INTERNATIONAL FORECOURT STANDARD FORUM

STANDARD FORECOURT PROTOCOL

PART II

COMMUNICATION SPECIFICATION

Version 1.80 - February 2000

This document was written by the IFSF - Working Group:

Name	Company	Telephone
Dieter Claβen	Scheidt & Bachmann GmbH Breite Strasse 132 41238 Mönchengladbach Germany	+49/2166/266363
Arnaud de Ferry	Schlumberger Technologies	
Peter Maeers	BP Oil Europe Gloucester House, Langley Quay Waterside Drive, Langley Slough Berkshire SL3 6EY United Kingdom	+44/1753/549420
Eduardo Rezende	Shell	
Jürgen Wedemann	Deutsche Shell AG VTP/4 Überseering 35 22297 Hamburg Germany	+49/40/63246445

Further copies of this document can be obtained from:

IFSF Technical Services
PO Box 245
Chester
England
CH1 6ZL
Tel +44 151 347 2225
Fax +44 151 347 2573

The latest revision of this document can be downloaded from the Internet

address: www.ifsf.org

Document Contents

0 Record of Changes

Date	Version Number	Modifications
April 93	1.01	The changes from version 1.00 to 1.01 are very significant and therefore a listing of the changes is not necessary.

Date	Version number	Modifications
May 93	1.02	- Consistency of the document reviewed (characters type and size, paragraphs, table of content,...) - Integration of the CECOD communication document which defines layer 1 to 6 implementation.

Date	Version number	Modifications
June 93	1.3	- separate chapter 5 (Message structure) from the previous document release to this document Communication Specification. - possible Architectures and LON features are added. - "Message structure" description (chapter 5) was reorganised to increase the readability and updated: - The "Question Message" were renamed "Read Message". - The "Write Message" type was added. - "Acknowledge message" were completed to provide a detailed answer. - The destination for "Unsolicited message" was organised in database to provide "Read", "Add" and "Delete" commands.

Date	Version number	Modifications
October 93	1.40	**General** - Add document part number on title page - Correction of document footer name to "Standard Forecourt Protocol" - Add document authors on page 2 - The chapters are renumbered - The word "Originator" is used instead of Sender, Source - The word "Recipient" is used instead of Receiver, Destination - The word Data Element is used instead of Data Variable - Spelling errors are corrected
		Chapter 1 - System Architecture - Chapter 2 is renumbered to 1 - Information about application layer and network layer is added
		Chapter 2 - Communication Principle - Chapter 1 is renumbered to 2.1 - Add more examples for physical layers - Change baud to bit/s - LON uses a CSMA protocol - Information about block length is added - Chapter 3.1 is renumbered to 2.2. The chapter name is now "Communication Basics" - The message types "Acknowledge" is added - The message type "Unsolicited Data" is divided into message types "Unsolicited with ACK" and "Unsolicited without ACK" - The table "Typical Message Sequences" is moved from chapter 3.3 to chapter 2.2 - The message sequence "read from multiple databases" is added

Chapter 3 - Application Message Format

- The message frame has changed
- A description of every message field is added
- The address for a device and sub-device is divided into a logical node address (LNA) and a database address (DB_Ad)
- A device is selected by the LNA
- The LNA consist of 2 bytes (Subnet and node)
- LNAO is used for the originator (source)
- LNAR is used for the recipient (destination)
- The Database Address (DB_Ad) specifies the address in the device
- A Database Address Length (DB_Ad_Lg) is included
- A message code (MC) is included
- The abbreviation for block number is now BL
- The abbreviation for message status is now M_St
- The abbreviation for message length is now M_Lg
- A description of the data field is added (Data_Id, Data_Lg, Data_El)
- A table of the generic message frame is added (originator application message, network message, recipient application message)
- Message Status (M_St) description is now in chapter 3.2.1 (instead 3.3)
- A chapter (3.2.2) is added to describe the message length (M_Lg)
- A chapter (3.2.3) is added to describe the database address (DB_Ad) and the database address length (DB_Ad_Lg)
- A chapter (3.2.4) is added to describe the application data (Data)
- The description of message type "READ" is now in chapter 3.3.1 (instead of chapter 3.4). In conjunction with the changes
- The chapter numbering for the different message types has changed:
  (3.4 -> 3.3.1),(3.5 -> 3.3.2),(3.6 -> 3.3.3),(3.7 -> 3.3.4),
  (3.6 -> 3.3.5)
- In conjunction with the changes described before the frame for the different message types is changed
- Two additional status for Data_ACK are defined

Chapter 4 - Communication Services

- A chapter for network addressing is included
- A chapter with network parameter is included
- A Message Code field "MC" is included in the message. The description is in chapter 4.4.
- The "Cutting Message" mechanism is now described in chapter 4.3
- The description of the Communication Service Database is now in chapter 4.5
- The Communication Service Database contains the Data_Ids: *Communication_Protocol_Ver, Local_Node_Addr, Recipient_Addr_Table, Heartbeat_Interval, Max_Block_Length, Heartbeat_Error, Add_Recipient_Addr, Remove_Recipient_Addr*
- A description how to manage (read/write/add/remove) the Recipient_Addr_Table is included
- A description of Unsolicited Data handling is included
- A description of Heartbeat handling is included
- A description of Node Address Configuration is included

Date	Version number	Modifications
March 95	1.50	**General** - Any reference to the maximum value for the Max_Block_Length has been changed from 229 to 228. **Chapter 2 - Communication Principle** - The selected LON physical layer has been changed from Transformer Coupled to Free Form Topology (FTT-10). - An application level time-out of 8 seconds is introduced. This specifies the maximum time allowed between the receipt of a "Read Message" and the responding "Answer Message". The same time-out value applies for the messages "Write Message" & "Unsolicited Data Message with Acknowledge" and the responding "Acknowledge Message". **Chapter 3 - Application Message Format** - In chapter 3.1 additional comments on transmitting application frames that are larger than the Max_Block_Length and the correct network frame header. - In chapter 3.3.2 details of the handling of a read attempt on a Data_Id that is not readable in the current device state. **Chapter 4 - Communication Services** - In chapter 4.1 the non defined subnets 14 to 32 have been allocated for the exclusive use of the IFSF, and the subnets 33 to 127 have been allocated to the manufactures, oil companies and other LON devices. - New Subnet assignments (Printer, PIN Pad & Magnetic Card Reader), changes to all Subnet assignments of devices previously with a Subnet > 4. - Comment on addressing multiple devices via a single NEURON chip.

Date	Version number	Modifications
January 96	1.51	**General** - The **MC** (message code) has been changed to **IFSF_MC** to avoid confusion with the LONTALK message code. - All message examples have been changed to reflect the message on the network.
		Chapter 2 - Communication Principle - 2.1 LON Features Corrected reference to CSMA to P-Persistent CSMA protocol - 2.2 Communication Basics - Message Sequence Overview 'Acknowledge' message added to the 'Requesting data elements from multiple databases in a device' example.
		Chapter 3 - Application Message Format - 3.3.2 "Answer" - Message Additional text explaining that the 'Acknowledge' message generated when a read on multiple databases occurs must use the same token as the 'Read' message. - 3.3.4 "Acknowledge" - Message A new MS_ACK code of 6 added. This value indicates that the Write message had an invalid data base address. A new MS_ACK code of 7 added. This value indicates that the recipient device is 'Busy' and not able to process the received application message. A new DATA_ACK code of 6 added. This value indicates that the command has net been accepted.

Chapter 4 - Communication Services

- 4.1 Addressing

  Subnet 2 changed from POS to Site controller/forecourt Controller

  Subnet 15 added to allow Point Of Sales to be defined on the network

  Subnet 16 added to allow Back Office System to be defined on the network.

- 4.2 Network Parameters

  Number of LONTALK retries changed from 4 to 3.

  LONTALK TX_TIMER changed from 128 ms to 96 ms.

- 4.3 Message Code “IFSF_MC”

  Extra text explaining that the LONTALK message code should be the same as the ‘IFSF_MC’ code.

