INTERNATIONAL FORECOURT STANDARDS FORUM

STANDARD FORECOURT PROTOCOL

PART III.I

DISPENSER APPLICATION

VERSION 2.25 November 2007

This document is written by the IFSF - Working Group:

Name	Company	Telephone
Erwin Busch	Scheidt & Bachmann GmbH, Germany	+49/2166/266330
Arnaud de Ferry Frank Simons Tjeerd Boettcher	Tokheim, NL	
Peter Maeers	VBi Limited, UK	
Eduardo Rezende John Carrier Jürgen Wedemann	Shell Europe Oil Products, Europe	
Jaroslav Dvorak	Beta Control Limited, Czech	

For further copies and amendments to this document please contact:

IFSF Technical Services via the IFSF Web Site (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,...) - Calculator message definition transformed in database definition have generated a lot of update which cannot be listed here (duplicate definitions suppressed, field renaming for consistency across the document, ...). - The field numbering was reviewed to provide an unique number per database. - Mandatory/Optional (M/O) column was suppressed (application dependent). - Read/Write (R/W) column was suppressed (application dependent). - Error codes were reviewed for simplicity.

Date	Version number	Modifications
June 93	1.3	**Chapter 1 - Definitions and Abbreviations** - Table layout changed - Controller Device description added - Outdoor Payment Terminal description added
		Chapter 2 - Fuelling Point Behavioural Model - Table layout for state and event description. - "Assign" function explained with the state machine. - **BUFFER FULL** state is suppressed. - the "Release_FP" command is only acceptable if at least one transaction buffer is empty. - "Terminate_FP" moves to the **IDLE** state from any state, storing if necessary a transaction. - "Close_FP" goes to the **CLOSED** state from any state, storing if necessary a transaction. - In the **IDLE** state, all nozzles must first be hooked before starting a new transaction by "Nozzle-up" or "Release_Fp". - Rename events "Time-out-end" to "Fill-Time-out" and "Auth-time-out". - Rename event "Time-out" to "No-progress". - Transition from state **SUSPENDED STARTED** to **SUSPENDED FUELLING**. - "Time-out" or "Terminate_Fp" from STARTED or **SUSPENDED STARTED** states goes to **IDLE**. - In state **IDLE** "Nozzle-Down" is requested before an acceptable "Nozzle-Up". - Add figure 3 (Fuelling Point State Table). - The event "Operative_FP" means that internal test is successful. - In state **CLOSED** a payable transaction may be exist. - "Major-error" and "Minor-Error" is added to state **CLOSED**.

		- "Inoperative_FP" could also be an internal command. - Event "Local-release" does not longer exist. - State **PAYABLE FUELLING** is renamed to **PAYABLE TRANSACTION**. - State **FUELLING RESERVED** is renamed to **LOCKED TRANSACTION**.
		Chapter 3 - Dispenser Database - Calculator Databases addresses are completed and moved from the "Communication" chapter to the beginning of the Calculator database description. Database address is indicated in each database description. - Common field formats: - DATE years using 4 digits. - Database Calculator: - new column indicates if the data can be Read and/or Write in which fuelling point state, - various minor update and completions, - add "Ticket_Header" and Ticket_Footer", - Drive_Off_Light_Mode, - OPT_Light_Mode, - add Auth_State_Mode, - add "Tempo_Synthesis", - add "Time_Out_Communication", - add " LCD_Backlight", - add "Display_Intensity", - simplify "Amount_Rounding_Type". - Database "Fuelling Point": - comments on "Assign_Contr_Id" and unsolicited message generation when changed, - add "Phy_Noz_State", - extend "Log_Noz_Mask" over 2 bytes, - add "Product_Mask" over 2 bytes, - add "ZeroTR_Mode", - "Current_Contr_Id" renamed "Release_Contr_Id", - add " Suspend_Contr_Id" - add "OPT_Light_Switch", - add "Drive_Off_Light_Switch". - Database "Logical Nozzle": - add "Log_Noz_Light", - add "Synthe_Mode", - add "Noz_Display_Mode". - Database "Transaction Buffer: - add "Display_Transaction" command. - Database "Voice Synthesis" is suppress, the software download feature is generalised to software and data manufacturer specific. - "Fuelling Point Error Database": - add "FP_Error_Description", - add "FP_Error_Type" takes values of "Error Code" - "Error Code" used as a key to access each Error record.

Date	Version number	Modifications
August 93	1.31	**General** - Add Document Part Number on the front page PART III.1 - Add the names and addresses from document authors - Additional description - Correction of spelling errors
		Chapter 1 - Definitions and Abbreviations - Add description where the numbering of LN, PN and M starts - Add definitions for: - Tank Level Gauge - Fuelling Mode - Stand Alone - Offline Mode - Online Mode - Transaction Buffer - Payable Transaction - Zero Transaction
		Chapter 2 - Fuelling Point Behaviour Model - Divide action description in "input action" and "output action" in all tables - Additional information when a unsolicited message is sent - Description for Figure 3 FUELLING POINT STATE TABLE is added - Event "Operative_FP" is renamed to "Operative" - Event "Inoperative" deleted - Event "Unable" included - Headline for Figure 2 changed to FUELLING POINT STATE DIAGRAM, ERROR CONDITIONS - Major error moves always to state **INOPERATIVE** - Additional description on Figure 2 has changed - Add information to state description **INOPERATIVE** - Add information to state description **CLOSED** - Delete event "Inoperative" - Include event "Unable" - Add information to state description **IDLE** - A "Release_FP" is only excepted if the FP is not assigned to another CD - Changes in the description **CALLING** - Max_Auth_Time is started in state **AUTHORISED** - Description of "zero transaction" has changed - The maximum authorization timer is started in state **AUTHORISED** - The maximum filling timer is started in state **STARTED** - Max_Noz_laydown_Time is not used any more - Add information to state description **SUSPENDED FUELLING**
		Chapter 3 - Dispenser Database - Changes for Data Address table (chapter 3.1): - Add address names FP_ID, PR_ID, PR_DAT, M_ID, SW_DAT, LN_ID, TR_DAT, AUD_ID, ER_DAT, ER_ID, FM_ID. - Reduce number of LN_ID to 8 - Reduce number of PR_ID to 8 - Delete 2nd address level (FM) for PR_ID - TR_Seq_Nb starts with 0000 - Prod_Nb starts with 00000001 - Changes for Common Field Formats (chapter 3.2): - Add description for binX, bcdX, ascX, hexX, CMD - Long_Total renamed in Long_Number

- Changes for Calculator Database (chapter 3.3):
  - Address name abbreviation included
  - Data_Id 1: Value 0-255 allowed
  - Data_Id 2: Rename to Nb_Products
  - Data_Id 3: Rename to Nb_Fuelling_Modes
  - Data_Id 4: Rename to Nb_Meters
  - Data_Id 5: Rename to Nb_FP
  - Data_Id 6: Add countries, field type bcd4
  - Data_Id 8: Add description, 0 means that light is not in use
  - Data_Id 9: Add description, 0 means that light is not in use
  - Data_Id 10: Add description, meaning off bit 1-7 has changed
  - Data_Id 12: Add variable Stand_Alone_Auth
  - Data_Id 20: Deleted
  - Data_Id 21-30: Specify what happens to value 0
  - Data_Id 40-42: Field type is bcd2
  - Data_Id 46: Deleted
  - Data_Id 47: Deleted
  - Data_Id 50-58: Write not allowed
  - Data_Id 60: Rename to SW_Change_Personal_Nb
  - Data_Id 70-72: Write allowed in state 1-9
- Changes for Meter Database (chapter 3.4):
  - Address name abbreviation included
  - Date_Id 1-2: 0 means not configured
  - Date_Id 4: Add description
  - Date_Id 5: Deleted
  - Date_Id 20: Totals are not resetable, totals are updated permanently
- Changes for Product Database (chapter 3.5):
  - Address name abbreviation included
  - Database access only with PR_ID
  - Description changed
  - Date_Id 1: Deleted
  - Date_Id 2: Description changed
- Changes for Product per Fuelling Mode Database (chapter 3.6):
  - Address name abbreviation included
  - Database access only with Prod_Nb
  - Date_Id 2-5: Write is allowed in FP state 1-4
- Changes for Product per Fuelling Point Database (chapter 3.7):
  - Address name abbreviation included
  - Data_Id 2-5: Abbreviation No_ is changed to Nb_
  - Data_Id 4: Nb_Logical_Nozzle is reduced to 1 to 8
  - Data_Id 7: Add variable Default_Fuelling_Mode
  - Data_Id 10: Description changed
  - Data_Id 11: Description changed
  - Data_Id 21: Change name to Log_Noz_State, Field type changed to bin8, description changed
  - Data_Id 20-22: Add information that unsolicited message is sent, if state has changed
  - Data_Id 23: Change name to Release_Mode
  - Data_Id 24: Change name to ZeroTR_Mode, Description changed
  - Data_Id 25: Description changed, number of logical nozzles is maximum 8, the field type is bin8
  - Data_Id 26: Deleted
  - Data_Id 27,28: Write allowed in state 3-4
  - Data_Id 29: Moved to the CURRENT TRANSACTION DATA, description changed
  - Data_Id 30: Moved to CURRENT TRANSACTION DATA,

		Description changed, Write not allowed - Data_Id 31: Moved to CURRENT TRANSACTION DATA, Description changed, Write not allowed - Data_Id 32: Change name to Release_Token - Data_Id 33: Change name to Fuelling_Mode - Data_Id 29-31, 34-40: Current transaction data are reset to 0 after storing the transaction data, current data can only be read in FP state 6-9 - Data_Id group COMMAND is renamed to FP CONTROL - Data_Id 62, 64-65: Description changed, Field Type is bin8, the CD Identifier is sent with the command - Data_Id 100: The Log_Noz_State is sent instead of the Current_Log_Noz - Changes for Logical Nozzle Database (chapter 3.8): - Address name abbreviation included - Address for logical nozzle has changed to 11H - 18H - Data_Id 1: Description changed - Data_Id 2: Deleted - Data_Id 4: Measured in centilitres - Data_Id 5: Description changed - Data_Id 6: Deleted - Data_Id 7, 9: Description changed, value is 0 to 16 - Data_Id 8: Field Type changed to bcd2, 0 means no blending - Data_Id 10: Description changed - Data_Id 12: Description added, Write only in state 1-2 - Data_Id 20: Rename to Log_Noz_Vol_Total, Write not allowed - Data_Id 21: Rename to Log_Noz_Amount_Total, Write not allowed - Data_Id 22: Field type is Long_Number, write not allowed - Data_Id 30: Rename to Log_Noz_SA_Vol_Total - Data_Id 31: Rename to Log_Noz_SA_Amount_Total - Data_Id 32: Field type is Long_Number - Changes for Fuelling Transaction Database (chapter 3.9): - Chapter name changed - Address name abbreviation included - Data_Id 1: Description changed - Data_Id 2-13: Add information which data must be stored, Read is allowed in state 1-9 - Data_Id 2: Description changed, value 1-254 allowed - Data_Id 9: Deleted - Data_Id 12: Description changed - Data_Id 14: Description changed - Data_Id 20-32: Write allowed in state 1-9 - Data_Id 30: Description changed, field type is bin8 - Data_Id 31: Deleted - Changes for Transaction Audit Database (chapter 3.10): - Chapter name changed - Address name abbreviation included - Data_Id 1: Renamed to ATR_Seq_Nb - Data_Id 8: The field type for the ATR_Prod_Nb is asc8 - Changes for Error Code Database (chapter 3.11): - Chapter nname changed - Address name abbreviation included - Data_Id 1: Description changed, value 1-255 allowed - Data_Id 1-4: Read allowed in state 1-9 - Error table totally changed - Changes for Data Download Database (chapter 3.12): - Data_Id 3: Rename to Data_Download

		- Data_Id 4: Variable Start_Addr added - Data_Id 5: Variable Nb_Bytes added - Data_Id 6: Variable Data_Checksum added - Data_Id 10: Description changed - Data_Id 11: Variable Restart added

Date	Version number	Modifications
Nov 93	1.40	**General** - English language improvements - "Data Variable" is renamed to "Data Element" - "Data Field" is renamed to "Data Element"
		Chapter 1 - Definitions and Abbreviations The description of the numbering for LN, PN and M is deleted. The numbering is manufacturer model specific. - An explanation for the LNA added
		Chapter 2 - Fuelling Point Behaviour Model - Chapter 2.1: - In figure 3 the event "Nozzle down" in state STARTED moves to state 3 or 5 - Chapter 2.2: - In figure 4 the state BUFFER EMPTY is renamed to CLEARED TRANSACTION - Additional information for the transaction buffer handling for non paid transactions and historic transactions - Chapter 2.2.1: - State name changed to CLEARED TRANSACTION - State description changed - Event description for "new payable transaction" changed - Chapter 2.2.2: - Event description for "Clear" is changed - Chapter 2.2.3: - Event description for "Clear" is changed
		Chapter 3 - Dispenser Database - All Chapters: - Column added to all data element tables indicating if the are a mandatory data elements or optional - The LNA and DA is removed from all database addresses - Chapter 3: - Explanation of the table columns added - Chapter 3.1: - Chapter renamed to Database Address - Additional information - LNA, DA columns from the Database Address table removed - AUD_ID (31H-3FH) not longer used - TR_Seq_Nb could be 0001-9999

- Chapter 3.3:
  - Additional information to use the CALCULATOR database
  - Data_Id 1: deleted, specified by the subnet (S) in the Logical Node Address (LNA)
  - Data_Id 10: Additional explanation to the clearing of a FP's display
  - Data_Id 25: deleted, no need
  - Data_Id 27: deleted, no need
  - Data_Id 29: deleted, no need
  - Data_Id 30: deleted, the heartbeat on communication level is used
  - Data_Id 61: deleted, no real time clock in the dispenser
  - Data_Id 62: deleted, no real time clock in the dispenser
  - Data_Id 200-255: free to manufacturer / oil company
- Chapter 3.4:
  - Additional information to use the METER database
  - The M_ID = 80H is used to have access to all meters
  - Data_Id 200-255: free to manufacturer / oil company
- Chapter 3.5:
  - Additional information to use the PRODUCT database
  - The PR_ID = 40H is used to have access to all products
  - Data_Id 200-255: free to manufacturer / oil company
- Chapter 3.6:
  - Additional information to use the PRODUCT PER FUELLING MODE database
  - The FM_ID = 10H is used to have access to all fuelling modes at a product
  - Data_Id 200-255: free to manufacturer / oil company
- Chapter 3.7:
  - Additional information to use the FUELLING POINT database
  - The FP_ID = 20H is used to have access to all fuelling points
  - Data_Id 2: Renamed to Nb_Tran_Buffer_Not_Paid, description changed
  - Data_Id 3: Renamed to Nb_Of_Historic_Trans, description changed
  - Data_Id 5: deleted, no need
  - Data_Id 21: Numbering description changed
  - Data_Id 22: Description changed, field type is bin16
  - Data_Id 25: Numbering description changed
  - Data_Id 41: Data element Transaction_Sequence_Nb created
  - Data_Id 29: Description changed
  - Data_Id 30: Description changed, field type is bin16
  - Data_Id 31: Description changed, field type is bin16
  - Data_Id 62: Description changed, field type is a CMD
  - Data_Id 64: Description changed, field type is a CMD
  - Data_Id 65: Description changed, field type is a CMD
  - Data_Id 100: The unsolicited message is without acknowledge
  - Data_Id 200-255: free to manufacturer / oil company

		- Chapter 3.8: - Additional information to use the LOGICAL NOZZLE database - The LN_ID = 10H is used to have access to all logical nozzle data at a fuelling point - The range for the LN_ID is 11H-18H - Data_Id 5: Numbering description changed - Data_Id 7: Numbering description changed - Data_Id 8: Field type changed to bin8, the value is 0-100 - Data_Id 9: Numbering description changed - Data_Id 12: deleted, no need - Data_Id 13: deleted, no need - Data_Id 14: deleted, no need - Data_Id 200-255: free to manufacturer / oil company - Chapter 3.9: - Additional information to use the FUELLING TRANSACTION database - The TR_DAT = 20H is used to have access to all non paid transaction data - Data_Id 1: Description changed - Data_Id 2: Description changed, field type is bin16 - Data_Id 21: Data element Trans_State created - Data_Id 20: Description changed - Data_Id 30: Description changed, field type is CMD - Data_Id 32: deleted, no need - Data_Id 100: The unsolicited message is without acknowledge, the Tr_Buff_Status_Message array consist of TR_Seq_Nb, Trans_State, TR_Buff_Contr_Id - Data_Id 200-255: free to manufacturer / oil company - Chapter 3.10: - the previous chapter 3.10 "Transaction Audit Data" is completely deleted, historic transaction data are accessible by the Fuelling Transaction database - The previous chapter 3.11 "Error Code Data" is now 3.10 - Additional information to use the ERROR CODE database - The ER_DAT = 10H is used to have access to all error code data - Data_Id 3: decription changed - Data_Id 4: deleted, no real time clock in the dispenser - Data_Id 5: date element FP_Error_State created - Major error 0AH "Download error" added - Major error 0BH "Checksum error" added - Chapter 3.11: - The previous chapter 3.12 "Data Download" is now 3.11 - Data_Id 1: deleted

Date	Version number	Modifications
March 95	1.50	**General Changes** - Document converted to Word 6 format. - Changes made based on comments from CECOD and individual suppliers. **Chapter 2** - In chapter 2.1.1 details of behaviour when a major & minor error occurs have been added to the state description. - In chapter 2.1.3 minor change to wording. - In chapter 2.2.3 extra comments regarding control device access for 'unlocking'.

