1	port_number	Level 3	<port_number>	The port number within the node
2	agent_type	Level 3	<agent_type>	The type of the MTA
3	interface_status	Level 3	<MTA_status>	The interface status
4	notify_status_ changes	Level 2	<boolean>	If true then inform the network manager about status changes
5	frame_tx	Level 2	<word16>	The number of frames transmitted
6	frame_rx	Level 2	<word16>	The number of frames received
7	frame_tx_failures	Level 2	<word16>	The number of frames that a failure occurred during transmission
8	frame_rx_failures	Level 2	<word16>	The number of frames that a failure occurred during reception
9	priority	Level 2	<word8>	The minimum level of priority of messages that can be carried by this MTA
10	next_nodes	Level 2	<destination_ nodes>	Indicates the nodes to which this MTA is currently connected
100 - 149	Spare	User definable	User definable	User definable

C.7 Parameters associated with a PW MTA

No	Name	Password level	Type	Description
21	verify_period	Level 2	<word16>	Time in seconds which is allowed before activity is created on the link
22	verify_timeout	Level 2	<word16>	Time in seconds which is allowed before deciding that a link has failed
23	frame_duration	Level 2	<word8>	Maximum period in seconds which is allowed for sending/receiving a frame
24	frame_timeout	Level 2	<word8>	Maximum period in seconds for a frame acknowledge to be returned
25	retries_allowed	Level 2	<word8>	The number of retries which are allowed

C8 Parameters associated with a asynch MTA

No	Name	Password level	Type	Description
21	verify_period	Level 2	<word16>	Time is seconds which is allowed before activity is created on the link.
22	verify_timeout	Level 2	<word16>	Time in seconds which is allowed before deciding that a link has failed.
23	frame_duration	Level 2	<word8>	Maximum period in seconds which is allowed for sending/receiving a frame.
24	frame_timeout	Level 2	<word8>	Maximum period in seconds for a frame acknowledge to be returned.
25	retries_allowed	Level 2	<word8>	The number of retries which are allowed.
26	ack_type	Level 3	<ASCII>	The acknowledgement character to be used for transmission.

C.9 Parameters associated with a PSTN MTA

No	Name	Password level	Type	Description
21	frame_duration	Level 2	<word8>	Maximum period in seconds which is allowed for sending/receiving a frame
22	frame_timeout	Level 2	<word8>	Maximum period in seconds for a frame acknowledge to be returned
23	retries_allowed	Level 2	<byte>	The number of retries which are allowed
24	dial_tones	Level 2	<dial_tones>	DTMF/pulse dialling
25	my_tel_no	Level 2	<tel_number>	The telephone number for this MTA
26	connect_stats	Level 2	<tel_stats>	Call record details

C.10 Parameters associated with a WAN MTA

No	Name	Password level	Type	Description
21	my_WAN_address	Level 2	<WAN_address>	WAN address of this MTA
22	connect_stats	Level 2	<WAN_stats>	Connection details

C.11 Parameters associated with a LAN MTA

No	Name	Password level	Type	Description
21	my_LAN_address	Level 2	<LAN_address>	LAN address of this MTA

C.12 Parameters associated with an ISDN MTA

No	Name	Password level	Type	Description
21	frame_duration	level 2	<word8>	Maximum period in seconds which is allowed for sending/receiving a frame
22	frame_timeout	level 2	<word8>	Maximum period in seconds for a frame acknowledge to be returned
23	retries_allowed	level 2	<word8>	The number of retries which are allowed
24	my_tel_no	Level 2	<tel_number>	The telephone number for this MTA
25	connect_stats	Level 2	<tel_stats>	Call record details

C.13 Parameters associated with all UAs

No	Name	Password level	Type	Description
1	port number	Level 3	<port_number>	The port number within the node
2	agent_type	Level 3	<agent_type>	The type of UA
3	control_address	Level 3	<comms_ address>	Destination for unsolicited messages
100 - 149	Spare	User definable	User definable	User definable