  LONTALK TX_TIMER changed from 128 ms to 96 ms.

  Extra text defining the IFSF recommended connectors & recepticals (Weidm ller BLZ).

- 4.5 Communication Service Database

  Data_Id 4 “Heartbeat_Interval” has additional text explaining the default setting after the device has been reset.

  Data_Id 5 “Max_Block_Length” has additional text explaining the default setting after the device has been reset.

- 4.5.1 Recipient Address Table

  Extra text explaining that devices should not remove addresses from the table when the respective device is deemed as being off-line.

- 4.5.3 Heartbeat

  New status added to the Heartbeat message that is used to indicate if the respective device needs to be configured or have its software downloaded.

Date	Version number	Modifications
April 1998	1.60	**Chapter 2 Communication Principle** 2.2 Communication basics - Text added to explain that an MS_ACK = 1 should be returned by the communications layer when there is no response within the required timeout. [IR058] **Chapter 3 Application Message Format** - 3.3.4 "Acknowledge" - Message Extra text added to explain the how to acknowledge a read from multiple data items within a database and also how to acknowledge a read when there is no answer message. [IR005] 3.3.4 "Acknowledge" - Message Addition of a new NAK to the Message Acknowledge Status table to allow the clear rejection of the entire message. [IR036] 3.3.5 "Unsolicited Data" - Message An example database entry has been added for unsolicited messages and an example of an unsolicited message is also given. This ensures that the reader understands that all values sent are given in hexadecimal, which is different to the decimal used for Data_Ids within the specifications. [IR048] 3.4 Concurrent message handling - added. These sections discuss the possible problems that can arise due to concurrent block cut messages, and gives implementation guidelines. 3.4.1 Ambiguity of Block Cut messges 3.4.2 Ambiguity of response messages 3.4.3 Cannot NACK Reads. [IR009]

		Chapter 4 Communication Service - 4.1 Addressing Additional subnet address allocations added to the list: 17 - Public Network Server 18 - Card Handling Server 19 - Human Interface Device (HID) 20 - Vehicle Identification 39 - Data logging [IR057] - 4.3 Message Code “IFSF_MC” Extra text explaining that LONTalk message code should **always** be the same as IFSF message code. [IR022] 4.5.1 Node address assignment Added - a standardised procedure for assigning node addresses, using a reserved node address of 127. [IR001] 4.5.4 Heartbeat A note has been added to clarify the use of the different heartbeat intervals on the same network. [IR 051] Extra text has been added to explain the meaning of Configuration Needed bit. [IR006]

Date	Version number	Modifications
November 1998	1.70	**Chapter 3** 3.3.4 “Acknowledge” - Message Typographical correction - MS_ACK range changed from 1 to 3 and 6 to 8. MS_ACK 4 (Message refused in this device state) removed from Message Acknowledge Status table. This has never been used within the self-certification tools and therefore has never been used in the field. It is threfore obsolete [IR004]. **Chapter 4** 4.1 Leak detection device subnet allocated to 21 (decimal). **Chapter 5** 5.1 Added implementation details for SC off-line [IR1017].
February 1999	1.71	**Chapter 4** **4.5.2** Added implementation guideline for Recipient Address Table entries [IR1018].

February 2000	1.80	**Chapter 2** **2.1.** Additional text added to clarify Lonworks Specific Network Parameters (IR1080). **2.2.** Added information should a recipient node be unable to respond (IR1027) & (IR1066). Text amended in “Unsolicited Data Message with Acknowledge” and “Unsolicited Data Message without Acknowledge” (IR1078). **Chapter 3** **3.3.4.** Amended explanation for READ all transactions from FP2 (IR1082). Added information should a recipient node receive data that is “out of boundary” (IR1067). **3.4.3.** Additional text added describing NACK reads (IR1027). **Chapter 4** **4.1.** Amended Subnet address table (IR1081) & (IR1048). **4.4.** Additional text added to clarify should blocks arrive out of sequence (IR1065). **4.5.** Heartbeat_Error text deleted (IR1075). **4.6.1.** Additional text added to support data encryption and source authentication (IR1044).

1 System Architecture

The protocol is defined to be able to support different configurations:

- **system using a single forecourt controller** which manages the different communications with the forecourt equipment (dispensers, outdoor payment terminals, price sign displays,..); this forecourt controller is able to store the dispenser transactions; if multiple Points of Sale (POS) are used, it is responsible for dispatching data between them;
- **decentralised system** in which the multiple POSs are directly connected to the bus, the dispenser and other controller devices themselves providing the management of their information (lock/unlock transactions,..).

The choice between these 2 solutions depends on:

- W&M rules (ability to buffer or not in the dispenser or in the forecourt controller,...);
- migrations (most existing systems are using the first architecture so they can easily implement it).

```
Architecture 1:
            ÚÄÄÄÄÄ¿    ÚÄÄÄÄÄ¿    ÚÄÄÄÄÄ¿    ÚÄÄÄÄÄ¿
            ³POS 1³    ³POS 2³    ³POS 3³    ³POS 4³
            ÀÄÄÂÄÄÙ    ÀÄÄÂÄÄÙ    ÀÄÄÂÄÄÙ    ÀÄÄÂÄÄÙ
               ÀÄÄÄÄÄÄÄÄÄÄÁÄÄÄÄÄÂÄÄÄÄÁÄÄÄÄÄÄÄÄÄÄÙ
                                ³
                         ÚÄÄÄÄÄÄÁÄÄÄÄÄÄ¿
                         ³ FORECOURT   ³
                         ³ CONTROLLER  ³
                         ÀÄÄÄÄÄÄÂÄÄÄÄÄÄÙ
   LON BUS                      ³
 ÄÄÄÄÄÄÄÄÄÄÂÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÂÄÄÁÄÄÄÄÄÄÄÄÄÄÄÄÂÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÂÄÄÄ
           ³                 ³               ³                 ³
      ÚÄÄÄÄÁÄÄÄÄ¿       ÚÄÄÄÄÁÄÄÄÄ¿      ÚÄÄÄÁÄÄÄ¿     ÚÄÄÄÄÄÄÄÁÄÄÄÄ¿
      ³DISPENSER³ . . . ³DISPENSER³      ³  OPT  ³     ³ MONOLITH / ³
      ³    1    ³       ³    n    ³      ³       ³     ³ POLE SIGN  ³
      ÀÄÄÄÄÄÄÄÄÄÙ       ÀÄÄÄÄÄÄÄÄÄÙ      ÀÄÄÄÄÄÄÄÙ     ÀÄÄÄÄÄÄÄÄÄÄÄÄÙ


Architecture 2:
            ÚÄÄÄÄÄ¿    ÚÄÄÄÄÄ¿    ÚÄÄÄÄÄ¿    ÚÄÄÄÄÄ¿
            ³POS 1³    ³POS 2³    ³POS 3³    ³POS 4³
            ÀÄÄÂÄÄÙ    ÀÄÄÂÄÄÙ    ÀÄÄÂÄÄÙ    ÀÄÄÂÄÄÙ
               ³          ³          ³          ³
   LON BUS     ³          ³          ³          ³
 ÄÄÄÄÄÄÄÄÄÄÂÄÄÄÁÄÄÄÄÄÄÄÄÄÄÁÄÄÂÄÄÄÄÄÄÄÁÄÄÄÄÄÄÄÂÄÄÁÄÄÄÄÄÄÄÄÄÄÄÄÄÄÂÄÄÄ
           ³                 ³               ³                 ³
      ÚÄÄÄÄÁÄÄÄÄ¿       ÚÄÄÄÄÁÄÄÄÄ¿      ÚÄÄÄÁÄÄÄ¿     ÚÄÄÄÄÄÄÄÁÄÄÄÄ¿
      ³DISPENSER³ . . . ³DISPENSER³      ³  OPT  ³     ³ MONOLITH / ³
      ³    1    ³       ³    n    ³      ³       ³     ³ POLE SIGN  ³
      ÀÄÄÄÄÄÄÄÄÄÙ       ÀÄÄÄÄÄÄÄÄÄÙ      ÀÄÄÄÄÄÄÄÙ     ÀÄÄÄÄÄÄÄÄÄÄÄÄÙ
```

Different points can be considered in the document to describe the exchanged messages:
- messages between the Originator Application and the Originator Communication layers,
- messages on the network,
- messages between the Recipient Communication layers and the Recipient Application.