		Chapter 3 - In chapter 3.1 additional comments have been made about the TR_Seq_Nb address format (i.e. bcd4) - In chapter 3.1 additional comments have been made about the Prod_Nb address format (i.e. bcd8). - In chapter 3.3 Data_Id 2 (Nb_Products) field range changed from 1-15 to 1-8 and Data_Id made mandotary. - In chapter 3.3 Data_Id 3,4 & 5 made mandatory Data_Id. - In chapter 3.3 new Data_Id 61 (SW_Checksum) used to allow the CD to interrogate the software checksum. - In chapter 3.3 additional comments have been made about the handling of Data_Id 70 (Calc_Illumination) when the calculator can not support illumination. - In chapter 3.6 the Data_Id's 2,3,4,5 can now be written in all states. - In chapter 3.7 the Data_Id 23 (Release_Mode) has some additional explanation. - In chapter 3.7 the Data_Id 30 (Release_Contrl_Id) has some additional explanation regarding the resetting of the value when the IDLE state is entered. - In chapter 3.7 the Data_Id 38 has had its field format changed from asc8 to bcd8. - In chapter 3.7 the Data_Id 100 has additional details of how the unsolicited data structure is set up (i.e. the individual data elements have their Data_Id and Data_Lg included in the message & the Data_Lg of Data_Id 100 = 0). - In chapter 3.8 the Data base address has been corrected. - In chapter 3.8 the Data_Id 8 & 9 have been changed to optional and have some additional explanation regarding their implementation. - In chapter 3.9 the Data_Id 2 (TR_Contr_Id) has had its field format changed from bin8 to bin16 and an explanation on the field details has been added. - In chapter 3.9 the Data_Id 10 (TR_Prod_Nb) has had its field format changed from asc8 to bcd8. - In chapter 3.9 the Data_Id 11 (TR_Prod_Description) has been changed from mandatory to optional. - In chapter 3.9 Data_Id 31 (Lock_Transaction) command has been added. - In chapter 3.9 Data_Id 32 (Unlock_Transaction) command has been added. - In chapter 3.9 the Data_Id 100 has additional details of how the unsolicited data structure is set up (i.e. the individual data elements have their Data_Id and Data_Lg included in the message & the data_Lg of Data_Id 100 = 0). - In chapter 3.10 the state error 80H has been changed from 'FP state OPERATIVE' to 'FP state INOPERATIVE'. - In chapter 3.11 Data_Id's 2,5,3,6,10 & 11 have all been changed from mandatory to optional.

Date	Version number	Modifications
January 96	1.51	

General Changes

- Changes made based on comments from CECOD and individual suppliers.
- Many Data_Id's descriptions have been changed to explain their usage in situations where the Dispenser for technical reasons (W&M requirements or a fixed relationship between the meters & nozzles, etc.) can not allow a mandatory Data_Id value to be written to or changed. Please check all Data_Id's for details.

Chapter 1 Definitions and Abbreviations

- Extra text definition defining Off-line.
- Extra text definition defining On-line.

Chapter 3 Dispenser Data Base

- Calculator Data Base
- Extra text explaining the operation for Data_Id's 2,3,4,5,6,7,21,22,23,24,26,40,41,42,43,44 & 45 when that do not permit the *Data_Id* to be changed remotely.
- Added optional Data_Id 46 Price_Set_Nb.
- 3.4 Meter Data Base
- Extra text explaining operation when Data_Id 1(Meter_Type), Data_Id 2 (Meter_Puls_Vol_Fact) & Data_Id 4(PR_Id) are not configured.
- 3.6 Product Data Per Fuelling Mode
- Data_Id's 2(Prod_Price), 3(Max_Vol), 4(Max_Fill_Time) & 5 (Max_Auth_Time) can now be written to in state 5 (Authorised).
- Data_Id 4(Max_Fill_Time) has extra text regarding defaults after a master reset/cold start and handling when Data_Id can not be written to.
- 3.7 Fuelling Point Data Base
- Data_Id's 2(Nb_Tran_Buffer_Not_Paid), 3(Nb_of_Historic_Trans), 4(Nb_Logical_Nozzle) have additional text explaining what to do when a write action occurs with a value that can not be supported.
- Optional Data_Id 42(Current_Price_Set_Nb) added.
- Extra text explaining what action to take when a state error occurs with one of the commands (Open, Close, Release, Resume, Suspend & terminate).
- 3.8 Logical Nozzle Data Base
- Data_Id's 1(PR_Id) & 7(Meter_1_Id) have additional text explaining what to do when a write action occurs with a value that can not be supported.
- 3.9 Fuelling Transaction Data Base
- Extra text explaining the multiple read of current transactions.
- Optional Data_Id 9(TR_Price_Set_Nb) added.
- Data_Id 20(TR_Buff_Contr_Id) has been made read only.
- Data_Id 31(Lock_Transaction) has additional text.
- Data_Id 32(Unlock_Transaction) has additional text explaining how to unlock a transaction when the transaction was locked by another CD..
- Extra text regarding the generation of unsolicited messages for Data_Id's 30,31 &32..
- 3.10 Error Code Data Base
- Database limited to 64 error codes (was 255).
- Data_Id 1 (FP_Error_Type) can no longer be written too.
- Data_Id 1 (FP_Error_Type) has a range of 1 to 255.
- Error Classification table has been changed to reflect the new range of error codes.

		Chapter 4 - Chapter 4 added. This chapter gives dispenser example configurations for information purposes. **Chapter 5** - Chapter 5 added. This chapter gives implementation details (reset handling and general state handling).

Date	Version number	Modifications
June 97	2.00	**General Changes** - Changes made based on comments from CECOD and individual suppliers. - Many Data_Id's descriptions have been changed to explain their usage in situations where the Dispenser for technical reasons (W&M requirements or a fixed relationship between the meters & nozzles, etc.) can not allow a mandatory Data_Id value to be written to or changed. Please check all Data_Id's for details. **Chapter 1 Definitions and Abbreviations** - **Chapter 2 Fuelling Point Behavioural Model** **Chapter 2.1.3 Idle State** - Extra text detailing the handling when the FP is in the Idle state with an 'illegal' nozzle removed. **Chapter 2.1.4 Calling State** - Extra text detailing the that the release command must be rejected if an 'illegal' nozzle removed. **Chapter 2.2.3 State Locked transaction** - Extra text detailing the that the a transaction resulting from an assigned FP will immediately go to the 'Locked' State.

		Chapter 3 Dispenser Data Base **Chapter 3.2 Common Field Formats** - Extra text explaining that the example at the beginning of the section represents an IFSF floating point. - Extra text explaining that the unit of measurement for the VOLUME field is implied. **Chapter 3.3 Calculator Data Base** - Data_Id 6 (*Country_Code*) has a new reference to the acceptance of ISO 3166 country codes. - Data_Id 10 (*Clear_Display_Mode*) has a new reference to the fuelling point's Data_Id 66 (*Clear_Display)* command. - Data_Id 13 (*Max_Auth_Time*) added (moved from the Product Data per Fuelling Mode data base, Data_Id 5). - Data_Id 44 (*Amount_Rounding_Type*) has extra examples. - Data_Id 80 (*W&M_Polynomial*) added.. - Data_Id 81 (*W&M_Seed*) added. **Chapter 3.4 Meter Data Base** - Data_Id 20 (*Meter_Total*) has additional text explaining that some CD's may write to this Data_Id. **Chapter 3.6 Product Data per Fuelling Mode Data Base** - Data_Id 5 (*Max_Auth_Time*) moved to Calculator data base Data_Id 13. **Chapter 3.7 Fuelling Point Data Base** - Data_Id 2 (*Nb_Tran_Buffer_Not_Paid*) has extra text explaining the correct handling when a write to this Data_Id is not allowed. - Data_Id 3 (*Nb_Of_Historic_Trans*) has extra text explaining the correct handling when a write to this Data_Id is not allowed. Additionally, it is now mandatory to support at least 1 historic transaction buffer. - Data_Id 8 Leak_Log_Noz_Mask has been added. - Data_Id 22 (Assign_Contr_Id) has extra text explaining that a transaction resulting from an assigned FP must go straight to the 'Locked' state. Additionally, there is a new DB protection linked to the FP being assigned. - Data_Id 66 (Clear_Display) has been added. - Data_Id 67 (Leak_Command) has been added. **Chapter 3.10 Error Code Data Base** - New 'Leak Error' Major Error code hex 0C/12 decimal added. **Chapter 5** - Chapter 5.4 added. This chapter gives implementation details in regard to the correct handling when dispensers or site controllers recognise that a device has gone off-line or come (back) on-line. - Chapter 5.5 added. This chapter gives implementation details in regard to the correct handling when switching a dispensers into stand alone mode. - Chapter 5.6 added. This chapter gives implementation details in regard to the handling of units of measure.

Date	Version number	Modifications
February 1999	2.01	**Chapter 2** 2.1.6 Additional text added to explain actions resulting from a LIMIT REACHED event. [IR019] 2.1.7 The state description for "Fill-Time-Out" is in conflict with the state diagram. It now reads: ".....moves the FP to the state IDLE." [IR020] 2.1.8 Additional text added to explain actions resulting from a LIMIT REACHED event. [IR019]
		Chapter 3 Chapter 3.7. Additional explanatory text to say that it is possible to clear a transaction from any CD. Clearing is not specific to the CD that released the FP. [IR043] Data_ID 26 added (Config_Lock) to lock the communications of a dispenser to one CD while the dispenser is being configured [IR040]. Additional text added to Data_Id 100 (FP_Status_Message) to clarify when the unsolicited message should be sent. [IR017] - Chapter 3.9. Data_Id 17 added to allow the dispenser to return the amount of tax for each transactions. An optional item for the Japanese market. [IR056] Chapter 3.9. Additional text added to Data_Id 100 (TR_Buff_Status_Message) to clarify when the unsolicited message should be sent. [IR017] Chapter 3.10. Error code data. Two additional error codes defined to indicate that a button has been pressed to suspend and resume the fuelling process (Japanese Requirement). Minor error, 28H - Fuelling suspended Minor error, 29H - Fuelling resumed [IR055] **Chapter 5** Chapter 5.4 added. A new implementation guideline - Actions when a dispenser recognises that the line is cut. [IR012] Chapter 5.5 - Additional text added to define Stand Alone mode [IR014]. Chapter 5.7 - Additional implementation guideline added to handle transitions between Fuelling and Idle [IR1016].

Date	Version number	Modifications
January 2000	2.10	
		Chapter 2 - Fuelling Point Behaviour Model Figure 1 - Fuelling State Diagram amended (IR1083). Figure 3 - Fuelling Point State Table amended (IR1083). 2.1.3 Additional text added to clarify if no unit price is available (IR1085). 2.1.4 Additional text added to clarify if no unit price is available (IR1085). 2.1.6 Additional text added to describe the minor error (Suspended_Fuelling) (IR1086) and (No_Progress) (IR1084). Text relating to (Limit-Reached) & (Fill-Time-Out) deleted. 2.1.7 Additional text added to describe the minor error (Fuelling_Resumed) (IR1086). 2.1.8 Additional text added to describe the minor error (Suspended_Fuelling) (IR1086). Additional text added to describe (Max_Vol) (IR1071 & IR1067) (Fill_Time_Out) (Limit_Reached) & (No_Progress) (IR1084). 2.1.9 Additional text added to describe the minor error (Fuelling_Resumed) (IR1086).
		Chapter 3 - Dispenser Database Data Id 9 Additional text added to describe (OPT_Light_Mode) and amended to indicate that values are now (0-255) (IR1069). Data Id 43 Additional text added for clarity purposes (IR1061). Data Id 44 Additional text added for clarity purposes (IR 1026). Data Id 76 Deleted some text and added new text to clarify that now Read_Only to avoid data integrity being compromised (IR1058). 3.5 Additional text added (IR1074). Data Id 2 Additional text added (IR1038). 3.6 Data Id 3 Deleted some text and added new text for clarity purposes (IR1071). Data Id 5 Previously removed in draft version 2. Reinstated in this version to ensure backwards compatibility (IR1063). Data Id 6 New data Id added (IR1071). 3.7 Data Id 8 Write value amended (IR1025). Data Id 11 Additional text added to describe (OPT_Light_Switch) and amended to indicate the values are now (0-255) (IR1069). Data Id 62 Additional text added (IR1085). Data Id 67 Write value amended (IR1025). 3.8 Data Id 20 Text and write value amended (IR1021). Data Id 21 Text and write value amended (IR1021). Data Id 22 Text and write value amended (IR1021). 3.10 Text amended to ER_DAT = 40H is used to ask for all error code data (IR1062). 22H Customer_Stop_Pressed added (IR1095). 3.11 Additional text added and Data Download Database moved to Communications Specification (IR1041).

Date	Version number	Modifications

March 2000	2.11	
		Chapter 1 - Definitions and Abbreviations Abbreviations PCD, PPP & PNA added. **Chapter 2 - Fuelling Point Behaviour Model** Protocol Converter Devices (PCD) Comments added to all sections. **Chapter 3 - Dispenser Database** Protocol Converter Devices (PCD) Comments added to all sections. **Chapter 6 - Protocol Converter Device Implementation Guidelines** New chapter added to help Protocol Converter Devices (PCD) implementations.