C.14 Parameters associated with peripheral UA

No	Name	Password level	Type	Description
21	station sounders	Level 3	<op_map>	The mapping from the function to a physical bit
22	station lights	Level 3	<op_map>	The mapping from the function to a physical bit
23	Spare	Level 3	<op_map>	The mapping from the function to a physical bit
24	Spare	Level 3	<op_map>	The mapping from the function to a physical bit
25	Spare	Level 3	<op_map>	The mapping from the function to a physical bit
26	station doors	Level 3	<op_map>	The mapping from the function to a physical bit
27-34	appliance indicators 1-8	Level 3	<op_map>	The mapping from the function to a physical bit
35	manual acknowledge pressed	Level 3	<ip_map>	The mapping from a function to a physical bit
36	power failed, batteries on	Level 3	<ip_map>	The mapping from a function to a physical bit
37	power failed, batteries on	Level 3	<ip_map>	The mapping from a function to a physical bit
38	repeat last message pressed	Level 3	<ip_map>	The mapping from a function to a physical bit

39	batteries low	Level 3	<ip_map>	The mapping from a function to a physical bit
40	paper low	Level 3	<ip_map>	The mapping from a function to a physical bit
41	alerter encoder output 1 ('E' relay operated)	Level 3	<ip_map>	The mapping from a function to a physical bit
42	alerter output 2	Level 3	<ip_map>	The mapping from a function to a physical bit
43	running_call_ telephone door opened	Level 3	<ip_map>	The mapping from a function to a physical bit
44	paper out	Level 3	<ip_map>	The mapping from a function to a physical bit
45	mobilising_printer	Level 2	<comms_ address>	
46	alerter_address	Level 2	<comms_ address>	Address of alerter UA for notification of manual acknowledgements
47	standby sounder	Level 3	<op_map>	The mapping from a function to a physical bit
48	spare op 7	Level 3	<op_map>	The mapping from a function to a physical bit
49 - 54	spare inputs 10 - 15	Level 3	<ip_map>	The mapping from a function to a physical bit
55	regen_time	Level 3	<word16>	number of seconds before repeat status

C.15 Parameters associated with printer UA

No	Name	Password level	Type	Description
21	default_src	Level 3	<address_range>	Address of the normal (default) source
22	notify_printer_ available	Level 3	<boolean>	Indicates whether changes in printer status should be notified to the network manager
23	alternative_src	Level 3	<address_table>	Contains up to 5 alternative address ranges
24	reprint_message	Level 2	<boolean>	Indicates whether the previous message should be printed first

D Reason codes

D.1 Alerter reason codes

Number	Reason code	Error description
1	tx_a_fail, 1	Transmitter a failed - message sent
2	tx_b_fail, 1	Transmitter b failed - message sent
3	tx_fail	Transmitters a and b failed - message not sent
4	gr_a_fail	Transmission to group a failed
5	gr_b_fail	Transmission to group b failed
6	gr_c_fail	Transmission to group c failed
7	enc_fail	Encoding unit failed - message not sent
8	power_fail	Power to alerter system failed - message not sent
9	other_fail	Alerter failed for other reason - message not sent
10	sys_busy	System has messages to transmit - engineering changes not made
11	no_fac	Alerter system is not specified to make the engineering changes requested
12	other_prob	Alerter system is unable to make the engineering changes requested for some other reason
13	wait_ack	Full <ACK> to follow
14	not_connected	No physical connection to alerter exists

D Reason codes

D.2 General reason codes

Number	Short name	Error description
1	no_bearer	No bearer available to destination
2	inv_param	Parameters are invalid for destination UA
3	inv_mess	Message type was invalid for destination address
4	avl_fail	Expected AVL data could not be found
5	status_fail	Expected status data could not be found
6	text_type	<Text_type> value of text message is not supported; message printed as though <Text_type> was zero.
7	inv_call	Callsign was invalid
8	not_accepted	The message was received but it was decided not to accept it
9	wait_ack	The message has been received and is being considered.
10	test_fail	The test has failed
11	inv_test	Specified test cannot be undertaken
12	inv_prot	Protocol is not understood by this user agent
13	check_error	Error during bcc test
14	abort	Do not send any further blocks
15	no_port	Router error if a port does not exist with the specified number
16	data_fail	Required data could not be found
17	proform_fail	Required pro-forma details could not be found