For interoperability reasons only the message on the network have to be considered, but for explanations it is sometime useful to also consider the interface between the Application (layer 7) and the Communication (layers 1 to 6).

```
ÚÄÄÄÄÄÄÄÄÄÄÄÄÄ¿   ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿        ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿   ÚÄÄÄÄÄÄÄÄÄÄÄÄÄ¿
³Originator  ³   ³Originator     ³        ³Recipient    ³   ³Recipient   ³
³Application ÃÄÄÄ´Communication  ³        ³CommunicationÃÄÄÄ´Application ³
³  layer 7   ³   ³ layers (1-6)  ³        ³layers (1-6) ³   ³ layer 7    ³
ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÙ   ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ        ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ   ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÙ
                          ³                    ³
                          ³                    ³
                          ³    network         ³
                          ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ
```

2 Communication Principle

2.1 LON Features

The LON component selected to provide the ISO communications layers 1 to 6 is already used in the "building wiring/services" business. The very large usage of this component offers a powerful solution at an acceptable cost.

The LON is able to use different physical layers (layer 1):
- RS485 (max speed: 39 kbit/s)
- Transformers (78 kbit/s or 1.25 Mbit/s)
- Power Lines (PLT-20, Cenelec C-Band)
- Optical Fibres
- Short Range Radio
- Infrared Light
- Intrinsically Safe
- Free Form Topology (FTT-10 78 kbits/s)

LonWorks Specific Network Parameters

Addressing	-	the top level of the addressing hierarchy of a LonWorks network is a domain. If different applications are to share a communication network, they may be separated into different domains.
	-	the second level of addressing is the subnet. A subnet is a logical grouping of nodes from one or more channels. There may be up to 255 subnets per domain.
	-	the third level of addressing is the node. There may be up to 127 nodes per subnet.

The Lontalk protocol offers four basic message services these being:-

ACKD, REQUEST/RESPONSE, UNACKD_RPT, UNACKD

The service implemented on the IFSF network is ACKD. The ACKD service is were a message is sent to a node and individual acknowledgements are expected back from each recipient node. If the acknowledgements are not all received, the sender times out and retries the transaction. The number of retries and the time-out value are variable and are set along with the network parameters by application level code (layer 7 OSI model) see below:-

Priority Slot:	Determines if message has priority over other messages, improves response time of critical messages.
Retry:	Specifies the number of times a node will repeat sending a message if it doesn’t receive an acknowledgement.
RPT_Timer:	Specifies how frequently the message is repeated when using unacknowledged/repeated service.
RCV_Timer:	Started when a node receives a message.
TX_Timer:	Determines how long a node waits for an acknowledgement before retrying.
Non_Group_Timer:	Started when a node receives a message providing it can allocate a receive transaction record.

The **chosen standard for layer 1 is the Free Form Topology FTT-10 at 78 kbits/s**.
This high speed will offer a very good response time even in the worst conditions and allows the bus topology to be any combination of a 'star', 'loop' or 'bus'. Please note that the 2 pin Weidm ller BLZ (or equal) connectors and receptacles are recommended for connecting IFSF FTT-10 devices.

The LON uses a P-Persistence CSMA protocol (similar to Ethernet) and so no "master" is required on the bus.

The LON chip uses 3 processors and so the communications on the bus (managed by 2 of them) never interferes with the transfer of messages with the application (managed by the third).

The forecourt communication expects to use a LON component for layer 1 to 6, using "Explicit Messages".
The length of such messages is set by the *Max_Block_Length* communication data element (see chapter 4.5 "Communication Service Database"). The range for the block length is 32 to 228. To allow messages longer than this value, a Block number "BL" is included in the message (for details see chapter 4.4 "Block Cutting").

2.2 Communication Basics

To avoid specifying message contents where application dependent, the choice was to use a read/write mechanism.

Six basic messages are used to access any data in the system:
- Read Messages
- Answer Messages
- Write Message
- Unsolicited Data Message with Acknowledge
- Unsolicited Data Message without Acknowledge
- Acknowledge Message

A **"Read Message"** allows an originator device to read the value(s) of any system data element from a recipient device within a specified database. It is possible to read multiple data elements in the one command. This is achieved by requesting a list of data elements. So only the accessible data elements must be requested and the messages are application dependent. The recipient must respond to the "Read Message" with an "Answer Message" within the time-out period of 8 seconds. However, if a recipient node is unable to respond it must reply with an ACK. If for instance the device was busy it would reply with a MSG_ACK of NAK7.

An **"Answer Message"** is a message which provides to the originator of a "Read Message" all the requested data elements.

A **"Write Message"** is used to send data from an originator to a recipient device. The identifier of the data element is defined in the recipient device. It is possible to write a multiple data elements in the one command. This is achieved by issuing a message including a list of data elements and their values.
Sending a "Command" is considered as writing a data element with no value. The same message is able to send commands and data elements used as parameters.
The recipient must respond to the "Write Message" with an "Acknowledge Message" within the time-out period of 8 seconds.

An **"Unsolicited Data Message with Acknowledge"** is used for sending data elements. The recipient must respond to this message with an "Acknowledge Message" within the defined time-out of 8 seconds.
The only difference from a "Write Message" is that the data elements (database address, data identifier, data element contents) are defined in the originator device.

An **"Unsolicited Data Message without Acknowledge"** is used for sending data elements (e.g. status change, error) where the recipient must not respond to the message.

The **"Acknowledge Message"** is used by the recipient of a message (write, unsolicited with acknowledge) to respond.

Message Sequences Overview

MESSAGES SEQUENCES		
Originator		Recipient
Requesting data elements from a single database in a device		
Read Message	-->	
	<--	Answer Message
Requesting data elements from multiple databases in a device		
Read Message	-->	
	<--	Answer Message
	<--	Answer Message
		...
	<--	Answer Message
	<--	Acknowledge Message
Sending data elements to a device		
Write Message	-->	
	<--	Acknowledge Message
Sending unsolicited data elements with acknowledge		
Unsolicited Data Message	-->	
	<--	Acknowledge Message first Recipient
	<--	Acknowledge Message second Recipient
		...
	<--	Acknowledge Message last Recipient
Sending unsolicited data elements without acknowledge		

Unsolicited Data Message	-->	

NOTE - If a node cannot be reached with a Read or Write Message within the required 8 seconds, then a Acknowledge message will be returned by the communications layer with MS_ACK = 1 (Recipient node not reachable).

3 Application Message Format

3.1 Generic Message Frame

This chapter describes the generic message application message format to be used in conjunction with any device application (layer 7) utilising this communication layer.

Field description and definitions

Message Field	Number of Bytes	Description
LNAR	2	*Logical Node Address Recipient* This field is the LNA of the Recipient of the message.
LNAO	2	*Logical Node Address Originator* This field is the LNA of the Originator of the message.
IFSF_MC	1	*Message Code* The Message Code is used to filter the received data in the communication layer.
BL	1	*Block Number* For messages longer than the Max_Block_Length the communication layer cuts the message, assigning to each part of the message a sequential Block Number.
M_St	1	*Message Status* The M_St defines the type of the message and also contains the token.
M_Lg	2	*Message Length* The Message Length specifies the number of bytes (database, data) that follow in the message.
DB_Ad_Lg	1	*Database Address Length* Number of address bytes which are used to specify a database of the device.