Date	Version number	Modifications
September 2002	2.12	**Chapter 3 – Dispenser Database** Data Id 57 Deleted some text & added new text (IR1096)(1123) Data Id 22 Text added – Communication databases (IR1114) Data Id 32 Deleted some text & added new text – Locking and unlocking transactions (IR1024) Data Id 42 Inserted Data Id 43, 44, 45 & 101 (IR1070) Data Id 5 Deleted some text & added new text – Product per fuelling mode database (IR1122) Data Id 6 Changed amount to volume (IR1097) Data Id 71 Changed 2 to 1 (IR1109) Data Id 72 Changed 2 to 1 (IR1109) Data Id 1 Changed 3 to 2 & (1-2) to (0-2) (IR1109) Data Id 10 Change 2 to 1 (IR1109) Data Id 23 Change 2 to 1 (IR1109) Fuelling point database Data Id 26. Text added. (IR1115) Product per fuelling mode database Data Id 6. Data element name changed. (IR1121) Error codes for Vapour Recovery added to the error code table.

June 2004	2.20	**General.** Version ID changed to 2.20 because functional changes made, all new attributes are optional to support backwards compatibility. White space removed and, header and footer size reduced to shorten document from 175 to 142 pages. Hexadecimal value of Data_Ids added **Chapter 2.2 Transaction Buffer State Diagram** Figure 4 - State 1 name changed to Cleared Transaction to be consistent with textual description. Text in 2.2.3 State Locked Transaction concerning unlocking clarified. **Chapter 3.2 Field Formats** Removed and reference to Engineering Bulletin 11 added. **Chapter 3.3 Calculator Data** Corrected Data_ACK for Data Id 80 and 81. **Chapter 3.6 – Product Per Fuelling Mode Database** Data ID 3 Write in State corrected to W(1-9). See Record of Changes version 1.50. Even though this is set by the supplier to ensure backward compatibility must be left at W(1-9). Data ID 6 Write in State corrected to W(1-9). This must be the same as data Id 3. PCD comment corrected. **Chapter 3.7 – Fuelling Point Data** Data ID 22 Unlocking of Locked FP's under error conditions clarified Data ID 59 added (optional). Typo correction to Data Id 101. Data ID 102 added (optional). Corrected Data_ACK for Data Id 45. **Chapter 3.9 – Fuelling Transaction Data** Data ID 20, 30-32 Unlocking of Locked Transaction Buffers under error conditions clarified. **Chapter 3.11 – Data Download** Section contents removed. **Chapter 5.2 – Implementation Guidelines & Recommendations** Action after Reset or Power Off text description clarified.
June 2005	2.21	**Chapter 2.1** Note added covering removal of more than one nozzle. **Chapter 2.1.4** Under "RELEASE_FP" note added on Data Ack to return, if no transaction buffer. **Chapter 3.3 – Calculator Database** Data ID 5 note added about single sided dispensers. Data ID 5 default value comment added. Data ID 11 note added, if Release_FP received with Authorised state not allowed. Data ID 44 examples improved and default value added. **Chapter 3.7 – Fuelling Point Data** Config_Lock made Read/ Write in state 1. **Chapter 3.8 – Logical Nozzle Data** Comments for Meter_2_Id made similar to Meter_1_Id. **Chapter 3.9 – Fuelling Transaction Data** Changed the state column, so the state now refers to the Transaction Buffer State Diagram and not the Point State Diagram. Command Clear_Transaction Data Id 30. Comment added about clear all transactions. Data ID 31 Unlock reference to Communications Specification added. **Chapter 3.10 – Error Code Data** Clarification of number of error codes to be returned.

September 2005	2.22	**Chapter 3.3 – Calculator Database** Data ID 61 option of being Read only. **Chapter 3.7 – Fuelling Point Data** Data ID 30 option of being Read only. **Chapter 3.10 – Error Code Data** Minor error 34 added.
March 2006	2.23	**Chapter 2.1.8 – State Fuelling [8]** Section on "MAX_VOL" minor error changed from Max_Vol to Limit_Reached. No Max_Vol error in error list. **Chapter 3.5 Product Data** MS_ACK changed from NAK 2 to NAK 6 (incorrect value). **Chapter 3.7 Fuelling Point Data** Terminate_FP changed from W(3-9) to W(4-9), typo/ error. OPT_Light_Switch TS Blink changed to Fast Blink. Clarification on time of blinking. Alarm structure added. **Chapter 3.9 Fuelling Transaction Data** Read/ Write in State corrected for Clear_Transaction, Lock_Transaction and Unlock_Transaction. Typo. **Chapter 3.10 Error Code Data** Further clarification on which errors to send back and support.
March 2007	2.24	**Enhanced Vapour Recovery bit elements added to FP_Alarm as agreed at IFSF WG meeting on 31st October 2006.**
November 2007	2.25	**Clarification of Multi-database read of Fuelling transaction database. See chapter 3.9**

1 Definitions and Abbreviations

Definition	Abbreviation	Description
Controller Device	CD	The CD is any device that is capable of controlling other forecourt devices (i.e. *Dispensers*, *Tank Level Gauges*, *Outdoor Payment Terminals*, etc.)
Dispenser	-	The complete dispensing unit consisting of one or more (maximum 4) *Fuelling Points*.
Dispenser Calculator	DC	The DC is the dispenser's electronic head for process control, communication and calculation.
Fuelling Point	FP	The item of forecourt equipment which is capable of dispensing a single motor fuel product at one time. The Fuelling Point contains one or more *Logical Nozzles*. The customer identifies this Fuelling Point normally with "Pump Number".
Logical Nozzle	LN	The logical nozzle specifies the motor fuel dispensed from a *physical nozzle*. In the case of blending two or three logical nozzles are assigned to one physical nozzle. If the product being dispensed is not a blended product the relationship between the physical nozzle and the logical nozzle is one/one.
Physical Nozzle	PN	The physical nozzle is the actual nozzle that a customer removes to start a transaction.
Meter	M	The meter is the device that measures the volume of fuel being delivered.
Product	PR	The product is the motor fuel dispensed. The product can be a base product or a blend product. ▪ A base product is a non blended motor fuel and is sourced directly from a tank. ▪ A blend product is a motor fuel that consists of two base products blended together at a given ratio.
Fuelling Mode	FM	The fuelling product could be dispensed in different modes (cash, credit, attendant, etc.)
Stand Alone	SA	The *dispenser* has a link to a *Controller Device*. The FP control (release, clear transaction) is done locally at the dispenser.
Dispenser Offline Mode	-	The *dispenser* is not controlled by a *Controller Device*. There is no link to a CD.
Dispenser Online Mode	-	The *dispenser* is controlled by a *Controller Device*.
CD Off-line Mode	-	CD is off-line when: The CD is not in the Communication Layers's Recipient Address Table The CD is in the Communication Layers's Recipient Address Table, but no heartbeat has been received in the expected time frame (3 x Heatbeat_Interval)..

CD On-line Mode	-	A CD is on-line when: The CD is entered in the Communication Layers's Recipient Address Table. A heartbeat has been received from the CD within the expected time frame (3 x Heatbeat_Interval).
Transaction Buffer	-	The finished fuelling transaction is stored in a transaction buffer.
Payable Transaction	-	A Payable Transaction is a finished fuelling transaction which must be cleared by a *Controller Device*.
Zero Transaction	-	A Zero Transaction is a finished fuelling transaction where the displayed volume and amount have the value of 0.
Outdoor Payment Terminal	OPT	A hardware device where the customer tenders payment for fuel, that is located outside a building.
Protocol Converter Device	PCD	A hardware device that converts the IFSF Dispenser protocol into a proprietary pump/dispenser protocol. This enables an IFSF compatible SC/CD to control non IFSF compatible pumps.
Proprietary Pump Protocol	PPP	A non-IFSF protocol developed and owned by the dispenser manufacturer.
Tank Level Gauge	TLG	A hardware device which measures the contents of a tank.
Logical Node Address	LNA	The LNA is the address that identifies a device on the IFSF network. The LNA consists of two bytes (Subnet & Node Address). Please reference the IFSF document "PART II, COMMUNICATION SPECIFICATION" for more details.
Physical Node Address	PNA	The PNA is the physical address that is used to physically address the device on the Echelon LonWorks network. The PNA consists of two bytes (Subnet & Node Address). Please note that the PNA may differ from the LNA. Please reference the IFSF document "PART II, COMMUNICATION SPECIFICATION" for more details.

2 Fuelling Point Behavioural Model

This chapter describes in detail each state, event and required actions of a fuelling point.

Protocol Converter Device Comment
It is not a straightforward task to convert one protocol to another when the two protocols have different state tables. However, the very nature of an IFSF Dispenser Protocol Converter requires the device to accomplish this task.
The following sections explain varies IFSF FP states, why they are in the respective state and what effect events have on the state table. In addition to the standard IFSF Dispenser Application text additional information has been included explaining what is expected of an IFSF Dispenser Protocol Converter. The PCD Comments are in *italic* to make it easier to recognise that the text is related to a PCD application.

In the following description **STATES** are shown in bold text and "EVENTS" are given in double quotes. [Control flows] and [Data flows] are contained in square brackets.

The table below is used. Its content has the following definition.

STATE DESCRIPTION	
STATE IDENTIFIER NAME	A short description of the state.
EVENT DESCRIPTION	
"EVENT-NAME"	A short description of the event. Used to describe to which new state the fuelling point has moved to, once all the actions are completed. *PCD Comment:* *A short description giving extra information needed for PCD implementations.* → Action: Input action description in terms of control and data flows between the CD and the FP. Action →: Output action description in terms of control and data flows between the FP and the CD.

The data elements which are sent by the control and data flows are described in chapter 3 "Dispenser Database".

Any change in the "Fuelling Point State", the "Transaction Buffer State", the "Logical Nozzle State" or the "FP Assign Control" is sent as an unsolicited message from the FP to the Controller Device.
The CD recipient addresses for the unsolicited messages are contained in the "Recipient Address Table" in the Communication Service Database (for further information see chapter 4.5 in the document "Part II, Communication Specification, Release 1.51").

2.1 Fuelling Point State Diagram

The fuelling point state diagram describes in detail the behaviour of the fuelling point in a dispenser.

States are represented on Figure 1 (FUELLING POINT STATE DIAGRAM) and Figure 2 (FUELLING POINT STATE DIAGRAM, ERROR CONDITIONS) by rectangles. The states are sequential numbered.

The arrows between the states are labelled with the event name or names that causes the FP to change from one state to another. The direction of state transfer is indicated by the arrowhead.

In Figure 3 all states and events are combined in a matrix.

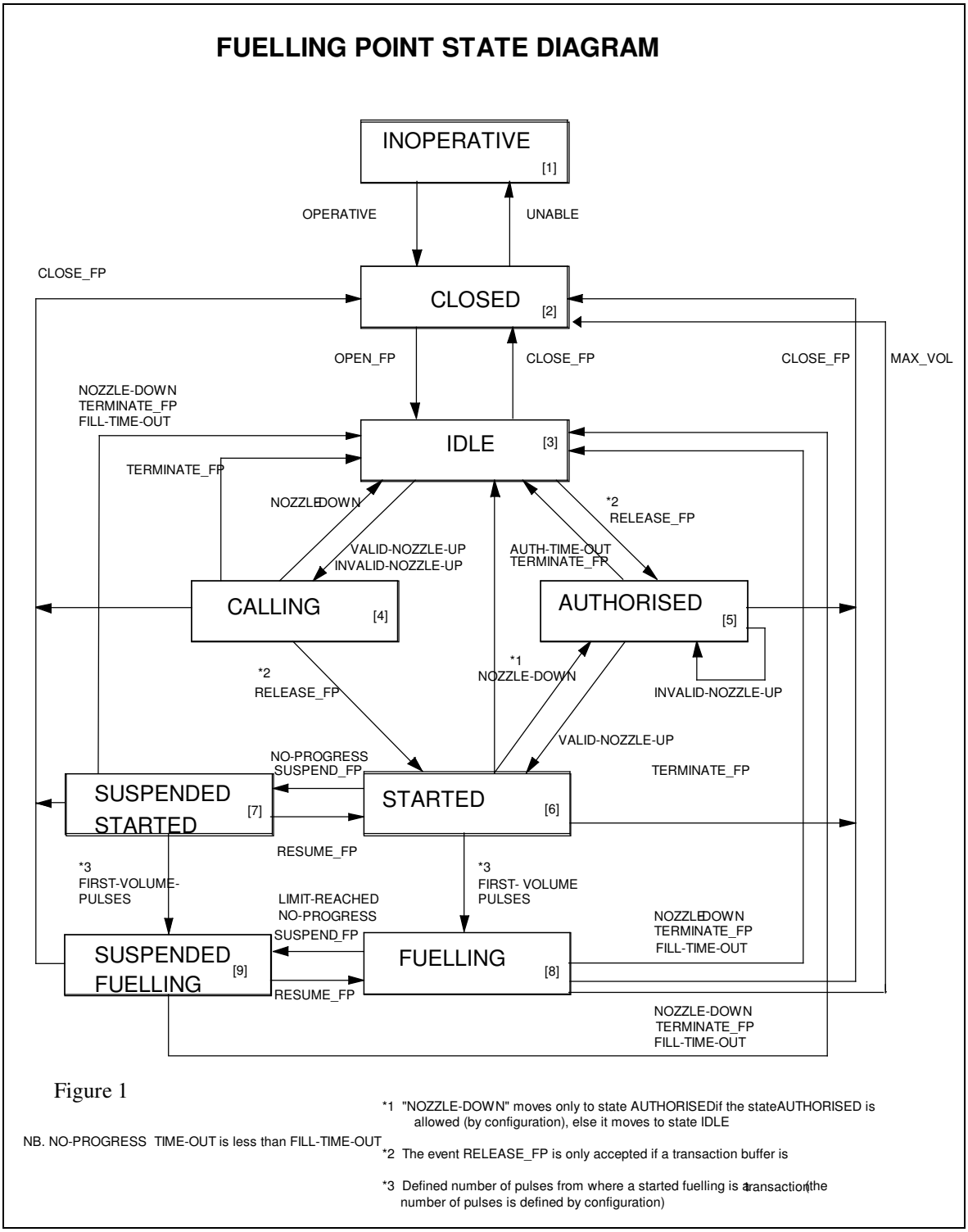


Figure 1

Note.
If more than one nozzle is removed, all nozzles after the first nozzle will be ignored. No nozzle out messages, etc will be sent.