D Reason codes

D.3 Peripheral reason codes

Number	Short name	Error description
1	no_door	(De)activate_peripherals message referred to door operating equipment which the station does not have
2	no_sounder	(De)activate_peripherals message referred to sounder which the station does not have
3	no_lights	(De)activate_peripherals message referred to lights which the station does not have
4 - 11	no_ind1 - no_ind8	(De)activate_peripherals message referred to appliance indicator which the station does not have
12	door_fail	Door equipment could not be (de)activated
13	sounder_fail	Sounder could not be (de)activated
14	lights_fail	Lights could not be (de)activated
15 - 22	ind1_fail - ind8_fail	Appliance indicator could not be (de)activated
23	operation_fail	Peripheral equipment failure other than those defined above
24	no_dev_response	Attached peripheral equipment has not responded to instruction

D.4 Printer reason codes

Number	Short name	Error description
1	no_print_res	No response from printer
2	off_line	Printer was off line - message not printed
3	no_paper	Printer was out of paper - message not printed
4	low_paper	Printer paper low - message was printed
5	no_power	Printer not powered
6	no_connection	No physical connection to printer

D.5 Parameter reason codes

Number	Short name	Error description
1	no_mod_access	Address requesting parameter change does not have authority to modify parameter value
2	invalid_syntax	Setting information in a parameter setting command is not appropriate for parameter to be changed
3	invalid_value	Parameter information sent is outside permitted range
4	invalid_password	The password offered is not the correct password for the level of access being requested.
5	invalid_table	The parameter table information is invalid
6	invalid_param	The parameter number is invalid
7	invalid_field	Parameter field information sent is outside permitted range
8	inv_entry	The requested entry(ies) do not exist
9	no_access	unable to access the non_volatile storage medium

E Standard serial port

E.1 Electrical characteristics

The standard serial port shall comply with CCITT V.24/V.28 and have the following characteristics:

Mode	-	Asynchronous
Data Rate	-	9,600 bits per second
Frame format	-	8 data bits + parity + start + 1 stop
Parity	-	none [even or odd (switch selectable)]
Bit Order	-	LSB (D_0) first in time
Bit Sense	-	normal 1 = space (-ve) 0 = mark (+ve)
Flow Control	-	Out-of-band handshake - CTS and DTR

DTEs must cease data flow at the first frame boundary subsequent to CTS being de-asserted.

Circuits 101 (screen) and 102 (ground) must be implemented.

The following input circuits must be implemented: 104 (RxD), 106 (CTS).

The following output circuits must be implemented: 103 (TxD), 108 (DTR).

When connecting to a modem: the following additional circuits must be implemented: 105 (RTS), 107 (DSR), 109 (DCD).

Connectors must conform to ISO 2110 (25 pin 'D' type).

Ports on the communications processor and station processors must be configured as DTEs.

F Comparison with the OSI reference model

F.1 OSI is often used as a frame of reference for describing general-purpose communications systems since it provides a well understood division of the functions to be provided. In this case, however, the communications system is intended in the first instance to serve a single well-defined application in which short messages must be transferred reliably and quickly in 'real-time'.

F.2 For this reason, the various OSI standards and GOSIP profiles (such as FTAM or a full implementation of the X.400 MHS) are not of direct relevance to the overall operation of the system, since the overhead involved in conforming to these standards far outweighs the benefits derived.

F.3 At the same time, it is clearly important to exploit established industry and/or OSI standards for those elements of the system where these are appropriate, eg the CCITT V.22 standard for 1,200 bps data transmission over analogue telephone circuits.

F.4 Notwithstanding the comments above, it is instructive to compare the system model described in section 2 with the OSI reference model. Accordingly, the functions performed by the various elements in the communications system are described in general terms in the following paragraphs, in terms of each layer in the OSI model, as shown in figure F-1.