DB_Ad		1-8	*Database Address* The Database Address specifies the database of the selected device. The database can be located in the Originator or in the Recipient according to the message type.
Data		n + 2 or n + 4	*Application Data* This is the application data which is sent between the originator and the recipient. This data is specified in the respective application documents.
	Data_Id	1	*Data Identifier* This is the Identifier of the application data element.
	Data_Lg	1 or 3	*Data Length* This is the length of the application data element. For data elements longer than 254 bytes, the Data_Lg will have the value 255 and the 2 following bytes indicate the data length.
	Data_El	n	*Data Elements* This is the contents of the application data element

Message on the Network	
LNAR Logical Node Address Recipient	**S** Subnet
	N Node
LNAO Logical Node Address Originator	**S** Subnet
	N Node
IFSF_MC IFSF Message Code	
BL Block	
M_Lg Message Length	
DB_Ad_Lg Database Address Length	
DB_Ad Database Address	
Data Application Data	**Data_Id** Data Identifier
	Data_Lg Data Length
	Data_El Data Element

The table above defines for a message sent from an application to an other application:
- the data sent by the originator application to the communication layers,
- the data frame on the network,
- the data received by the recipient application (layer 6 to layer 7).

Please note that when an application message is transmitted that has a length greater than the *Max_Block_Length* parameter, the application message will be split up and transmitted in several network data frames. In this case, all network data frames will have the standard LNAR, LNAO, IFSF_MC & BL header.

3.2 Application Message Fields

3.2.1 Message Status "M_St"

M_St byte is to indicate the kind of data which is carried by the message because different messages types are used for:
- "Read Message"
- "Answer Message"
- "Write Message"
- "Acknowledge Message"
- "Unsolicited Data Message with Acknowledge"
- "Unsolicited Data Message without Acknowledge"

M_St				
bit 8	bit 7	bit 6	bit 5 to 1	Description
0	0	0	token 0 -> 31	Read Message
0	0	1	"	Answer Message
0	1	0	"	Write Message
0	1	1	"	Unsolicited Data Message with Acknowledge
1	0	0	"	Unsolicited Data Message without Acknowledge
1	1	1	"	Acknowledge Message

The token is used by the message originator to be able to make the link between a received message and a message previously sent (question, command,...). The token value is chosen by the originator and returned by the recipient without any processing so the originator can always use the same token (if it doesn't need to use this link) or different value for each message.

A token is generated for all message types except an ANSWER or an ACKNOWLEDGE. These two types merely return the token they received in the message to which they are response.

3.2.2 Message Length "M_Lg"

The Message Length M_Lg is used to specify the number of bytes that follow in the message fields:
- Database Address Length "DB_Ad_Lg"
- Database Address "DB_Ad"
- Application Data "Data"

The Message Length is specified in two bytes (bin16).

3.2.3 Database Address "DB_Ad" and Length "DB_Ad_Lg"

Every data element in a device is stored in a database. In some implementations it may be real database or only a software organisation (object or tasks), for instance if a separate processor manages each meter. So the addressing mode must be flexible enough to address any database level.

These database levels are addressed by the Database Address (DB_Ad) using a variable number of bytes which is indicated in the Database Address Length (DB_Ad_Lg) field.
The number of address bytes to specify a database is 1 to 8.

Every forecourt application document (IFSF-document Part III) has a description showing how these addresses are specified.

The only common definition for every forecourt device is to use the first byte of the database address as follows:

00H	for Communication Service Data
01H	for Common Data in a device

DB_Ad							
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
00H Communi-cation Service	**00H-FFH** Application dependent	**00H-FFH** Application dependent	**00H-FFH** Application dependent	**00H-FFH** Application dependent	**00H-FFH** Application dependent	**00H-FFH** Application dependent	**00H-FFH** Application dependent
01H Common Data							
02H-FFH Application dependent							

3.2.4 Application Data "Data"

The Application Data consist of the Data Identifier "Data_Id", the Data Length "Data_Lg" and the Data Element "Data_El". All this data is specified in Application Document of each forecourt device (see IFSF document Part III).

3.3 Application Message Type

3.3.1 "Read" - Message

The general format for a READ message is:

Message Field		Field Type	Value	Description
LNAR	**S**	bin8	1 to 255	*Logical Node Address Recipient*
	N	bin8	1 to 127	
LNAO	**S**	bin8	1 to 255	*Logical Node Address Originator*
	N	bin8	1 to 127	
IFSF_MC		bin8	0	*Message Code* - the IFSF_MC is 0 for application messages
M_St		bin8	000xxxxx (bit map)	*Message Status* - the message type is READ - the token (xxxxx) is created by the originator of the message
M_Lg		bin16		*Message Length* - the number of bytes in the READ-message following
DB_Ad_Lg		bin8	1 to 8	*Database Address Length* - the number of address bytes which are used to select the database from where the data should be read
DB_Ad		x * bin8 (x = 1 to 8)		*Database Address* - the recipient database address from where the data should be read
Data_Id		bin8	1 to 255	*Data Identifier* - the identifier for the first requested data element in the recipients database

Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the second requested data element in the recipients database
. . .			
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the last requested data element in the recipients database

The data identifier "Data_Id" indicates the requested data fields in the addressed database "DB_Ad". The individual forecourt device is selected by the Logical Node Address "LNAR" (S,N).

3.3.2 "Answer" - Message

The general format for an ANSWER message is:

Message Field		Field Type	Value	Description
LNAR	**S**	bin8	1 to 255	*Logical Node Address Recipient*
	N	bin8	1 to 127	
LNAO	**S**	bin8	1 to 255	*Logical Node Address Originator*
	N	bin8	1 to 127	
IFSF_MC		bin8	0	*Message Code* - the IFSF_MC is 0 for application messages
M_St		bin8	001xxxxx (bit map)	*Message Status* - the message type is ANSWER - the token (xxxxx) is the same as the token generated by the originator of the READ-message
M_Lg		bin16		*Message Length* - the number of bytes in the ANSWER-message following
DB_Ad_Lg		bin8	1 to 8	*Database Address Length* - the number of address bytes which are used to specify from where the data has come
DB_Ad		x * bin8 (x = 1 to 8)		*Database Address* - the database address from where the data has come
Data_Id		bin8	1 to 255	*Data Identifier* - the identifier for the first requested application data element

Data_Lg	bin8 or bin8, bin16	0 to 254 or 255, 0 - 65535	*Data Length* - number of bytes for the first application data element contents - if the length is bigger than 254 see ***1** - if data element does not exist or can not be read in the current device state see ***2**
Data_El	appli- cation dependent	appli- cation dependent	*Data Element Contents* - the contents of the first requested application data element
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the second requested application data element
Data_Lg	bin8 or bin8, bin16	0 to 254 or 255, 0 - 65535	*Data Length* - number of bytes for the second application data element contents - if the length is bigger than 254 see ***1** - if data element does not exist or can not be read in the current device state see ***2**
Data_El	appli- cation dependent	appli- cation dependent	*Data Element Contents* - the contents of the second requested application data element
. . .			
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the last requested application data element
Data_Lg	bin8 or bin8, bin16	0 to 254 or 255, 0 - 65535	*Data Length* - number of bytes for the last application data element contents - if the length is bigger than 254 see ***1** - if data element does not exist or can not be read in the current device state see ***2**
Data_El	appli- cation dependent	appli- cation dependent	*Data Element Contents* - the contents of the last requested application data element

*[1]	For data longer than 254 bytes, the Data_Lg will have the value 255 and the 2 following bytes indicate the data length.
*[2]	If a Data_Id cannot be provided by an equipment (previous version or partial implementation) or a read on the Data_Id is not permitted in the current device state then the Data_Lg will be equal to zero and no DATA will be present.

For answers coming from the device, each set of data is identified in the answer by:
- A Database Address "DB_Ad" according to the preceding rule which identifies from which database data has come from
- and the data itself "Data_Id", "Data_Lg", "Data_El"

A READ-message requesting a single database will receive a single ANSWER-message. But a request for data from multiple similar databases will receive multiple answers in separate messages with the same token (the one of the originator message). The last answer will be followed by an ACKNOWLEDGE-message (with the same token as used in the preceding ANSWER-messages) to indicate to the originator of the READ-message the end of the multiple answers (otherwise the originator would have to wait for a time-out to establish the number of expected messages).