FUELLING POINT STATE DIAGRAM
(ERROR CONDITIONS)

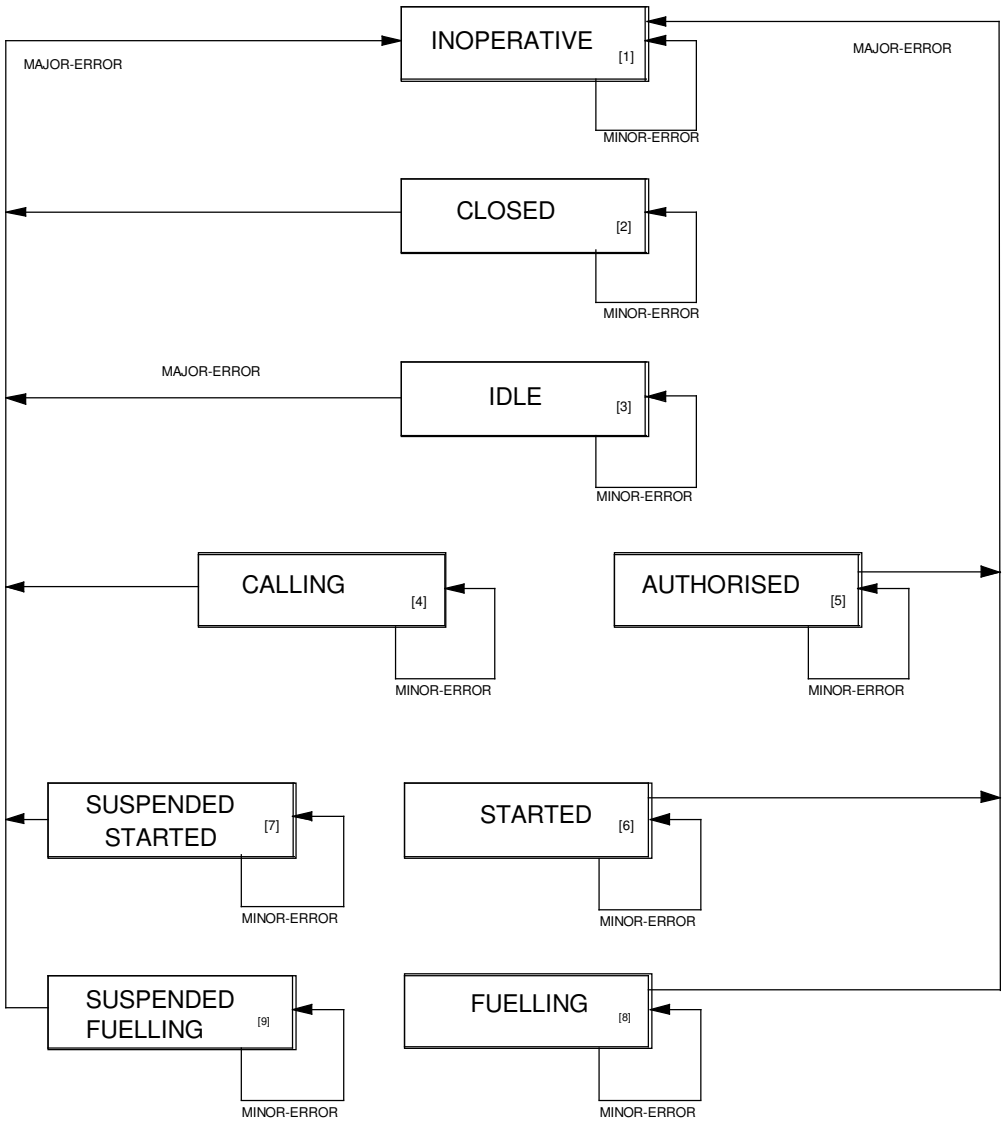


FIGURE 2

The classification of the errors (major error, minor error) is in the database description (chapter 3.11 ERROR CODE DATABASE

FIGURE 3 **FUELLING POINT STATE TABLE**

State Event	1 Inoperative	2 Closed	3 Idle	4 Calling	5 Authorised	6 Started	7 Suspended started	8 Fuelling	9 Suspended Fuelling
Operative	–> 2	-	-	-	-	-	-	-	-
Unable	1	–> 1	State error 6	State error 6	State error 6	State error 6	State error 6	State error 6	State error 6
Open_FP	State error 1	–> 3	3	State error 3	State error 3	State error 3	State error 3	State error 3	State error 3
Close_FP	State error 1	2	–> 2	–> 2	–> 2	–> 2	–> 2	–> 2	–> 2
Valid-nozzle-up	1	2	–> 4	-	–> 6	-	-	-	-
Invalid-nozzle-up	1	2	–> 4	-	5	-	-	-	-
Nozzle-down	1	2	3	–> 3	5	–> 5/3 *	–> 3	–> 3	–> 3
Release_FP	State error 1	State error 2	–> 5	–> 6	5	State error 5	State error 5	State error 5	State error 5
Auth-time-out	-	-	-	-	–> 3	-	-	-	-
Fill-time-out	-	-	-	-	-	-	–> 3	–> 3	–> 3
Suspend_FP	State error 1	State error 2	State error 4	State error 4	State error 4	–> 7	7	–> 9	9
Resume_FP	State error 1	State error 2	State error 4	State error 4	State error 4	6	–> 6	8	–> 8
Terminate_FP	State error 1	State error 2	3	–> 3	–> 3	–> 3	–> 3	–> 3	–> 3
No-progress	-	-	-	-	-	–> 7	7	–> 9	9
Limit-reached	-	-	-	-	-	-	7	–> 9	9
Max_Vol	-	-	-	-	-	-	-	–> 2	-
First-volume-pulses	-	-	-	-	-	–> 8	–> 9	-	-
Major-error	1	-> 1	–> 1	–> 1	–> 1	–> 1	–> 1	–> 1	–> 1
Minor-error	-	2	3	4	5	6	7	8	9

Description

State error 1	FP is in state INOPERATIVE
State error 2	FP is in state CLOSED
State error 3	FP is already opened
State error 4	Transaction not in progress
State error 5	Transaction already started
State error 6	Parameter / Configuration change not possible
n	no state change
➔ n	State changes to state n
■	not applicable
•	for details see event in state description

2.1.1 State Inoperative [1]

STATE DESCRIPTION	
INOPERATIVE	The FP is in the **INOPERATIVE** state when it is not possible to open a FP. The reason for this is that essential configuration data (e.g. W&M parameter) is missing or a major error has been detected. The FP will also be in the **INOPERATIVE** state during the changing of essential data (e.g. software download). Note: Payable transaction may exist. *PCD Comment:* *The PCD will indicate that an IFSF FP is inoperative when it can't open a proprietary pump or when the PCD itself is unable to operate.*
EVENT DESCRIPTION	
"OPERATIVE"	When the FP has been configured with the essential data to operate (W&M parameters, pump configuration parameters) and no major errors have been detected (see 3.11 Error Code Data), the FP goes to the **CLOSED** state. *PCD Comment:* *When the PCD detects that the proprietary FP has been configured with the correct data to operate it should change the IFSF FP state to the **CLOSED** state.* Action →: The FP state change is sent as an unsolicited data array [FP_Status_Message].
"MAJOR_ERROR"	If a major error event occurs the FP stays in the **INOPERATIVE** state. *PCD Comment:* *When the PCD detects a major error with the proprietary FP or with itself it must leave the IFSF FP status as **INOPERATIVE**.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].
"MINOR_ERROR"	If a minor error event occurs the FP does not change the state. *PCD Comment:* *When the PCD detects a minor error with the proprietary FP or with itself it must leave the IFSF FP status as **INOPERATIVE** and generate the respective IFSF error message.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].

2.1.2 State Closed [2]

STATE DESCRIPTION	
CLOSED	The FP is completely configured and no major error has been detected. The FP is waiting to be opened by a CD or the power to be switched off. This may be used to temporarily shut down one or more FP's when business is slack. The FP must respond to all communications from controller devices. *PCD Comment:* *If the proprietary FP does not have the equivalent **CLOSED** state the PCD will have to assure that no new transactions can start on the proprietary FP. Where possible proprietary protocol features that can be used to indicate that a FP is not available to the customer should be utilized (e.g. switching display lights off).* Note: Payable transactions may exist.
EVENT DESCRIPTION	
"UNABLE"	During configuration, changing essential parameter or a data download the FP is not able to work. During this time the FP's state changes to **INOPERATIVE**. *PCD Comment:* *The PCD can also change the IFSF FP state to INOPERATIVE when the proprietary FP or itself is having essential data/parameters changed or receiving a data download.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"OPEN_FP"	The FP will become available to the customer. An open command moves the FP into the **IDLE** state. → Action: The FP receives the [Open_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"MAJOR-ERROR"	If a major error event occurs the FP moves into the **INOPERATIVE** state. *PCD Comment:* *When the PCD detects a major error with the proprietary FP or with itself it must change the IFSF FP status to **INOPERATIVE**.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes]. The FP state change is send as an unsolicited data array [FP_Status_Message].
"MINOR-ERROR"	If a minor error event occurs the FP does not change the state. *PCD Comment:* *When the PCD detects a minor error with the proprietary FP or with itself it must leave the IFSF FP status as **CLOSED** and generate the respective IFSF error message.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].

2.1.3 State Idle [3]

STATE DESCRIPTION	
IDLE	The FP is opened and no delivery has started. On entry to the **IDLE** state any outstanding transactions have been stored in the transaction buffer and all fuelling parameters must have been reset to their default values. Note: When the **IDLE** state is entered with a nozzle removed, it is necessary to wait until the nozzle is returned before allowing the state to change away from the Idle state. This implies that any attempt to release the pump while the 'illegal' nozzle is removed must be rejected with a Data_ACK of 6 (Command not accepted). After the nozzle is returned a new transaction is able to start.
EVENT DESCRIPTION	
"VALID-NOZZLE-UP" "INVALID-NOZZLE-UP"	The customer selects any logical nozzle and the FP moves to the **CALLING** state. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"RELEASE_FP"	The pre-authorization can only be accepted if at least one transaction buffer is available. The number of transaction buffers is configured by the contents of the data element Nb_Tran_Buffer_Not_Paid. If there are is no unit price available any attempt to release a fuel pump should be rejected with a Data ACK of 6. A FP could be assigned to a CD by the contents of the data element Assign_Contr_Id (Data_Id 22 in the Fuelling Point Database). If the FP is assigned to a CD the FP can only be released by the CD that assigned it. Any fuelling limit must be transmitted to the FP before the Release_FP command is transmitted. The pre-authorization could be done without any limit, with a volume limit (preset mode) or an amount limit (prepay mode). The FP receives a pre-authorization and the FP moves to the **AUTHORISED** state. *PCD Comment:* *As most proprietary FP's only allow one transaction at a time, the PCD will have to manage multiple transaction itself (assuming that the PCD supports more than 1 transaction buffer).* *As most proprietary FP's don't support assignment, the PCD will have to manage the assignment regulations itself.* *As most proprietary FP's don't support pre-authorization, the PCD will have to manage the pre-authorization itself (e.g. be in a state where it will automatically release the proprietary FP when the customer removes the nozzle).* It is also possible to pre-authorise a FP locally by a sales assistant (on attendant operated FPs). Action: For preset or prepay mode the FP receives the [Remote_Volume_Preset] or [Remote_Amount_Prepay] data. The FP receives [Release_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].

"CLOSE_FP"	The FP will be closed down and the FP moves into the **CLOSED** state. This may be used to temporarily shut down one or more FP's when business is slack. *PCD Comment:* *Where possible the PCD must utilises proprietary FP protocol features that indicate that the FP is not available to the customer (e.g. switching display lights off).* ➔ Action: The FP receives a [Close_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"MAJOR-ERROR"	If a major error event occurs the FP moves into the **INOPERATIVE** state. *PCD Comment:* *When the PCD detects a major error with the proprietary FP or with itself it must change the IFSF FP status to **INOPERATIVE**.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes]. The FP state change is send as an unsolicited data array [FP_Status_Message].
"MINOR-ERROR"	If a minor error event occurs the FP does not change the state. *PCD Comment:* *When the PCD detects a minor error with the proprietary FP or with itself it must leave the IFSF FP status as **IDLE** and generate the respective IFSF error message.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].

2.1.4 State Calling [4]

STATE DESCRIPTION	
CALLING	A logical nozzle has been selected by a customer and the FP is waiting to be released.
EVENT DESCRIPTION	
"RELEASE_FP"	The release can only be accepted if at least one transaction buffer is available. The number of transaction buffers is configured by the contents of the data element Nb_Tran_Buffer_Not_Paid. If a transaction buffer is not available any attempt to release a fuel pump should be rejected with a Data ACK of 6. If the FP is assigned to a CD the FP can only be released by the CD that assigned it. Any fuelling limits and grade masks must be transmitted to the FP before the Release_FP command is transmitted. The release could be done without any limits, with a volume limit (preset mode) or an amount limit (prepay mode). If there are is no unit price available any attempt to release a fuel pump should be rejected with a Data ACK of 6. A Release command will be rejected if the customer has removed an invalid nozzle i.e. one not permitted in the grade mask (FP Data Base Data_Id 25 *Log_Noz_Mask*). If this situation occurs, the dispenser will reject the Release command with a Data_ACK value of 6 (Command not accepted). The FP receives a release and the FP moves to the **STARTED** state. It is also possible to release a FP locally by a sales assistant (on attendant operated FPs). *PCD Comment:* *As most proprietary FP's only allow one transaction at a time, the PCD will have to manage multiple transaction itself (assuming that the PCD supports more than 1 transaction buffer).* *As most proprietary FP's don't support assignment, the PCD will have to manage the assignment regulations itself.* *Where the proprietary FP protocol doesn't indicate the selected nozzle the PCD will have no choice but to ignore the Log_Noz_Mask and release the FP.* Action →: For preset or prepay mode the FP receives the [Remote_Volume_Preset] or [Remote_Amount_Prepay] data. The FP receives [Release_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"NOZZLE-DOWN"	The customer returns the first selected logical nozzle into its holster and the FP returns to the **IDLE** state. This allows the customer to select another logical nozzle from the same FP if the wrong one has been picked up. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].

"TERMINATE_FP"	The FP is forced to move to the **IDLE** state. *PCD Comment:* *As most proprietary FP's don't support the concept of 'terminating' a **CALLING** FP the PCD will have to manage the state transitions from **CALLING** to **IDLE**. This action will involve dealing with the proprietary FP being **IDLE** but having its nozzle removed.* ➔ Action: The FP receives a [Terminate_FP] command. Action →: The status change is send as an unsolicited data array [FP_Status_Message].
"CLOSE_FP"	The FP will be closed down and the FP moves into the **CLOSED** state. *PCD Comment:* *As most proprietary FP's don't support the concept of 'closing' a **CALLING** FP the PCD will have to manage the state transitions from **CALLING** to **CLOSED**. This action will involve dealing with the proprietary FP being **CLOSED** but having its nozzle removed.* ➔ Action: The FP receives a [Close_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"MAJOR-ERROR"	If a major error event occurs the FP moves into the **INOPERATIVE** the. *PCD Comment:* *When the PCD detects a major error with the proprietary FP or with itself it must change the IFSF FP status to **INOPERATIVE**.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes]. The FP state change is send as an unsolicited data array [FP_Status_Message].
"MINOR-ERROR"	If a minor error event occurs the FP does not change the state. *PCD Comment:* *When the PCD detects a minor error with the proprietary FP or with itself it must leave the IFSF FP status as **CALLING** and generate the respective IFSF error message.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].