F.5 **Application Layer:** Application Layer functions may be divided into those concerned with message envelopes (equivalent to the X.400 MTAs) and those concerned with message contents (equivalent to the X.400 UAs). The former functionality is provided by the router at each node, while the latter functionality is provided by the various UAs.

F.6 In the present case, it is not particularly helpful to attempt to separate the stripped down set of X.400 MTA functions from those data transfer functions already provided at the Transport Layer. The combined set of X.400 MTA and Transport Layer functions are essentially provided by the routers.

F.7 **Presentation Layer:** The presentation layer is responsible for data compression and decompression. Since a common alphabet and message set is specified at the Application Layer, it need perform no other function. Presentation Layer functions may therefore be carried out by the UAs.

F.8 **Session Layer:** Since the system is intended to support a well-defined application by transferring short messages in a 'store and forward' fashion, no session control is required, and the Session Layer is therefore NULL.

F.9 **Transport Layer:** This layer provides for routing between sub-networks (inter network routing). Transport Layer functions include determination of the most appropriate sub-network for a given message, ensuring that message prioritisation is preserved, and any necessary retries of failed messages. These functions are carried out by the routers.

F Comparison with the OSI reference model

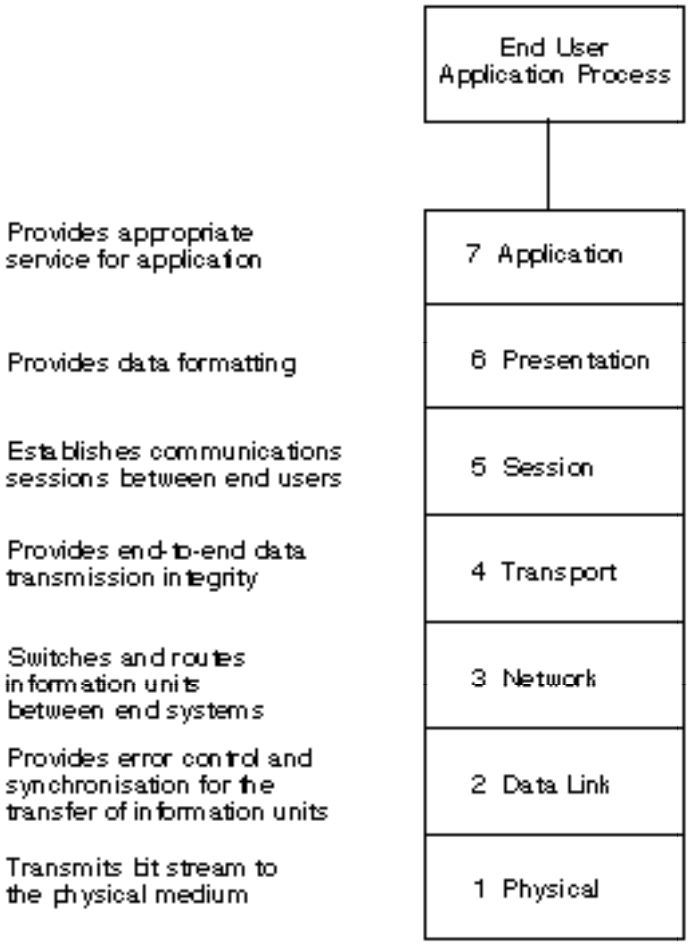

Figure F-1
OSI Reference Model

F.10 **Network, Data Link and Physical Layers:** These layers provide for the allocation of messages to physical bearers, for point to point communications with error indication and for any required bearer management and access control (eg auto-dialling for the PSTN). Network, Data Link and Physical Layer functions are carried out by the communications handlers and drivers at each node.

F.11 To summarise the above discussion, rigid adherence to the OSI model is not considered helpful in the definition of a communications system for Fire Service Mobilising Systems. It is preferable instead to focus on the concept of a message as comprising an envelope and a contents, and to examine the operations carried out by MTAs, UAs and routers in processing these separate components.