3.3.3 "Write" - Message

The general format for a WRITE message is:

Message Field		Field Type	Value	Description
LNAR	**S**	bin8	1 to 255	*Logical Node Address Recipient*
	N	bin8	1 to 127	
LNAO	**S**	bin8	1 to 255	*Logical Node Address Originator*
	N	bin8	1 to 127	
IFSF_MC		bin8	0	*Message Code* - the IFSF_MC is 0 for application messages
M_St		bin8	010xxxxx (bit map)	*Message Status* - the message type is WRITE - the token (xxxxx) is created by the originator of the message
M_Lg		bin16		*Message Length* - the number of bytes in the WRITE-message following
DB_Ad_Lg		bin8	1 to 8	*Database Address Length* - the number of bytes used to specify the database address where the data should be written to
DB_Ad		x * bin8 (x = 1 to 8)		*Database Address* - the recipient database address where the data should be written to
Data_Id		bin8	1 to 255	*Data Identifier* - the identifier for the first application data element

Data_Lg	bin8 or bin8, bin16	0 to 254 or 255, 0 - 65535	*Data Length* - number of bytes for the contents of the first application data element - if the length is bigger than 254 see ***[1]** - the length could be 0 (e.g. commands without data)
Data_El	appli- cation dependent	appli- cation dependent	*Data Element Contents* - the contents of the first application data element which shall be written to the specified database
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the second application data element
Data_Lg	bin8 or bin8, bin16	0 to 254 or 255, 0 - 65535	*Data Length* - number of bytes for the contents of the second application data element - if the length is bigger than 254 see ***[1]** - the length could be 0 (e.g. commands without data)
Data_El	appli- cation dependent	appli- cation dependent	*Data Element Contents* - the contents of the second application data element which shall be written to the specified database
. . .			
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the last application data element
Data_Lg	bin8 or bin8, bin16	0 to 254 or 255, 0 - 65535	*Data Length* - number of bytes for the contents of the last data element - if the length is bigger than 254 see ***[1]** - the length could be 0 (e.g. commands without data)

Data_El	appli-cation dependent	appli-cation dependent	*Data Element Contents* - the contents of the last application data element which shall be written to the specified database
***1**	For data longer than 254 bytes, the Data_Lg will have the value 255 and the 2 following bytes indicate the data length		

The data elements the recipient receives (in one message or in different messages) must be processed sequentially. That means that all parameter used by a command must be written before the command is transmitted (in the same message or the previous one).

3.3.4 "Acknowledge" - Message

The general format for an ACKNOWLEDGE message is:

Message Field		Field Type	Value	Description
LNAR	**S**	bin8	1 to 255	*Logical Node Address Recipient*
	N	bin8	1 to 127	
LNAO	**S**	bin8	1 to 255	*Logical Node Address Originator*
	N	bin8	1 to 127	
IFSF_MC		bin8	0	*Message Code* The IFSF_MC is 0 for application messages
M_St		bin8	111xxxxx (bit map)	*Message Status* - the message type is ACKNOWLEDGE - the token (xxxxx) is the same as used by the message which must be acknowledged
M_Lg		bin16		*Message Length* - the number of bytes in the ACKNOWLEDGE-message following
DB_Ad_Lg		bin8	1 to 8	*Database Address Length* - the number of bytes used to specify the database address from where the acknowledge is coming
DB_Ad		x * bin8 (x = 1 to 8)		*Database Address* - the database address from where the acknowledge is coming
MS_ACK		bin8	0 to 8	*Message Acknowledge Status* - the message acknowledge status byte (details see at the end of this table *****^{1})
If the MS_ACK has the value 0 to 3 or 6 to 8 no additional information is sent in the ACKNOWLEDGE-message.				

If some of the data from a received message is not acceptable the complete message will be refused (MS_ACK = 5) and the ACKNOWLEDGE-message will include the following detailed information. Please note that all valid writes to Data_Id's (i.e. with a Data_ACK of 0) must be applied to the data base.			
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the first application data element which was received
Data_ACK	bin8	0 to 7	*Data Acknowledge Status* - the data acknowledge status byte for the first application data element (details see at the end of this table ***2**)
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the second application data element which was received
Data_ACK	bin8	0 to 7	*Data Acknowledge Status* - the data acknowledge status byte for the second application data element (details see at the end of this table ***2**)
. . .			
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the last application data element which was received
Data_ACK	bin8	0 to 7	*Data Acknowledge Status* - the data acknowledge status byte for the last application data element (details see at the end of this table ***2**)

NOTE
In response to READ messages from multiple data items (within the same database of course) the database address of the ACKNOWLEDGE should be the same as the one sent in the READ message.

If there is no answer message, for example no outstanding transactions in a transactions database, then for the case of a dispenser (refer to dispenser specification)

READ all transactions from FP2:

01 02 02 01 00 80 15 00 0A 04 22 20 00 01 05 06 07 08 0A

ACK no transactions:

02 01 01 02 00 80 F5 00 06 04 22 20 00 01 00

Table *[1] ***Message Acknowledge Status***

MS_ACK	Description	
0	ACK	Positive acknowledge, data received
1	NAK	Recipient node not reachable
2	NAK	Application out of order or non existent on the recipient node
3	NAK	Inconsistent message (block missing)
5	NAK	Message refused, some of the data is not acceptable, detailed information follows
6	NAK	Message refused, unknown data base address (DB_Ad) or illegal write to multiple data base address.

7	NAK	Message refused, receiving device is busy and unable to accept the application message. The transmitting device should stop transmitting the remainding frames of the application message and retransmit the complete application message. A minimum of 2 retries should be attempted (i.e. application message transmitted 3 times in total) before the transmitting device can indicate that an error has occured. Please note that a 'busy' receiving device should generate the NAK response on the first frame of the transmitted application message. Any other frames (i.e. those with a block number > 1) should be ignored and no response is expected.
8	NAK	Message unexpected. This NAK acknowledges an unsolicited message with Acknowledge. This premits the clear rejection of the entire message.
9	NAK	Device already locked. This NAK is sent by a device in response to other devices attempting to communicate with them while they are being configured.

Table *[2] ***Data Acknowledge Status***

Data_ACK		Description
0	ACK	Positive acknowledge: data acceptable
1	NAK	Invalid value (too big/ too small)
2	NAK	Read only data or Not Writable
3	NAK	Command refused in that state
4	NAK	Data does not exist in this device
5	NAK	Command not understood / implemented
6	NAK	Command not accepted

NOTE
If a recipient node receives data which is too big or too small i.e. out of boundary, the node will reply with a NAK1. The value written should be then ignored and not updated in the node databases.

3.3.5 "Unsolicited Data" - Message

The general format for a UNSOLICITED DATA message is:

Message Field		Field Type	Value	Description
LNAR	**S**	bin8	1 to 255	*Logical Node Address Recipient*
	N	bin8	1 to 127	
LNAO	**S**	bin8	1 to 255	*Logical Node Address Originator*
	N	bin8	1 to 127	
IFSF_MC		bin8	0	*Message Code* - the IFSF_MC is 0 for application messages
M_St		bin8	yyyxxxxx (bit map)	*Message Status* - the message type (yyy) is 011 for unsolicited messages which must be acknowledged by the recipient - the message type (yyy) is 100 for unsolicited messages which must not be acknowledged - the token (xxxxx) is created by the originator of this message
M_Lg		bin16		*Message Length* - the number of bytes of the message following
DB_Ad_Lg		bin8	1 to 8	*Database Address Length* - the number of bytes used to specify the database address from where the data has come
DB_Ad		x * bin8 (x = 1 to 8)		*Database Address* - the database address from where the data are coming

Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the first application data element which shall be sent
Data_Lg	bin8 or bin8, bin16	0 to 254 or 255, 0 - 65535	*Data Length* - number of bytes for the contents of the first application data element - if the length is bigger than 254 see ***1** - the length could be 0 (e.g. commands without data)
Data_El	appli- cation dependent	appli- cation dependent	*Data Element Contents* - the contents of the first application data element which shall be sent
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the second application data element which shall be sent
Data_Lg	bin8 or bin8, bin16	0 to 254 or 255, 0 - 65535	*Data Length* - number of bytes for the contents of the second application data element - if the length is bigger than 254 see ***1** - the length could be 0 (e.g. commands without data)
Data_El	appli- cation dependent	appli- cation dependent	*Data Element Contents* - the contents of the second application data element which shall be sent
. . .			
Data_Id	bin8	1 to 255	*Data Identifier* - the identifier for the last application data element which shall be sent
Data_Lg	bin8 or bin8, bin16	0 to 254 or 255, 0 - 65535	*Data Length* - number of bytes for the contents of the last application data element - if the length is bigger than 254 see ***1** - the length could be 0 (e.g. commands without data)

Data_El	appli-cation dependent	appli-cation dependent	*Data Element Contents* - the contents of the last application data element which shall be sent
***1**	For data longer than 254 bytes, the Data_Lg will have the value 255 and the 2 following bytes indicate the data length		

These messages are generated by the application, but the actual recipient addresses of the messages are generated by the communication layer by using the *Recipient_Addr_Table*. For more details see chapter 4.5, 4.5.1 and 4.5.2.