2.1.5 State Authorised [5]

STATE DESCRIPTION	
AUTHORISED	The FP has been pre-authorised and it is now waiting for the customer to select a valid logical nozzle (grade selection and physical nozzle). Coming into this state the timer for the maximum authorization time Max_Auth_Time is started. The customer display could be reset in this state (dictated by the contents of the data element Clear_Display_Mode). *PCD Comment:* *As most proprietary FP's don't support pre-authorization, the PCD will have to manage the pre-authorization itself (e.g. be in a state where it will automatically release the proprietary FP when the customer removes the nozzle).* Note: This state is not allowed in same countries (configured by the contents of the data element Auth_State_Mode).
EVENT DESCRIPTION	
"VALID-NOZZLE-UP"	The customer selects a valid logical nozzle (dictated by the product/logical nozzle restrictions) and the FP moves into the **STARTED** state. *PCD Comment:* *If the proprietary FP protocol doesn't indicate the nozzle selected then the PCD will not be able to determine if the selected nozzle is valid or invalid. Hence the PCD will have to treat all nozzles as being valid (as if the Log_Noz_Mask had been set to 255/FFH).* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"INVALID-NOZZLE-UP"	The customer selects an invalid logical nozzle (dictated by the product/logical nozzle restrictions) and the FP stays in the **AUTHORISED** state. *PCD Comment:* *If the proprietary FP protocol doesn't indicate the nozzle selected then the PCD will not be able to determine if the selected nozzle is valid or invalid. Hence the PCD will have to treat all nozzles as being valid (as if the Log_Noz_Mask had been set to 255/FFH).* Action →: The FP sends the unsolicited data array [FP_Status_Message].

"AUTH-TIME-OUT"	A nozzle is not removed during a period of time (configured by the contents of the data element Max_Auth_Time) and the FP returns to the **IDLE** state. *PCD Comment:* *As most proprietary pump protocols do not support maximum authorization timeouts the PCD will have to carry out the watchdog timing itself and when the timer has expired, automatically clear the pre-authorization and move the IFSF FP status back to **IDLE**.* If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"TERMINATE_FP"	The FP is forced to move to the **IDLE** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *As most proprietary FP's don't support the concept of 'terminating' an **AUTHORIZED** FP the PCD will have to manage the state transitions from **ATHORIZED** to **IDLE**.* → Action: The FP receives the [Terminate_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"CLOSE_FP"	The FP will be closed down and the FP moves into the **CLOSED** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *As most proprietary FP's don't support the concept of 'closing' an **AUTHORISED** FP the PCD will have to manage the state transitions from **AUTHORISED** to **CLOSED**.* → Action: The FP receives a [Close_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"MAJOR-ERROR"	If a major error event occurs the FP moves into the **INOPERATIVE** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *When the PCD detects a major error with the proprietary FP or with itself it must change the IFSF FP status to **INOPERATIVE**.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes]. The FP state change is send as an unsolicited data array [FP_Status_Message].

"MINOR-ERROR"	If a minor error event occurs the FP does not change the state. *PCD Comment:* *When the PCD detects a minor error with the proprietary FP or with itself it must leave the IFSF FP status as **AUTHORISED** and generate the respective IFSF error message.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].

2.1.6 State Started [6]

STATE DESCRIPTION	
STARTED	This state implies that the FP was released and a valid logical nozzle has been selected by the customer. This means (explicitly) that the actual fuel transaction (filling up) has not yet started until a defined minimum volume has been registered (configured by the contents of the Min_Fuelling_Vol). Coming into this state the timer for the maximum filling time Max_Fill_Time is started. The timer for the maximum authorization time is stopped. *PCD Comment:* *If the proprietary pump protocol doesn't have the equivalent **STARTED** state and goes straight from **CALLING** to **FUELLING** then the PCD should create a dummy **STARTED** state that it resides in only for the length of time it takes to inform the CD that it is in this state. After the state change has been notified to the CD the PCD can change the state to **FUELLING**.* The customer display could be reset in this state (configured by the contents of the data element Clear_Display_Mode).
EVENT DESCRIPTION	
"NOZZLE-DOWN"	The customer returns the nozzle and the FP moves into the **AUTHORISED** state. In this event a very important customer tolerant feature is satisfied; the customer may have selected the wrong grade (i.e. the wrong nozzle) and, so long as dispensing has not started (state **FUELLING**) he is able to select another if he wishes. Note: In some countries the **AUTHORISED** state is not permitted. In this case the FP returns to state **IDLE**. The way of going back is defined by configuration in the contents of the data element Auth_State_Mode. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"FIRST-VOLUME-PULSES"	The customer presses the trigger mechanism on the safety nozzle and the flow meter registers a preset minimum volume signifying that dispensing has started. The FP moves into the **FUELLING** state. The minimum volume is defined by configuration (contents of data element Min_Fuelling_Vol). Note that the minimum volume for the first display update (contents of data element Min_Display_Vol) could be different from the minimum volume to start a transaction. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"SUSPEND_FP"	The FP receives a suspend command for whatever reason and the FP moves into the **SUSPENDED STARTED** state. → Action: The FP receives the [Suspend_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message]. Action →: The FP sends an unsolicited data array [FP_Error_Type_Mess] with the minor error **Suspended_ Fuelling** to the CD and the error is stored within TR_Error_Code

"NO-PROGRESS"	This event occurs when a FP is released and a valid logical nozzle is selected but no volume pulses are registered within a defined period of time (configured by the contents of the data element Max_Time_W/O_Prog). The FP moves to the **SUSPENDED STARTED** state. *PCD Comment:* *If the proprietary pump protocol doesn't indicate that volume pulses have been generated the PCD will have to ignore this event.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message]. Action →: The FP sends an unsolicited data array (FP_Error_Type_Mess) with minor error **No_Progress** to the CD and the error is stored within *TR_Error_Code*.
"TERMINATE_FP"	The FP is forced to move to the **IDLE** state. *PCD Comment:* *As most proprietary FP's don't support the concept of 'terminating' a **STARTED** FP the PCD will have to manage the state transitions from **STARTED** to **IDLE**. This action will involve dealing with the proprietary FP being **IDLE** but having its nozzle removed.* If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. ➔ Action: The FP receives the [Terminate_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"CLOSE_FP"	The FP will be closed down and the FP moves into the **CLOSED** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *As most proprietary FP's don't support the concept of 'closing' a **STARTED** FP the PCD will have to manage the state transitions from **STARTED** to **CLOSED**. This action will involve dealing with the proprietary FP being **CLOSED** but having its nozzle removed.* ➔ Action: The FP receives a [Close_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].

"MAJOR-ERROR"	If a major error event occurs the FP moves into the **INOPERATIVE** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *When the PCD detects a major error with the proprietary FP or with itself it must change the IFSF FP status to **INOPERATIVE**.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes]. The FP state change is send as an unsolicited data array [FP_Status_Message].
"MINOR-ERROR"	If a minor error event occurs the FP does not change the state. *PCD Comment:* *When the PCD detects a minor error with the proprietary FP or with itself it must leave the IFSF FP status as **STARTED** and generate the respective IFSF error message.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].

2.1.7 State Suspended Started [7]

STATE DESCRIPTION	
SUSPENDED STARTED	The transaction was suspended while in the **STARTED** state. *PCD Comment:* *In some cases proprietary pump protocols will not allow a suspended pump to be re-started. Please see the text detailing how the PCD will have to treat this situation.*
EVENT DESCRIPTION	
"RESUME_FP"	When the FP is resumed the same transaction continues from where it was paused, the state changes to the **STARTED** state. Only the device that has suspended the transaction can restart it (for exceptions see data variable Suspend_Contr_Id). *PCD Comment:* *If the proprietary pump protocol doesn't allow a suspended pump to be re-started then the PCD should NACK the Resume_FP command (with a MS_ACK=5 & Data_ACK=5) and stay in **the SUSPENDED STARTED** state.* ➔ Action: The FP receives the [Resume_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message]. Action →: The FP sends an unsolicited data array [FP_Error_Type_Mess] with the minor error **Fuelling_Resumed** to the CD and the error is stored within TR_Error_Code.
"FIRST-VOLUME-PULSES"	The suspend command was received during the first pulses and the flow meter registers a preset minimum number of pulses signifying that dispensing has started just after reaching the **SUSPENDED STARTED** state. The FP moves into the **SUSPENDED FUELLING** state. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"NOZZLE-DOWN"	The customer finishes a started fuelling (no volume pulses are registered) by returning the nozzle. The FP returns to the **IDLE** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *If the proprietary pump protocol doesn't support zero transactions the PCD will have to recognise the zero transaction situation and store the respective transaction details in the transaction buffer.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].

"FILL-TIME-OUT"	The FP times out when the duration of the fuelling operation exceeds the maximum time allowed for this product (defined by the contents of the data element Max_Fill_Time). The FP moves back to the **IDLE** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *If the proprietary pump protocol doesn't support zero transactions the PCD will have to recognise the zero transaction situation and store the respective transaction details in the transaction buffer.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"TERMINATE_FP"	The FP is forced to move to the **IDLE** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *As most proprietary FP's don't support the concept of 'terminating' a **SUSPENDED STARTED** FP the PCD will have to manage the state transitions from **SUSPENDED STARTED** to **IDLE**. This action will involve dealing with the proprietary FP being **IDLE** but having its nozzle removed. The PCD must stop the proprietary pump dispensing fuel.* *If the proprietary pump protocol doesn't support zero transactions the PCD will have to recognise the zero transaction situation and store the respective transaction details in the transaction buffer.* ➔ Action: The FP receives the [Terminate_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"CLOSE_FP"	The FP will be closed down and the FP moves into the **CLOSED** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *As most proprietary FP's don't support the concept of 'closing' a **SUSPENDED STARTED** FP the PCD will have to manage the state transitions from **SUSPENDED STARTED** to **CLOSED**. This action will involve dealing with the proprietary FP being **CLOSED** but having its nozzle removed. The PCD must stop the proprietary pump dispensing fuel.* *If the proprietary pump protocol doesn't support zero transactions the PCD will have to recognise the zero transaction situation and store the respective transaction details in the transaction buffer.* ➔ Action: The FP receives a [Close_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].

"MAJOR-ERROR"	If a major error event occurs the FP moves to the **INOPERATIVE** state. If a "zero transaction" is required (dictated by the contents of the data element ZeroTR_Mode") the transaction with a zero value must be stored in the transaction buffer. *PCD Comment:* *When the PCD detects a major error with the proprietary FP or with itself it must change the IFSF FP status to **INOPERATIVE**.* *If the proprietary pump protocol doesn't support zero transactions the PCD will have to recognise the zero transaction situation and store the respective transaction details in the transaction buffer.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes]. The FP state change is send as an unsolicited data array [FP_Status_Message].
"MINOR-ERROR"	If a minor error event occurs the FP does not change the state. *PCD Comment:* *When the PCD detects a minor error with the proprietary FP or with itself it must leave the IFSF FP status as **SUSPENDED STARTED** and generate the respective IFSF error message.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].

2.1.8 State Fuelling [8]

STATE DESCRIPTION	
FUELLING	The FP has now dispensed at least a minimum quantity of fuel (configured by the contents of the data element Min_Fuelling_Vol) and is in the **FUELLING** state. Observe that the FP can never return directly to the **STARTED** state from this state.
EVENT DESCRIPTION	
"NOZZLE-DOWN"	The customer finishes the transaction by returning the nozzle. The transaction is stored in the transaction buffer and the FP moves to the **IDLE** state. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"SUSPEND_FP"	The FP receives a suspend command for whatever reason and the FP moves into the **SUSPENDED FUELLING** state. → Action: The FP receives the [Suspend_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message]. Action →: The FP sends an unsolicited data array [FP_Error_Type_Mess] with the minor error **Suspended_Fuelling** to the CD and the error is stored within TR_Error_Code.
"LIMIT-REACHED"	When the dispensed volume (as calculated by the DC) equals the maximum permitted quantity (Remote_Volume_Preset or Remote_Amount_Prepay or User_Max_Amount) the event "LIMIT REACHED" occurs. The FP moves into the **SUSPENDED FUELLING** state. The nozzle at this time is still out. *PCD Comment:* *As some proprietary pump protocols don't indicate when a pump transaction has reached the supported limits it is impossible for a PCD to always recognize this event. In the case where this event isn't recognized the PCD will simply never move from the **FUELLING** state into the **SUSPENDED FUELLING** state.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message]. Action → The FP sends an unsolicited data array [*FP_Error_Type_Mess*] with minor error **Limit_Reached** to the CD and the error is stored within *TR_Error_Code*.

"MAX_VOL"	When the dispensed volume (as calculated by the DC) equals the Max_Vol which is specified by a W&M limit, the FP moves into a closed state. *PCD Comment:* *As some proprietary pump protocols don't indicate when a pump transaction has reached the supported limits it is impossible for a PCD to always recognize this event. In the case where this event isn't recognized the PCD will simply never move from the **FUELLING** state into the **SUSPENDED FUELLING** state.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message]. Action →: The FP sends an unsolicited data array [*FP_Error_Type_Mess*] with minor error **Limit_Reached** to the CD and the error is stored within *TR_Error_Code*.
"NO-PROGRESS"	The FP times out when no pulses have been detected for a period greater than the defined time (Max_Time_W/O_Prog). The FP moves into the **SUSPENDED FUELLING** state. *PCD Comment:* *If the proprietary pump protocol doesn't support the Max_Time_W/O_Prog then the PCD will have to create its own watchdog timer for this purpose and when the timer has expired must stop the pump dispensing and move into the **SUSPENDED FUELLING** state.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message]. Action →: The FP sends an unsolicited data array [*FP_Error_Type_Mess*] with minor error **No_Progress** to the CD and the error is stored within *TR_Error_Code*.
"FILL-TIME-OUT"	The FP times out when the duration of the fuelling operation exceeds the maximum time allowed for that product (Max_Fill_Time). The transaction is stored in the transaction buffer and the FP moves to the **IDLE** state. *PCD Comment:* *If the proprietary pump protocol doesn't support the Max_Fill_Time then the PCD will have to create its own watchdog timer for this purpose and when the timer has expired must stop the pump dispensing and move into the **IDLE** state.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message]. Action →: The FP sends an unsolicited data array [*FP_Error_Type_Mess*] with minor error **Fill_Time_Out** to the CD and the error is stored within *TR_Error_Code*.
"TERMINATE_FP"	The FP is forced to move to the **IDLE** state. The transaction is stored in the transaction buffer. *PCD Comment:* *As most proprietary FP's don't support the concept of 'terminating' a **FUELLING** FP the PCD will have to manage the state transitions **FUELLING** to **IDLE**. This action will involve dealing with the proprietary FP being **IDLE** but having its nozzle removed. The PCD must stop the proprietary pump dispensing fuel.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].

"CLOSE_FP"	The FP will be closed down and the FP moves into the **CLOSED** state. The transaction is stored in the transaction buffer. *PCD Comment:* *As most proprietary FP's don't support the concept of 'closing' a **FUELLING** FP the PCD will have to manage the state transitions from **FUELLING** to **CLOSED**. This action will involve dealing with the proprietary FP being **CLOSED** but having its nozzle removed. The PCD must stop the proprietary pump dispensing fuel.* ➔ Action: The FP receives a [Close_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"MAJOR-ERROR"	If a major error event occurs the FP must store the transaction in the transaction buffer (it must include the error code that caused the transaction to be terminated). The FP moves to the **INOPERATIVE** state. *PCD Comment:* *When the PCD detects a major error with the proprietary FP or with itself it must change the IFSF FP status to **INOPERATIVE**.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes]. The FP state change is send as an unsolicited data array [FP_Status_Message].
"MINOR-ERROR"	If a minor error event occurs the FP does not change the state. *PCD Comment:* *When the PCD detects a minor error with the proprietary FP or with itself it must leave the IFSF FP status as **FUELLING** and generate the respective IFSF error message.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].