The data elements the recipient receives must be processed sequentially. That means that all parameter used by a command must be written before the command is transmitted (in the same message or the previous one).

The following example shows the relationship between a database entry for an unsolicited message and the message returned.

EXAMPLE DATABASE

DB_Ad = 32H

Data_Id	***Variable Name*** Description	Field Type (Value)
CONFIGURATION		

100	***EX_Unsolicited_Message*** The EX_Unsolicited_Message includes: - EX_1 (Data_Id = 20) - EX_2 (Data_Id = 21) - EX_3 (Data_Id = 22) Please note that the EX_Unsolicited_Message Data_Id is built up as follows: 100,0,20,01,X,21,01,Y,22,02,Z,S where Data_Id, EX_1, EX_2 and EX_3 are expressed in decimal form. However, the message sent will be expressed in hexadecimal.	bin8, bin8, bin16

For the example given in the example database the values sent in the unsolicited message would be:

<header> 80 00 00 0E 01 32 64 00 14 01 XX 15 01 YY 16 02 ZZ SS

where all values are given in hexadecimal.

3.4 Concurrent Message Handling

If a device sends a message and receives an ACKNOWLEDGE or ANSWER before the next message is sent, then there will not be a problem with concurrent messages from a given device. However it is possible for concurrent messages to occur a) when a device sends another message before a response is received to a previous message, and b) multiple devices may send messages simultaneously to a given device so that it receives a number of messages concurrently, so that multiple devices each send multiple messages to a target device.

3.4.1 Ambiguity of Block Cut Messages

It is important that there should be no concurrent block cut messages from any one device since the header for a block cut message does not contain a token.

<MESSAGE HEADER> = <LNAR, LNAO, IFSF_MC, BL>

Thus it is not possible to recombine block cut messages correctly if there are concurrent block cut messages from a given device.

3.4.2 Ambiguity of Response Messages

There may be confusion about which particular Answer/Acknowledge response pertains to which original Read/Write message. Since some Read messages also receive Acknowledge messages, there can only be one sequence of tokens, not separate sequences for Reads and Writes. Hence the maximum number of outstanding messages from a given device should be 32, or less.

3.4.3 NACK Reads

There is a limit to the number of messages which a target device can buffer. In the case of messages overrunning the target device, it could NACK any messages beyond its capacity using MS_ACK = 07 (Device busy). The initiator should then wait for approximately 2 seconds and then repeat its message. A NACK read will also occur when it is not possible to read from a database location.

4 Communication Services

This chapter defines the services provided by the communications layer and the way to access to these services. That means the interface between the application (layer 7) and the communication (layer 6).

4.1 Addressing

Each device will be locally able to define its own node address that means: Subnet and Node numbers (S,N). Different solutions are possible:

- dip switches or similar mechanical solution,
- address provided by the application running in the host processor.

- **Domain:** 1 byte
  D : 1 for all forecourt equipment.
  This field is reserved for future extensions but not considered for the time being in our addressing capacity.

- **Subnet (S), Node address (N):** 2 bytes
  The choice is to use the Subnet address (S) to indicate the device type.

S :	1	Dispenser
	2	Site Controller/Forecourt Controller
	3	CHD - Printer
	4	CHD - PIN Pad
	5	CHD - Card Reader
	6	CHD - Card Acceptor
	7	CHD - Bank Note Acceptor (BNA)
	8	Price Pole
	9	Tank Level Gauge
	10	Car Wash
	11	Tanker Delivery Control
	12	Vending Machine
	13	Car Valet Machine
	14	Monitor Equipment
	15	Point of Sales (POS)
	16	Back Office System (BOS)
	17	CHD - Public Network Server
	18	CHD - Card Handling Server

19	Human Interface Device (HID)
20	Vehicle Indetification/Tagging
21	Environmental Monitoring Sensor
22	Line Leak Detector
23	Customer Operated Payment Terminal
24	CHD - Network Configuration Manager
25	Code Generator
26 - 32	Reserved for future use
39	Data logging
40 - 127	Free to the manufacturer or oil companies

The range that is “Reserved for future use” may not be used without prior authorisation of the IFSF committee.

N : All devices belonging to the same Subnet (so having the same type) are identified by a unique Node address (N). The values are 1 to 127.

- **Group** addressing facilities are not used.

Please note that the above Subnet/Node assignments must be implemented at the Application level. However, to allow multiple devices to be interfaced via a single Echelon Neuron chip (i.e. An OPT that consists of a PIN Pad, Card Reader, Receipt Printer & Journal Printer) it is permitted to use a different Subnet/Node address in the application message than the actual network Neuron chip Subnet/Node address.

Example:

When addressing a PIN Pad (S4:N1) that is connected to an OPT device with its Neuron chip configured as (S6:N1), the Transmitting Neuron chip will transmit at network level to the OPT device (S6:N1), but in the application message will use a Recipient Subnet address of 4 and a Recipient Node address of 1.

Where: S4:N4 = S4 is Subnet 4
N1 is Node 4.

4.2 Network Parameters

These parameters defines the way the NEURON chip capabilities are used in our system.

- **Priority slot:** 1 (allowing to create emergency messages if necessary)

- **Retry** = 3

- **RPT_Timer** : not used

- **RCV_Timer** : not used

- **TX_Timer** = 96 ms (LONTALK TX_TIMER value 5)

- **Non_Group_Timer** = 768 ms

- **Block Length**:

  The Block Length is set by the Max_Block_Length communication data element. The range for the block length is 32 to 228. To allow messages longer than this value a Block Number "BL" is included in the message.

- **Communication Services**

  All messages are point to point with acknowledge. If a message doesn't receive an acknowledge after the given number of retries, this is reported to the application which can take the decision (repeat, cancel, state change,...).

4.3 Messages Code "IFSF_MC"

The IFSF message code (IFSF_MC) can be used by the recieving device to easily filter received data.

Please note that the LONTALK Message code should always have the same value as the IFSF_MC. This enables the NUERON chip to filter the messages.

IFSF_MC	Definition
0	Message for the application, not handled by the NEURON
1	Heartbeat message
2	Message for the communication database

4.4 Block Cutting "BL"

For messages bigger than the maximum block length (Max_Block_Length), the communication layer cuts the message, assigning to each part of the message a block number. Blocks are numbered from n = 0 using sequential binary numbers and the last block number is identified as being the last by having its value increased by 128 (bit 8 = 1).

BL							
bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1
1 = last block 0 = intermediate blocks	not used		block number				

The maximum number of blocks is defined by the buffer size of the receiving device (around 1 Kbytes in the IFSF Dispenser Application) and the Block Length.

Example: If the Block Length is 32 the maximum number of blocks is 32 (numbered from 0 to 31).

If the Block Length is 228 the maximum number of blocks is 4 (numbered from 0 to 3).

When the message is cut, blocks are created by the communication layer according to the maximum possible block length on the network.

Blocks may arrive at a recipient node out of sequence. It is up to the recipient node application to ensure that this is taken into consideration when the message is reassembled.