2.1.9 State Suspended Fuelling [9]

STATE DESCRIPTION	
SUSPENDED FUELLING	The FP has been suspended while in the **FUELLING** state. It may be restarted. It is also possible for the FP state to change from **SUSPENDED STARTED** to this state (the dispensed volume is greater than defined in the data variable Min_Fuelling_Vol).
EVENT DESCRIPTION	
"RESUME_FP"	When the FP is released again by the same device (for exception see data variable Suspend_Contr_Id) the same transaction continues from where it was paused. The FP returns to the **FUELLING** state. *PCD Comment:* *If the proprietary pump protocol doesn't allow a suspended pump to be re-started then the PCD should NACK the Resume_FP command (MS_ACK=5 Data-ACK=5) and stay in the **SUSPENDED FUELLING** state.* ➔ Action: The FP receives the [Resume_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message]. Action →: The FP sends an unsolicited data array [FP_Error_Type_Mess] with the minor error **Fuelling_Resumed** to the CD and the error is stored within TR_Error_Code.
"NOZZLE-DOWN"	The customer finishes the fuelling by returning the nozzle. The transaction is stored in the transaction buffer. The FP moves to the **IDLE** state. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"FILL-TIME-OUT"	The FP times out when the duration of the fuelling operation exceeds the maximum time allowed for that product (Max_Fill_Time). The transaction is stored in the transaction buffer. The FP moves to the **IDLE** state. *PCD Comment:* *If the proprietary pump protocol doesn't support the Max_Fill_Time then the PCD will have to create its own watchdog timer for this purpose and when the timer has expired must stop the pump dispensing and move into the **IDLE** state.* Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"TERMINATE_FP"	The FP is terminated for whatever reason. The transaction is stored in the transaction buffer and the FP moves to the **IDLE** state. *PCD Comment:* *As most proprietary FP's don't support the concept of 'terminating' a **SUSPENDED FUELLING** FP the PCD will have to manage the state transitions from **SUSPENDED FUELLING** to **IDLE**. This action will involve dealing with the proprietary FP being **IDLE** but having its nozzle removed. The PCD must stop the proprietary pump dispensing fuel.* ➔ Action: The FP receives a [Terminate_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].

"CLOSE_FP"	The FP will be closed down. The transaction is stored in the transaction buffer and the FP moves into the **CLOSED** state. *PCD Comment:* *As most proprietary FP's don't support the concept of 'closing' a **SUSPEND FUELLING** FP the PCD will have to manage the state transitions from **SUSPEND FUELLING** to **CLOSED**. This action will involve dealing with the proprietary FP being **CLOSED** but having its nozzle removed. The PCD must stop the proprietary pump dispensing fuel.* Action: The FP receives a [Close_FP] command. Action →: The FP state change is send as an unsolicited data array [FP_Status_Message].
"MAJOR-ERROR"	If a major error event occurs the FP must store the transaction in the transaction buffer (it must include the error code). The FP moves to the **INOPERATIVE** state. *PCD Comment:* *When the PCD detects a major error with the proprietary FP or with itself it must change the IFSF FP status to **INOPERATIVE**.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes]. The FP state change is send as an unsolicited data array [FP_Status_Message].
"MINOR-ERROR"	If a minor error event occurs the FP does not change the state. *PCD Comment:* *When the PCD detects a minor error with the proprietary FP or with itself it must leave the IFSF FP status as **SUSPENDED FUELLING** and generate the respective IFSF error message.* Action →: The FP sends the unsolicited data [FP_Error_Type_Mes].

2.2 Transaction Buffer State Diagram

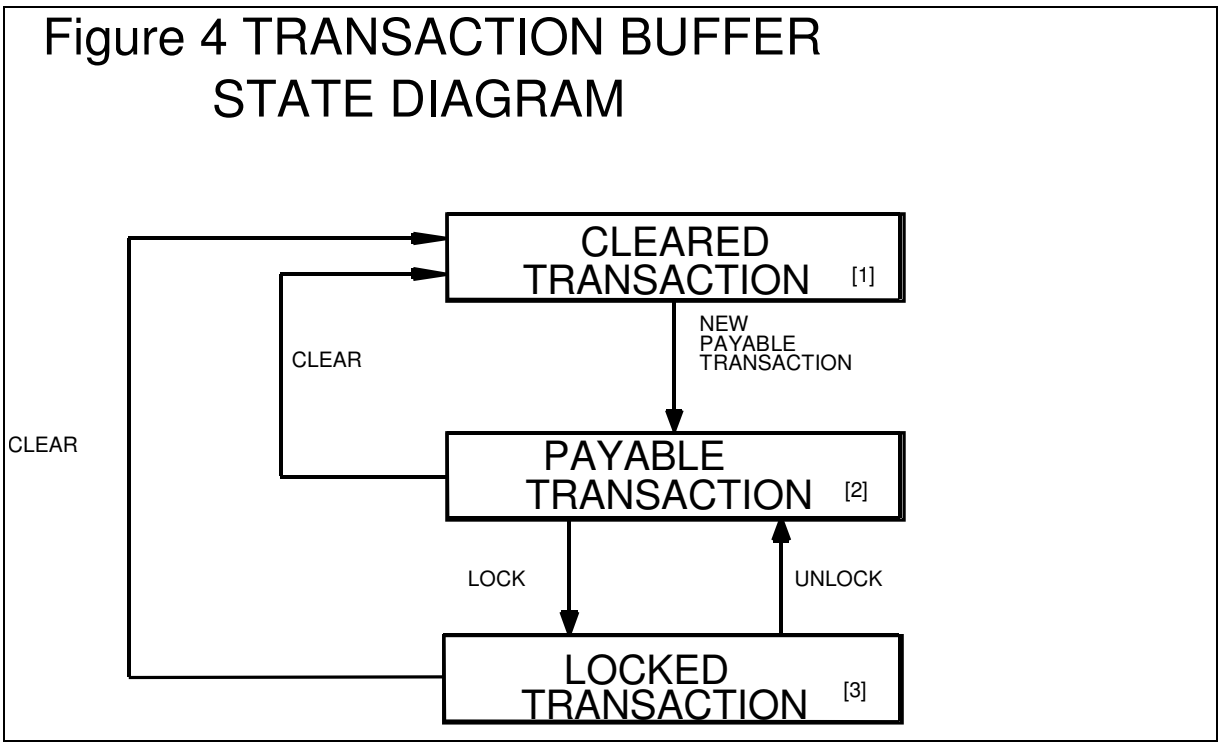
Every fuelling point has a defined number of transaction buffers (configured by the data element *Nb_Tran_Buffer_Not_Paid*) which are used for unpaid fuelling transactions. As long as the Controller Device has not cleared the fuelling transaction the FP is responsible for the transaction and the fuelling transaction data must be stored at the dispensers FP.

After a fuelling transaction is cleared by a CD (Transaction Buffer State = 1) transaction data are still available. The number of historic transactions is configured by the data element *Nb_Of_Historic_Trans*. Only the latest transaction data are available (first in, first out).

PCD Comment:
The PCD must be 100% compliant with the Transaction Buffer State Handling.

Every transaction buffer has the following state machine:

Figure 4 TRANSACTION BUFFER STATE DIAGRAM



2.2.1 State Cleared Transaction [1]

STATE DESCRIPTION	
CLEARED TRANSACTION	This particular transaction buffer is available for the next fuelling transaction. The CD has access to the previous transaction data (configured by the contents of the data element Nb_Of_Historic_Trans).
EVENT DESCRIPTION	

"NEW PAYABLE TRANSACTION"	The customer has finished the fuelling. The transaction must be stored in a cleared transaction buffer. The transaction buffer with the oldest transaction data is used to store the new payable fuelling. The transaction buffer state moves to the **PAYABLE TRANSACTION** state. Action →: The FP sends the transaction buffer state change as an unsolicited data array [TR_Buff_Status_Message].

2.2.2 State Payable Transaction [2]

STATE DESCRIPTION	
PAYABLE TRANS-ACTION	The customer has finished the fuelling and in the particular transaction buffer is now a payable transaction.
EVENT DESCRIPTION	
"CLEAR"	The FP receives a clear command indicating that the transaction buffer is available for a new fuelling a that the transaction details were read. The transaction buffer state moves to the **CLEARED TRANSACTION** state. If the FP runs in "stand alone" mode the transaction data totalized and the buffer is cleared automatically. The transaction buffer state moves to the **CLEARED TRANSACTION** state. → Action: The FP receives a [Clear_Transaction] command. Action →: The transaction buffer state change is send as an unsolicited data array [TR_Buff_Status_Message].
"LOCK"	The FP receives a command to reserve the payable transaction in this particular transaction buffer. The fuelling transaction can now only be cleared by the "locking" CD. The transaction buffer state moves to the **LOCKED TRANSACTION** state. → Action: The FP receives the data [Trans_Buff_Contr_Id]. Action →: The transaction buffer state change is send as an unsolicited data array [TR_Buff_Status_Message].

2.2.3 State Locked Transaction [3]

STATE DESCRIPTION	
LOCKED TRANSAC-TION	The payable transaction is reserved by a CD. No other CD can clear the transaction data (see special exception in the "UNLOCK" description). Please note that a transaction resulting from a assigned FP will be flagged as 'Locked' as soon as it is stored.
EVENT DESCRIPTION	
"CLEAR"	The FP receives a clear command indicating that the transaction buffer is available for a new fuelling. The transaction buffer state moves to the **CLEARED TRANSACTION** state. → Action: The FP receives a [Clear_Transaction] command. Action →: The transaction buffer state change is send as an unsolicited data array [TR_Buff_Status_Message].

"UNLOCK"	If a CD has locked the wrong payable transaction it is possible to unlock it but the transaction can only be unlocked by the CD that locked it. A special exception condition exists where the CD that locked the transaction is not able to unlock it or clear it due to a fatal error. A fatal error may be that the CD has crashed or is no longer on-line. In this exceptional case any CD that generates an Unlock command with the Originator Subnet set to 0 and the Originator Node set to 0 may unlock the transaction. The transaction buffer state moves back to the **PAYABLE TRANSACTION** state. → Action: The FP receives the data [Trans_Buff_Contr_Id]. Action →: The transaction buffer state change is send as an unsolicited data array [TR_Buff_Status_Message].

3 Dispenser Database

This part of the document details the standard data organisation for a Dispenser.

Every data element in the dispenser database is described in this chapter. The access to the data element is done by a Database Address "**DB_Ad**" and a Data_Identifier "**Data_Id**".

The data elements are presented in the following form:

DATABASE DB_Ad =				
Data _Id	***Data Element Name*** Description	Field Type	Read/Write in State	M/O

The Data_Id is an unique identifier for a data element in a database. The database is defined by the database address "DB_Ad" (for details see document "Part II, Communication Specification).

In the second column the name of the data element is defined. In this column is also the description of the data element (Including PCD comments in *italic* text).

The field types in column three are described in chapter 3.2 of this document.

The "Read/Write in State" column indicates if the related data can be Read and/or Written by any device and in which Fuelling Point state (states are indicated between brackets).

The M/O column (Mandatory/Optional) indicates if the data element must be supported / implemented by the Fuelling Points and any Controller Device controlling Fuelling Points. "M" indicates that the data element must be supported, "O" indicates that the data element is optional. Note: All mandatory data elements must be supported/implemented for a device to be IFSF compatible.

3.1 Database Address

Every data element in a device is stored in a database. In some implementation it may be real database or only a software organisation (object or tasks), for instance if a separate processor manages each meter.

These database levels are addressed by the Database Address (DB_Ad) using a variable number of bytes. The number of address bytes to specify a database is 1 to 8.
(For more details are in the document "PART II, COMMUNICATION SPECIFICATION").

<table>
<tr><th colspan="8">Database Address DB_Ad</th></tr>
<tr><th>BYTE 1</th><th>BYTE 2</th><th>BYTE 3</th><th>BYTE 4</th><th>BYTE 5</th><th>BYTE 6</th><th>BYTE 7</th><th>BYTE 8</th></tr>
<tr><td>COM_SV 00H
Commun-cation Service Data</td><td colspan="7"></td></tr>
<tr><td>C_DAT 01H
Calcul-ator
Data</td><td colspan="7"></td></tr>
<tr><td>FP_ID 21H-24H
Fuelling Point Identifier (1-4)</td><td colspan="7"></td></tr>
<tr><td rowspan="3">FP_ID 21H-24H
Fuelling Point Identifier (1-4)</td><td>LN_ID 11H-18H
Logical Nozzle Identifier (1-8)</td><td colspan="6"></td></tr>
<tr><td>TR_DAT 21H
Trans-action Data</td><td colspan="2">TR_Seq_Nb 0001-9999
Transaction Sequence Number
(bcd4 format)</td><td colspan="4"></td></tr>
<tr><td>ER_DAT 41H
Error Data</td><td>ER_ID 01H-FFH
Error Identifier (0-255)</td><td colspan="5"></td></tr>
<tr><td>PR_ID 41H-48H
Product Identifier (1-8)</td><td colspan="7"></td></tr>
</table>

Database Address DB_Ad							
BYTE 1	**BYTE 2**	**BYTE 3**	**BYTE 4**	**BYTE 5**	**BYTE 6**	**BYTE 7**	**BYTE 8**
PR_DAT 61H Product Data	**Prod_Nb 00000001-99999999** Product Number (bcd8 format)				**FM_ID 11H-18H** Fuelling Mode Identifier (1-8)		
M_ID 81H-90H Meter Identifier (1-16)							
SW_DAT A1H Software and Data Down-load							

3.2 Common Field Formats

IFSF application Field Formats are given in IFSF Engineering Bulletin No. 11. The following statement is made for fields of type Volume.

Field	Format	Description
VOLUME	bin8 + bcd8	Volume value (used for fuelling transaction data). Please note that the unit of volume is implied. I.E. if the dispenser is installed in a country where the unit of volume is **Litres**, then the volume will be in **Litres**. Alternatively, if the dispenser is installed in a country where the unit of volume is **Gallons**, then all volume will be in **Gallons**.

3.3 Calculator Data

This data allows the CD to configure the calculator in the dispenser.
The access to the calculator database is done by the database address C_DAT (Calculator Data).
All Fuelling Points have to be in the indicated state because the updated data are common to the different fuelling points.