4.5 Communication Service Database

This data allows the CD to specify the communication service data. The database address 00H (first byte of DB_Ad) in each device is used for communication services.

This database is accessible through the standard READ / WRITE - messages from local or remote devices. Except:

- The *Local_Node_Address* is only used to communicate with the local application as it is explained further.
- The *Communication_Protocol_Ver* is only readable.

COMMUNICATION SERVICE DATABASE **DB_Ad = 00H**		
Data_Id	***Variable Name*** Description	Field Type (Value)
CONFIGURATION		
1	***Communication_Protocol_Ver*** To allow the CD to read the version number of the communication protocol being used. The Communication_Protocol_Ver number format is '9999999999.99'.	bcd12
2	***Local_Node_Address*** To define the local Subnet address and node address.	bin8, bin8 (1-255, 1-127)
3	***Recipient_Addr_Table*** This table contains a list of all recipients that must receive unsolicited messages. The table able to have 64 recipient addresses. See Data_Id 11 and 12 for adding and removing from the table. Each recipient address is identified by 2 bytes: - Subnet = device type - Node = 1 to 127	x * bin8,bin8 x = < 64 (1-255, 1-127)

4	***Heartbeat_Interval*** Time in seconds to give a heartbeat message on the network. 0 = no heartbeat generated Please reference the respective IFSF Application Protocol description for details of the default *Heartbeat_Interval* to be used when device has been reset or the current *Heartbeat_Interval* value has been corrupted.	bin8 (0-255)
5	***Max_Block_Length*** The range for the block length is 32 to 228 byte. Messages which are longer are split into appropriate sized blocks. Please note that to achieve complete interoperability between IFSF devices utilising the LonWorks and standard LonWorks devices the *Max_Block_Length* must be set to 32 bytes. The default value (i.e. the value after a reset of the device) of the *Max_Block_Length* is 32 bytes.	bin8 (32-228)
COMMANDS		
11	***Add_Recipient_Addr*** Each recipient address is identified by 2 bytes: - Subnet = device type - Node = 1 to 127	bin8, bin8 (1-255, 1-127)
12	***Remove_Recipient_Addr*** Each recipient address is identified by 2 bytes: - Subnet = device type - Node = 1 to 127	bin8, bin8 (1-255, 1-127)

4.5.1 Initial Node Installation

When installing a device on the IFSF network for the first time, it is necessary to configure its node address. This should be done by activating an internal switch, which sets the devices' Node address to 127. The Site Contoller then addresses the device with Node = 127 and assigns a new Node number to Db_Ad 00H, Data_Id 02H. Note that it is only possible to configure one device an a time.

4.5.2 Recipient Address Table

Unsolicited messages are sent to all recipients which have requested to receive them.

A maximum number of 64 recipient addresses are necessary for this function (this is the maximum number of nodes allowed on the network without a repeater).

Please note that it is not expected that devices remove recipients from the address table should they be deemed as being off-line (i.e. no heart beat has been received). For performance reasons, a device can decide to not send an unsolicited message to devices listed in the recipient address table but deemed as off-line.

To manage the address table the following functions are usable:
- read the recipient address table,
- write the recipient address table,
- add one recipient address in the table,
- remove one recipient address from the table.

Implementation Guideline

To enable a site controller to configure the recipient address tables of each node on the network, it must know both the logical and physical addresses of each node. This can only be done when a heartbeat message has been transmitted from a node, since it is only then that the site controller receives the physical address.

For example, consider the case of a site controller (PNA 2/1 LNA 2/1) that wants to configure a dispenser (PNA 1/1 LNA 1/1) to send its unsolicited messages to a tank gauge (PNA 9/1 LNA 9/1). It cannot do this unless it knows the physical node address of the tank gauge, since the “Write Recipient Address Table” command only contains a logical address field. Once a heartbeat command has been received from the tank gauge then its physical address can be obtained and added to a “Physical Address Table” within the site controller.

Note also that each subnet/node address must be unique within an IFSF/LON network.

4.5.2.1 Read "Recipient Address Table"

Message provided by the application	Request message on the Network
LNAR	LNAR
	LNAO
IFSF_MC = 2	IFSF_MC = 2
	BL
M_St = Read + token	M_St = Read + token
M_Lg = 3	M_Lg = 3

DB_Ad_Lg = 1
DB_Ad = 0
	DB_Ad_Lg = 1
	DB_Ad = 0

Data_Id = 3	Data_Id = 3

Answer message on the Network	Answer received by the originator application
LNAO	
LNAR	LNAR
IFSF_MC = 0	IFSF_MC = 0
BL	
M_St = Answer + token	M_St = Answer + token
M_Lg	M_Lg

DB_Ad_Lg = 1
DB_Ad = 0
	DB_Ad_Lg = 1
	DB_Ad = 0

Data_Id = 3 Data_Lg = 2 * x Data_El = S & N S & N . . (x * S & N)	Data_Id = 3 Data_Lg = 2 * x Data_El = S & N S & N . . (x * S & N)

The LNA = 0 (Subnet = 0, Node = 0) is used for internal messages (application layer to communication layer). For this internal access, the network messages don't exist, the answer is sent back directly to the local application.

DB_Ad = 0 means "Communication Service Database".

Undefined recipient addresses in the *Recipient_Addr_Table* are defined by S & N = 0.

4.5.2.2 Write "Recipient Address Table"

Message on the Network
LNAR
LNAO
IFSF_MC = 2
BL
M_St = Write + token
M_Lg

DB_Ad_Lg = 1
DB_Ad = 0

Data_Id = 3 Data_Lg = 2 * x Data_El = S & N S & N . . (x * S & N)

The LNAR = 0 (Subnet = 0, Node = 0) is used for internal messages (application layer to communication layer). For this internal access, the network messages don't exist, the answer is sent back directly to the local application.

DB_Ad = 0 means communication level database.

Undefined recipient addresses in the *Recipient_Addr_Table* are defined by S & N = 0.

4.5.2.3 Add "Recipient Address"

Message on the Network
LNAR
LNAO
IFSF_MC = 2
BL
M_St = Write + token
M_Lg

DB_Ad_Lg = 1
DB_Ad = 0

Data_Id = 11
Data_Lg = 2
Data_El = S & N

The LNAR = 0 (Subnet = 0, Node = 0) is used for internal messages (application layer to communication layer). For this internal access, the network messages don't exist, the answer is sent back directly to the local application.

If all the recipient addresses are occupied, the "ADD" command is refused and a NAK message is returned (MS_ACK = 5, Data_ACK = 5).

4.5.2.4 Remove "Recipient Address"

Message on the Network
LNAR
LNAO
IFSF_MC = 2
BL
M_St = Write + token
M_Lg

DB_Ad_Lg = 1
DB_Ad = 0

Data_Id = 12
Data_Lg = 2
Data_El = S & N

The LNAR = 0 (Subnet = 0, Node = 0) is used for internal messages (application layer to communication layer). For this internal access, the network messages don't exist, the answer is sent back directly to the local application.

If the recipient address to remove is not in the table, a NAK message is returned (MS_ACK = 5, Data_ACK = 5).

4.5.3 Unsolicited Data

Messages on the Network
LNAR = First address (S,N) from the *Recipient_Addr_Table*
LNAO
IFSF_MC = 0
BL
M_St = Unsolicited + token
M_Lg
DB_Ad_Lg DB_Ad
Data (Data_Id, Data_Lg, Data_El) Data (Data_Id, Data_Lg, Data_El) . . . Data (Data_Id, Data_Lg, Data_El)
LNAR = Second address (S,N) from the *Recipient_Addr_Table*
LNAO
IFSF_MC = 0
BL
M_St = Unsolicited + token
M_Lg
DB_Ad_Lg DB_Ad

Data (Data_Id, Data_Lg, Data_El) Data (Data_Id, Data_Lg, Data_El) . . . Data (Data_Id, Data_Lg, Data_El)
LNAR = Last address (S,N) from the *Recipient_Addr_Table*
LNAO
IFSF_MC = 0
BL
M_St = Unsolicited + token
M_Lg
DB_Ad_Lg DB_Ad
Data (Data_Id, Data_Lg, Data_El) Data (Data_Id, Data_Lg, Data_El) . . . Data (Data_Id, Data_Lg, Data_El)

4.5.4 Heartbeat

Heartbeat message on the network (sent by any equipment)	
broadcast on Domain 1	
LNAO (Originator Subnet + Node)	
IFSF_MC = 1	
DEVICE_STATUS	bin8 (0-255) bit 1 = Configuration Needed bit 2-7 = Reserved for future use bit 8 = Software refresh required

Periodically each device sends on the network a broadcast message. The period for this "Heartbeat-message" is defined by Data_Id 4 *"Heartbeat_Interval"* in the Communication Service Database. Each device can use it to control that the communication with this equipment is still available.