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
CONFIGURATION DATA				
2 (02H)	***Nb_Products*** Number of products defined. 0 = not configured n = number of products Please note that dispensers that do not permit the *Nb_Products* to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Nb_Products* to the value of products that is hard coded in their program. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value.	Bin8 (1-8)	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
3 (03H)	***Nb_Fuelling_Modes*** Number of fuelling modes defined. 0 = not configured n = number of fuelling modes Please note that dispensers that do not permit the *Nb_Fuelling_Modes* to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Nb_Fuelling_Modes* to the value of fuelling modes that is hard coded in their program. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value.	Bin8 (1-8)	R(1-9) W(1-2)	M
4 (04H)	***Nb_Meters*** Number of meter defined. 0 = not configured n = number of meters Please note that dispensers that do not permit the *Nb_Meters* to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Nb_Meters* to the value of meters that is hard coded in their program. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value.	Bin8 (1-16)	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
5 (05H)	***Nb_FP*** Number of fuelling points controlled by the dispenser calculator. 0 = not configured n = number of fuelling points Please note that dispensers that do not permit the *Nb_FP* to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Nb_FP* to the value of fuelling points that is hard coded in their program. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. The relationship between Fuelling Point Numbers and Fuelling Point Identifications is NOT fixed e.g. FP1 does not necessarily have to be address 21H. In most cases a single sided dispenser is the same as a double sided dispenser with only one fuelling point. Usually a left handed single sided dispenser is Side 1 and will be database address 21H. A right handed single sided dispenser is Side 2 and will be database address 22H. Controller Devices should map the relationship between Fuelling Point Numbers and Fuelling Point Identifications. The default value should be non zero and is determined by the physical number of fuelling points on the dispenser.	Bin8 (1-4)	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
6 (06H)	***Country_Code*** Country where the dispenser is installed. The Country_Code uses the International PTT dialling code from the country where it is or the ISO 3166 standard. Examples of International PTT dialling codes are: 0030-Greece, 0031-Netherlands, 0032-Belgium, 0033-France, 0034-Spain, 0351-Portugal, 0352-Luxembourg, 0353-Ireland, 0354-Iceland, 0358-Finland, 0359-Bulgaria, 0036-Hungary, 0039-Italy, 0040-Rumania, 0041-Switzerland, 0042-Czech, 0043-Austria, 0044-United Kingdom, 0045-Denmark, 0046-Sweden, 0047-Norway, 0048-Poland, 0049-Germany, 0090-Turkey When the ISO 3166 standard is used the most significant digit must be set to 9 and the other three less significant digits must be set to the respective countries ISO 3166 three digit code (i.e. 9xxx). This allows the reading device to establish that the country code being returned is following the ISO 3166 convention. Please note that dispensers that do not permit the *Country_Code* to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Country_Code* to the hardcoded country code value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value.	Bcd4	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
7 (07H)	***Blend_Tolerance*** Specifies the blending error tolerance, the percentage (0-99%) indicates the calculation accuracy. 0 = no control is done Please note that dispensers that do not permit the *Blend_Tolerance* to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Blend_Tolerance* to the hardcoded country code value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value.	Bcd2 (0-99)	R(1-9) W(1-2)	M
8 (08H)	***Drive_Off_Lights_Mode*** The external visible status light for "drive off" could be controlled on different ways: 0 = drive off light not used 1 = remote control by the Drive_Off_Light_Switch 2 = internal control: Red light mode (light is on when a fuelling has started, it goes off when the fuelling transaction is paid) 3 = internal control: Green light mode (FP is available for the next fuelling) *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	bin8 (0-3)	R(1-9) W(1-2)	O

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
9 (09H)	***OPT_Light_Mode*** The external visible status light for up to four OPT-use could be controlled on different ways: 0-1 reference light 1 2-3 reference light 2 4-5 reference light 3 6-7 reference light 4 and values of: 0 = OPT light not used 1 = remote control by the OPT_Light_Switch 2 = internal control (light is on, when the release is done by the assigned CD) A OPT_Light_Mode value of for example 80h would mean light 4 on internal control lights one, two and three not used. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	bin8 (0-255)	R(1-9) W(1-2)	O

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
10 (0AH)	***Clear_Display_Mode*** The 'clearing of the FP display' could be done in different states on different ways: Bit 1-2 describing when the data is cleared and Bit 3-7 describing which display fields must be cleared: Bits 2,1:= 00 -> clear display in state STARTED = 01 -> clear the display in state IDLE (transaction data stored) = 10 -> clear display in state AUTHORIZE or STARTED Bit 3: = 0 -> clear Volume display (set to 0) = 1 -> don't clear Volume display Bit 4: = 0 -> clear Amount display (set to 0) = 1 -> don't clear Amount display Bit 5: = 0 -> clear Unit Price display (set to 0) = 1 -> don't clear Unit Price display Bit 6: = 0 -> clear Product name display (nothing displayed) = 1 -> don't clear Product display Bit 7: = 0 -> clear Fuelling Mode display (nothing displayed) = 1 -> don't clear Fuelling Mode display Please note that the CD can reset the FP display via the FP DB Dat_Id 66 (*Clear_Display*) Command. Please note that dispensers that do not permit this data-Id to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set this Data_Id to the hardcoded *Clear_Display_Mode* value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value.	Bin8	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
11 (0BH)	***Auth_State_Mode*** Specifies if the calculator FPs may operate with a pre-authorization (dictates if the FP state AUTHORISED state may be entered). 0 = AUTHORISED state allowed 1 = AUTHORISED state not allowed If Auth_State_Mode is set to 01, this means the Authorised state is not allowed. If a Release_FP command is received in the Idle state the Data_Ack returned should be 06 (command not accepted).	bin8 (0-1)	R(1-9) W(1-2)	M
12 (0CH)	***Stand_Alone_Auth*** Specifies how the dispenser shall work in 'stand alone' mode. 0 = transaction starts by "Nozzle-Up" 1 = manual FP release by a separate key	bin8 (0-1)	R(1-9) W(1-2)	M
13 (0DH)	***Max_Auth_Time*** Specifies the maximum amount of time (in 10 second units) that the FP will stay in the AUTHORISED state. 0 = authorization time is unlimited Please note that a write can occur to this Data_Id in any state. However, the new value will only become active when the FP next goes into states 1 to 5. *PCD Comment:* *Where the proprietary pump connected to the PCD can not support this timer function directly the PCD will have to implement its own watchdog timer to recognise when the timer has expired and then carry out the required actions.*	Bin8 (0-255)	R(1-9) W(1-9)	M

LIMIT DATA

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
21 (15H)	***Max_Time_W/O_Prog*** Specifies the maximum time in seconds allowed between pulses. If the time is exceeded the calculator must stop the FP motors. 0 = no check Please note that dispensers that do not permit the *Max_Time_W/O_Prog* to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 0 Must set the *Max_Time_W/O_Prog* to the hardcoded maximum time without progress value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *Where the proprietary pump connected to the PCD can not support this timer function directly the PCD will have to implement its own watchdog timer to recognise when the timer has expired and then carry out the required actions. If the proprietary pump protocol doesn't differentiate between the **STARTED** pump state and a **FUELLING** pump state the PCD will not be able to recognize if a time out has occurred and hence will not be able to activate a stop command to the pump. In this case the PCD will not be able to support this functionality. However the PCD should still allow the CD to read and write this Data_Id as if the functionality were supported.*	Bin8 (0-255)	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
22 (16H)	***Min_Fuelling_Vol*** Specifies the minimum volume in millilitres required before the transaction can be considered as having 'started' (the FP status should change from **STARTED** to **FUELLING**). 0 = the FP state moves directly to **FUELLING** Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *Where the proprietary pump connected to the PCD can not support this volume limit directly the PCD will have to try to implement its own mechanism to recognise when the volume limit has been exceeded and change the states appropriately. If the PCD can find no mechanism to support this feature then it should treat this Data_Id as having a hardcoded and unchangeable value of 0 (See above for more explanation).*	Bin8 (0-255)	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
23 (17H)	***Min_Display_Vol*** Specifies at what volume in millilitres the FP starts to display the transaction data. 0 = no 'volume delay' to start a transaction Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *Where the proprietary pump connected to the PCD can not support this volume limit directly the PCD will have to try to implement its own mechanism to recognise when the volume limit has been exceeded and clear the display appropriately. If the PCD can find no mechanism to support this feature then it should treat this Data_Id as having a hardcoded and unchangeable value. (See above for more explanation).*	Bin8 (0-255)	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
24 (18H)	***Min_Guard_Time*** Specifies the minimum time in seconds between two transactions. 0 = no limitation Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *Where the proprietary pump connected to the PCD can not support this timer directly the PCD will have to try to implement its own mechanism to recognise when the timer has expired and allow or restrict the next transaction respectively. If the PCD can find no mechanism to support this feature then it should treat this Data_Id as having a hardcoded and unchangeable value. (See above for more explanation).*	Bin8 (0-255)	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
26 (1AH)	***Pulser_Err_Tolerance*** Specifies the maximum number of error pulses allowed in one transaction. 0 = no error pulses allowed Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *Where the proprietary pump connected to the PCD can not support this pulser error handling mechanism directly the PCD will have to treat this Data_Id as having a hardcoded and unchangeable value. (See above for more explanation).*	Bin8 (0-255)	R(1-9) W(1-2)	M
28 (1CH)	***Time_Display_Product_Name*** Time in second to display the product name on the Volume/Amount displays. 0 = no product displayed on the volume/amount display Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-255)	R(1-9) W(1-2)	O

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
DISPLAY AND ROUNDING CONFIGURATION				
40 (28H)	***Digits_Vol_Layout*** Configure displayed layout of the Volume field. LNIB = volume display field length HNIB = decimal point position left justified Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Handling:* *Where the proprietary pump connected to the PCD can not support this display handling mechanism directly the PCD will have to treat this Data_Id as having a hardcoded and unchangeable value. (See above for more explanation).*	Bcd2	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
41 (29H)	***Digits_Amount_Layout*** Configure displayed layout of the Amount field. LNIB = amount display field length HNIB = decimal point position left justified Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Handling:* *Where the proprietary pump connected to the PCD can not support this display handling mechanism directly the PCD will have to treat this Data_Id as having a hardcoded and unchangeable value. (See above for more explanation).*	Bcd2	R(1-9) W(1-2)	M
42 (2AH)	***Digits_Unit_Price*** Configure displayed layout of the Unit Price field. LNIB = unit price display field length HNIB = decimal point position left justified Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Handling:* *Where the proprietary pump connected to the PCD can not support this display handling mechanism directly the PCD will have to treat this Data_Id as having a hardcoded and unchangeable value. (See above for more explanation).*	Bcd2	R(1-9) W(1-2)	M

CALCULATOR DATABASE

DB_Ad = C_DAT (01H)

Data _Id	***Data Element Name*** Description	Field Type (Values)	Read/Write in State	M/O
43 (2BH)	***Unit_Price_Mult_Fact*** Specifies the multiplication factor (ten to the power of x = 10^{x}) between the displayed Unit Price value and the Unit_Price field. The range of the field is: +/-, 0-9. Bit8: = 0 -> positive = 1 -> negative bit4-1: = 0 – 9 If the basic country currency is in pounds, and the unit price is in pence, there is a clear ratio of 100 to one i.e. 10^{2}. In this case the Unit_Price_Mult_Fact would be 02H. The Unit_Price_Mult_Fact is always with respect to the countries basic currency. Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Handling:* *Where the proprietary pump connected to the PCD can not support this display handling mechanism directly the PCD will have to treat this Data_Id as having a hardcoded and unchangeable value. (See above for more explanation).*	Bin8	R(1-9) W(1-2)	M

44 (2CH)	***Amount_Rounding_Type*** Specifies the rounding process applied to the amount field. The rounding is done on the last 4 digits. Rounding is done to the closest value to minimize the error between the real value and the rounded value. If the difference to the high value and the low value is the same then the value should be rounded up. Rounding must be done to the closest value to minimise the error between the real value and the rounded value. The rounded value should be a multiple of the Amount_Rounding_Type (ART). An ART of 0000 is not allowed. Default ART value is 0001. Examples: Amount = 1234.56, ART=0001, Result = 1234.56 '0001' is the default value and as the last four digits '3456' are a multiplication of 1, no rounding up/down occurs. Amount = 0123.45, ART=0002, Result = 0123.46 The last four digits '2345' must be rounded up/down by two. As there are 2 numbers the same distance apart which are a multiple of 2, '2344' and '2346', the original value is rounded up. Amount = 1234.56, ART=0005, Result = 1234.55 As the nearest (multiple of 5) value to the original '3456' is '3455', in this case the original value is rounded down. Amount = 1234.56. ART=0010, Result = 1234.60 As the nearest (multiple of 10) value to the original '3456' is '3460', in this case the original value is rounded up. Please note that dispensers that do not permit this Data_Id to be changed remotely should: • Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). • Must set the *Data_Id* to the hard-coded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Handling:* *Where the proprietary pump connected to the PCD can not support this rounding mechanism directly the PCD will have to treat this Data_Id as having a hardcoded and unchangeable value. (See above for more explanation).*	Bcd4	R(1-9) W(1-2)	M

45 (2DH)	***Preset_Rounding_Amount*** When a value (Amount/Volume) is preset, the final delivery value is rounded to the preset one if the difference between the reached number of pulse and the preset number of pulse is lower that this rounding amount. If not, no rounding is done. 2 values are indicated because this difference may be negative (delivery lower than the preset value) or positive (delivery higher). To authorize the rounding: if the difference is negative its value must be lower than the lnib byte; if the difference is positive, its value must be lower than the hnib byte. Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Handling:* *Where the proprietary pump connected to the PCD can not support this rounding mechanism directly the PCD will have to treat this Data_Id as having a hardcoded and unchangeable value. (See above for more explanation).*	Bcd2	R(1-9) W(1-2)	M

46 (2EH)	***Price_Set_Nb*** This Data_Id is used as a reference number for the unit price details currently configured in the dispenser. It allows the control devices to interrogate the dispenser and establish if a new set of prices have been downloaded by another control device. This feature is useful when operating in an environment where more than one control device is connected to the network and only one of them is responsible for downloading unit prices. Please note that dispensers that do not permit this Data_Id to be changed remotely should: 0 Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). 1 Must set the *Data_Id* to the hardcoded value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bcd4 (0-9999)	R(1-9) W(1-9)	O
IDENTIFICATION DATA				
50 (32H)	***Manufacturer_Id*** To allow the CD to interrogate the manufacturer identity. *PCD Comment:* *The PCD should set this Data_Id to that of the proprietary dispenser manufacturer's Id being controlled.*	Asc3	R(1-9)	M
51 (33H)	***Dispenser_Model*** To allow the CD to interrogate the dispenser model. *PCD Comment:* *The PCD should set this Data_Id to that of the proprietary dispenser model Id being controlled.*	Asc3	R(1-9)	M
52 (34H)	***Calculator_Type*** To allow the CD to interrogate the calculator type. *PCD Comment:* *The PCD should set this Data_Id to reflect the type of proprietary dispenser's calculator being controlled.*	Asc3	R(1-9)	M

53 (35H)	***Calculator_Serial_No*** To allow the CD to interrogate the calculator's serial number. *PCD Comment:* *The PCD should set this Data_Id to reflect the connected proprietary calculator's serial number. Where the PCD can not determine the calculator's serial number it should leave this Data_Id set to spaces.*	Asc12	R(1-9)	M
54 (36H)	***Appl_Software_Ver*** To allow the CD to interrogate the version number of the application software. The Appl_Software_Ver number format is '9999999999.99'. *PCD Comment:* *The PCD should set this Data_Id's 12 ASCII bytes to supply the following data structure:* *"PCDAAABBBBBB"* *where: "PCD" is fixed as ASCII "PCD" and used to indicate that the pumps are controlled via a PCD)* *"AAA" is the 3-character code indicating the PCD manufacturer's Id.* *"BBBBBB" is the software version number of the software running in the PCD.*	asc12	R(1-9)	M
55 (37H)	***W&M_Software_Ver*** To allow the CD to interrogate the version number of the software routines related to the direct control of the dispensing of the fuel and are of interest to W&M. The W&M_Software_Ver number format is '9999999999.99'. *PCD Comment:* *As this Data_Id is a unique Data_Id to the IFSF Dispenser application protocol it is very unlikely that a proprietary pump protocol will support this feature. Hence, any W&M algorithms will be implemented in the PCD and not the pump. So this Data_Id will have to be set by the PCD to reflect the version of the W&M software it is utilising internally.*	bcd12	R(1-9)	M
56 (38H)	***W&M_Software_Date*** To allow the CD to interrogate the date of the approval of the W&M software. *PCD Comment:* *As this Data_Id is a unique Data_Id to the IFSF Dispenser application protocol it is very unlikely that a proprietary pump protocol will support this feature. Hence, any W&M algorithms will be implemented in the PCD and not the pump. So this Data_Id will have to set by the PCD to reflect the creation date of the W&M software it is utilising internally.*	Date	R(1-9)	M