The *"Recipient_Addr_Table"* is used by the communication layers to control that heartbeat messages are received from devices recorded in this address table.

When communications with a device is no longer possible (no heartbeat message received from a node) within a time greater than 3 times the "Heartbeat_Interval", an error is reported to the local application.

DEVICE_STATUS - Bit 1 is set, *i.e.* Configuration Needed, means that all data that is needed for correct operation is not available. Example: In the case of power loss, a dispenser should send a Configuration Needed heartbeat since it can not know if it has the correct prices.

NOTE - To allow devices on the same network to have different heartbeat intervals, each device should read the other heartbeat intervals and use these (x3) before deciding not to send unsolicited messages to that device.

4.6 Data Encryption

In order to maintain system integrity for security purposes, a method of data encryption must be provided. Although it is possible to encrypt data at a LonWorks level, this is not implemented in an IFSF network instead, IFSF uses a separate means of data encryption. The method described below is used for the Customer Operated Payment Terminal (COPT).

4.6.1 COPT Data Encryption

This method supports the desired ability to encrypt data collected at the COPT and transfer this data to the controller where it can be decrypted. It provides a secure method where the communications between the COPT and the controller can not be monitored to collect sensitive data. It also supports and eases equipment support in field. A COPT need not be unique for a customer and does not need to be injected at the factory or at the site.

This method provides security against attacks such as physical penetration; device substitution; tapping; bypassing encryption; and transaction replay or alteration. The architecture employs a local key management zone. The zone always uses DUKPT and unique key per terminal. This provides several advantages. First of all it allows the COPT to always use DUKPT and to employ a unique derivation key per COPT, regardless of the key management method employed in the rest of the system. Because the COPT gets its initial key from the controller after installation, the COPT does not have to be dedicated to specific customers, reducing spares cost and decreasing MTTR. The method used to provide the initial key to the COPT is Exponential Key Exchange (Diffie-Hellman), a public key technique.

During operation, the data is collected at the COPT and encrypted under the DES Encryption Algorithm creating an encrypted data block. Using DUKPT the encrypted data is sent to the controller which can then be decrypted. The keys used by the COPT may be changed at any time, if compromise is suspected (or just for additional security), without returning them to a key injection facility or taking a key loading device to the site.

At the heart of the security structure is the use of Exponential Key Exchange to initialize the COPT from the controller. This technique makes possible many of the benefits mentioned earlier while providing a high level of security for data within the system.

In the controller the data is decrypted. With the decrypted data in clear text in the controller, the controller is always a possible target of attack.

The method used to provide the initial key in the COPT is to calculate the key at both the COPT and the controller. This method involves the transmission of intermediate data between the controller and the COPT. In order to prevent discovery of the encryption keys by monitoring the transmission link the COPT implements the algorithm termed ‘Exponential Key Exchange”. This method is shown below.

Controller	**COPT**
Controller calculates	**COPT** calculates
R = mod q(a to the r power)	S =mod q(a to the s power)
Controller encrypts R under the default key	**COPT** encrypts S under the default key
Controller sends encrypted R to the COPT	**COPT** sends encrypted S to the Controller
Controller calculates	**COPT** calculates
KD = mod q(S to the r power)	KD = mod q(R to the s power)
KEK = f(KD)	KEK = f(KD)

Note: The “mod q” operation finds the integer remainder after long division, by q, of (a to the r).

“a” and “q” are large prime numbers (128 bits minimum) known to both the controller and the COPT. Their exact value is not and does not need to be a secret. “r” and “s” are large random numbers generated in the controller and the COPT and are kept secret. ‘R” and “S” are large numbers and are the intermediate values sent across the communication link, encrypted under a default DES key. Knowledge of “a”, “q”, “R”, “S” and even the default DES key is not sufficient to discover “r”, “s” or KD.

Note that the KD value is different for every COPT in the site because it is based on random numbers “r” and “s”, and is different every time the calculations are performed even if by the same COPT. The latter facilitates replacement of a suspected compromised key.

To break this method requires an iterative approach and boils down to “which value of “r” in the calculation “mod q(a to the r)” produces ‘R””. In addition to discovering “r”, a successful penetration would also have to discover “s”.

The KD is then manipulated by the Function f() to produce a DES key which we term the Key Exchange Key (KEK). The KEK is used to encrypt and decrypt derivation keys, which are shipped from the controller to the COPT to be used for DUKPT operation.

Below are the exchange sequences needed to initialize the COPT and to send encrypted data to the controller.

Key Exchange Sequence

	COPT	**CD**
1.	Loads version number into the version_number data field and generate a DMK, Default Master Key from it. This version_number is read-only.	
2.		Read version_number from COPT and generate the DMK, Default Master Key.
3.		Calculate 'R', encrypt 'R' with DMK, and write encrypted 'R' into the COPT data field encrypted_R.. This encrypted_R is write-only.
4.	Calculate 'S' and encrypt 'S' with DMK.	
5.	Decrypt encrypted_R with DMK and calculate KD.	
6.	Derive KEK, Key Exchange Key, from the KD.	
7.	Encrypt the DMK using the KEK.	
8.	Send unsolicited message with two data fields: e'S' = 'S' encrypted with DMK eDMK = DMK encrypted using KEK	
9.		Decrypt e'S' with DMK and calculate KD.
10.		Derive KEK, Key Exchange Key, from KD.
11.		Decrypt eDMK with KEK and compare it to the DMK that was generated locally. The decrypted DMK must equal the

		locally generated DMK. If not, restart at step 2.
12.		Generate a KSNR, Key Serial Number, and a CBDK, COPT Base Derivation Key from the LBDK, Local Base Derivation Key.
13.		Encrypt the CBDK using the KEK.
14.		Write the KSNR into the KSNR data field in the COPT. Write the encrypted CBDK in the encrypted_CBDK data field in the COPT. These two fields are write-only fields in the COPT.
15.	Decrypt the encrypted_CBDK using the KEK and generate future keys, DUKPT, for the local UKPT function.	

Encrypting Data for CD

	COPT	**CD**
1.	Encrypt data using DUKPT and store in database.	
2.	Store KSNR used to identify key used to encrypt data in database.	
3.	Send unsolicited message that encrypted data is ready.	
4.		Read encrypted data, KSNR and decrypt using DUKPT.

Reinitializing COPT DUKPT

At CD defined intervals:

	COPT	**CD**
1.		Generate a KSNR, Key Serial Number, and a CBDK, COPT Base Derivation Key from the LBDK, Local Base Derivation Key.
2.		Encrypt the CBDK using the KEK.
3.		Write the KSNR into the KSNR data field in the COPT. Write the encrypted CBDK in the encrypted_CBDK data field in the COPT. These two fields are write-only fields in t he COPT.
4.	Decrypt the encrypted_CBDK using the KEK and generate future keys, DUKPT, for the local UKPT function.	

5 Implementation Guidelines & Recommendations

5.1 Actions when a Device Recognises that a SC is Off-Line

A device recognises that a SC device, that has been entered into its recipient table, has gone off-line or that there is a line break when it does not receive a heartbeat within the duration of 3 times the heartbeat interval (normally 3 times 10 seconds = 30 seconds).

When this happens:

- Stop sending unsolicited messages to the off-line SC device

- Do not remove the off-line SC device from the Recipient Table.