57 (39H)	***W&M_Security_Type*** To allow the CD to specify the type of W&M security method used in the transaction data. 0 = no security type. 1 = security type as defined in the IFSF Bulletin: Dispenser CRC Signature Generation. *PCD Comment:* *As this Data_Id is a unique Data_Id to the IFSF Dispenser application protocol it is very unlikely that a proprietary pump protocol will support this feature. Hence, any W&M algorithms will be implemented in the PCD and not the pump. So this Data_Id will have to set by the PCD to reflect the W&M security type it is utilising internally.*	Bin8 (0-255)	R(1-9)	M
58 (3AH)	***Protocol_Ver*** To allow the CD to interrogate the version number of the protocol being used by the dispenser. The Protocol_Ver number format is '9999999999.99'. *PCD Comment:* *This Data_Id should be set to the IFSF Dispenser Application protocol version number being used by the PCD to communicate with the Site Controller/CD.*	bcd12	R(1-9)	M
59 (3BH)	***SW_Change_Date*** To allow the CD to interrogate the date of the installation of the currently installed software. *PCD Comment:* *This Data_Id should be set to the date when the PCD's software was last changed.*	Date	R(1-9) W(1-2)	M
60 (3CH)	***SW_Change_Personal_Nb*** To allow the CD to interrogate the personal id of the person who installed the current software. The field format is ooooppppppppp. Where: oooo = 4 digit Organisation number ppppppppp = 10 digits personal number. *PCD Comment:* *This Data_Id should be set to the Id or the service engineer who last changed the PCD's software.*	Bcd14	R(1-9) W(1-2)	M

61 (3DH)	***SW_Checksum*** To allow the CD to interrogate the checksum of the software. The field format is HHHH. Where: HHHH consists of four hexadecimal digits (ASCII 0-9,A-F) Please note that dispensers that do not permit the *SW_Checksum* to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). This is the preferred solution. *PCD Comment:* *This Data_Id should be set to the Software checksum of the PCD's software.*	Asc4	R(1-9) W(1-2)	M
ILLUMINATION CONTROL DATA				
70 (46H)	***Calc_Illumination*** To allow the CD to switch the dispenser's illumination: 0 = light off 1 = light on Please note that when the Calculator does not have the ability to control a light in the Dispenser then this Data_Id must still be supported at the read & write level (Obviously, the Calculator will not be able to actually control a light). *PCD Comment:* *Where the proprietary pump protocol or the pump itself does not allow the display light to be turned on remotely the PCD should handle the situation as described in the previous paragraph.*	Bin8 (0-1)	R(1-9) W(1-9)	M
71 (47H)	***LCD_Backlight_Switch*** To allow to switch the LCD back light: 0 = light off 1 = light on *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-1)	R(1-9) W(1-9)	O

72 (48H)	***Display_Intensity*** To allow to switch the intensity of the display: 0 = normal intensity 1 = high intensity *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-1)	R(1-9) W(1-9)	O
W&M TRANSACTION SECURITY				
80 (50H)	***W&M_Polynomial*** To allow the CD to configure the Polynimial used by the dispenser to calculate the W&M security checksum. Please note that in implementations where the *W&M_Polynomial* may not be changed. The Write should be rejected with a Data_ACK of 2 (Not Writable). Please note that this Data_Id is a write only field. Any attempt to read it must result in the answer message being returned with the Data_Id's Data_Lg set to zero (i.e. 80,00). *PCD Comment:* *The PCD is likely to be solely responsible for the generation of the W&M checksum so where the W&M functionality is required the PCD will have to implement it as described above. The acceptability of a PCD between the Site Controller/CD in this environment will have to be checked with the body responsible for giving the W&M approval.*	Bin16 (0-65535)	W(1-9)	M

81 (51H)	***W&M_Seed*** To allow the CD to configure the seed used by the dispenser to calculate the W&M security checksum. Please note that in implementations where the *W&M_Seed* may not be changed. The Write should be rejected with a Data_ACK of 2 (Not writable). Please note that this Data_Id is a write only field. Any attempt to read it must result in the answer message being returned with the Data_Id's Data_Lg set to zero (i.e. 80,00). *PCD Comment:* *The PCD is likely to be solely responsible for the generation of the W&M checksum so where the W&M functionality is required the PCD will have to implement it as described above. The acceptability of a PCD between the Site Controller/CD in this environment will have to be checked with the body responsible for giving the W&M approval.*	Bin16 (0-65535)	W(1-9)	M

MANUFACTURER / OIL COMPANY SPECIFIC				
200 to 255	Free to the manufacturer / oil company			

3.4 Meter Data

This data allows the CD to configure a meter in the calculator.
The access to the meter data is done by the database address M_ID (meter identification).
The M_ID = 80H is used to ask for all meters.

Please note that to allow dispensers to operate in 'stand alone' mode, the dispenser must have default settings for some of the Data_Id's contained in this database, *i.e.* the dispenser must configure these Data_Id's itself after a master reset/cold start.

METER DATABASE

DB_Ad = M_ID (81H-90H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
CONFIGURATION				
1 (01H)	***Meter_Type*** Specifies the meter type: 0 = not configured If this Data_Id has not been configured, then the dispenser should use its default value. 1 = normal speed 2 = high speed *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-2)	R(1-9) W(1-2)	O

METER DATABASE

DB_Ad = M_ID (81H-90H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
2 (02H)	***Meter_Puls_Vol_Fact*** Specifies the volume in tenth of millilitre of each pulse generated by the pulser connected to the meter. 0 = not configured. If this Data_Id has not been configured, then the dispenser should use its default value. Please note that dispensers that do not permit this Data_Id to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Data_Id* to the hardcoded default value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *It is unlikely that a proprietary pump protocol will allow the remote changing of this Data_Id. If it doesn't, then the PCD should not permit this Data_Id to be changed by the Site Controller/CD and should respond as described in the previous paragraph.*	Bin8 (1-255)	R(1-9) W(1-2)	M
3 (03H)	***Meter_Calib_Fact*** Specifies the meter calibration factor used by intelligent pulsers and piston meters. The field format is 0.000 to 9.999 (the decimal point is implied and not transmitted). *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bcd4	R(1-9) W(1-2)	O

METER DATABASE

DB_Ad = M_ID (81H-90H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
4 (04H)	***PR_Id*** Identifier of the product measured by this meter. The PR_Id (value 1-8) specifies the product which is stored in the Product Database PR_ID (address 41H-48H). 0 = no product assigned 1 = product in Product Database with address 41H 2 = product in Product Database with address 42H . . 8 = product in Product Database with address 48H Please note that dispensers that do not permit this Data_Id to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Data_Id* to the hardcoded default value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *It's important that the PCD supports this configuration Data_Id as it will be critical in calculating totals where the proprietary pump protocol can not supply the totals directly.*	Bin8 (1-8)	R(1-9) W(1-2)	M
TOTAL				
20 (14H)	***Meter_Total*** Total for the single pulse meter. The total is permanently updated during the fuelling transaction. *PCD Comment:* *Some proprietary protocols will not allow the meter totals to be read remotely. If this is the case as soon as a transaction data is received the PCD will have to calculate this total itself. Obviously, it may not be possible for the PCD to update the meter totals during the fuelling transaction. The PCD should also consider that it is advisable that these totals are stored securely and with some type of memory backup.*	Long_ Volume	R(1-9)	M

METER DATABASE

DB_Ad = M_ID (81H-90H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
MANUFACTURER / OIL COMPANY SPECIFIC				
200 to 255	Free to the manufacturer / oil company			

3.5 Product Data

This data allows the CD to specify the product data in the calculator. Per Calculator up to 8 different *Prod_Nb* could be defined.
The access to this data is defined by the PR_ID address (product identifier). This address key is used for internal links between databases (product, logical nozzle, meters). These links depend on the way the dispenser is built and are dispenser model dependent.
The PR_ID = 40H is used to ask for all products.

Any attempt to operate on a DB_Ad which has not been implemented should be rejected with a MS_ACK set to NAK 6 (Message refused, unknown database address).

PRODUCT DATABASE

DB_Ad = PR_ID (41H-48H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
CONFIGURATION				
2 (02H)	***Prod_Nb*** The *Prod_Nb* is assign by the CD during the system configuration and may be used to send product parameters (names, prices) by equipment or programs which don't need to have the knowledge of each dispenser configuration. The *Prod_Nb* must be unique for a dispenser (this is controlled by the dispenser before accepting the Prod_Nb to PR_Id link during the configuration). Therefore the product database should be declared to have only one record per unique product. A write action for a address PR_ID with the *Prod_Nb* 00000000 means that the associated data must be deleted. *PCD Comment:* *It's important that the PCD supports this configuration Data_Id as it will be critical in calculating totals where the proprietary pump protocol can not supply the totals directly.*	Bcd8	R(1-9) W(1-2)	M
3 (03H)	***Prod_Description*** Specifies the product description for the product. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Asc16	R(1-9) W(1-2)	O
VAPOUR RECOVERY				

PRODUCT DATABASE

DB_Ad = PR_ID (41H-48H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
10 (0AH)	***Vap_Recover_Const*** Specifies the Vapour Recovery constant. The Vapour_Recovery record allows the CD to configure the vapour recovery factor for each product. Currently the exact specification for vapour recovery is not known. Therefore the Vapour_Recovery message and data elements are optional and do not have to be implemented. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-255)	R(1-9) W(1-2)	O

MANUFACTURER / OIL COMPANY SPECIFIC				
200 to 255	Free to the manufacturer / oil company			

3.6 Product Data per Fuelling Mode

This data allows the CD to configure the product parameter per fuelling mode.
The access to the product fuelling mode data is done by the database address PR_DAT (product data) + Prod_Nb (Product Number) + FM_ID (fuelling mode identifier).
The FM_ID = 10H is used to ask for all fuelling modes at a product.

Please note that to allow dispensers to operate in 'stand alone' mode, the dispenser must have default settings for some of the Data_Id's contained in this database. I.e. the dispenser must configure these Data_Id's itself after a master reset/cold start.

PRODUCT PER FUELLING MODE DATABASE

DB_Ad = PR_DAT (61H) + Prod_Nb (00000001-99999999) + FM_ID (11H-18H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
CONFIGURATION				
1 (01H)	***Fuelling_Mode_Name*** Specifies the Fuelling mode name. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Asc8	R(1-9) W(1-2)	O
2 (02H)	***Prod_Price*** Specifies the product/fuelling mode's Unit Price. Please note that a write can occur to this Data_Id in any state. However, the new value will only become active when the FP next goes into states 1 to 5. *PCD Comment:* *The PCD will have to take care that the new price is sent to the proprietary pump as soon as it can be accomplished.*	Unit_ Price	R(1-9) W(1-9)	M

PRODUCT PER FUELLING MODE DATABASE

DB_Ad = PR_DAT (61H) + Prod_Nb (00000001-99999999) + FM_ID (11H-18H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
3 (03H)	***Max_Vol*** Specifies the product/fuelling mode's maximum volume allowed. 0 = no maximum limit for volume If the Max_Vol limit is reached the transaction should go to closed. *PCD Comment:* *Some proprietary pump protocols don't allow the maximum volume to be changed. In this case the PCD will have to accept a write of this Data_Id and must store the value just in case a SC/CD tries to read the value. It is also worth mentioning that this maximum volume must also be considered with the other preset volumes possible and that the PCD where possible always makes sure that the proprietary pump never fills above the lowest of the volume limits.*	Volume	R(1-9) W(1-9)	M
4 (04H)	***Max_Fill_Time*** Specifies the product/fuelling mode's maximum fuelling time (in 10 second units) allowed. 0 = fuelling time is unlimited Please note that a write can occur to this Data_Id in any state. However, the new value will only become active when the FP next goes into states 1 to 5. Please note that dispensers that do not permit this Data_Id to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Data_Id* to the hardcoded default value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *As many proprietary pump protocols won't allow a maximum timer to be set remotely the PCD will have to have its own watch dog timer to time the filling time. If this timer expires the PCD will have to stop the filling. Alternatively the PCD can view this Data_Id as not writable and handle the situation according to the paragraph above.*	Bin8 (0-255)	R(1-9) W(1-9)	M

PRODUCT PER FUELLING MODE DATABASE

DB_Ad = PR_DAT (61H) + Prod_Nb (00000001-99999999) + FM_ID (11H-18H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
5 (05H)	***Max_Auth_Time*** (see note 1) Specifies the maximum amount of time (in 10 second units) that the FP will stay in the AUTHORISED state. 0 = authorisation time is unlimited Please note that a write can occur to this Data_Id in any state. However, the new value will only become active when the FP next goes into states 1 to 5. Note 1. The Max_Auth_Time Data Id 5 in the Product Per Fuelling mode data base in only in this data base to ensure backwards compatibility. To determine the time a FP should stay in the authorised state, Data Id 13 calculator data base should be used. *PCD Comment:* *As many proprietary pump protocols won't allow a maximum authorisation timer to be set remotely the PCD will have to have its own watch dog timer to time the filling time. If this timer expires the PCD will have to stop the filling.*	Bin (0-255)	R(1-9) W(1-9)	M
6 (06H)	***User_Max_Volume*** Specifies the Max Volume the FP will fuel before it will move into the suspend fuelling state. *PCD Comment:* *As many proprietary pump protocols won't allow a user maximum volume to be set remotely, the PCD will have to accept a write of this Data_Id and must store the value just in case a SC/CD tries to read the value. It is also worth mentioning that this user maximum volume must also be considered with the other preset volumes possible and that the PCD where possible always makes sure that the proprietary pump never fills above the lowest of the volume limits.*	Volume	R(1-9) W(1-9)	M
MANUFACTURER / OIL COMPANY SPECIFIC				
200 to 255	Free to the manufacturer / oil company			

3.7 Fuelling Point Data

This data allows the CD to configure and control a FP in the dispenser.
The access to the fuelling point data is done by the database address FP_ID (fuelling point identification). The FP_ID = 20H is used to ask for all fuelling points.

Please note that to allow dispensers to operate in 'stand alone' mode, the dispenser must have default settings for some of the Data_Ids contained in this database. I.e. the dispenser must configure these Data_Ids itself after a master reset/cold start.

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
CONFIGURATION				
1 (01H)	***FP_Name*** Allows to associate a name or number to the Fuelling Point. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Asc8	R(1-9) W(1-2)	O
2 (02H)	***Nb_Tran_Buffer_Not_Paid*** Specifies the number of non paid transactions (not cleared by the CD) that may be stored by each FP. The acceptable range is 1 to 15. If a write action occurs to this Data_Id with a value greater than can be supported by the dispenser, the dispenser should reject the message with a Data_ACK value of 1 (Invalid value (too big/small)). Please note that dispensers that do not permit this Data_Id to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). - Must set the *Data_Id* to the hard coded default value. *PCD Comment:* *It is desirable that the PCD supports as many transaction buffers as is allowed by the local W&M and can be stored in the device.*	Bin8 (1-15)	R(1-9) W(1-2)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
3 (03H)	***Nb_Of_Historic_Trans*** Specifies the number of cleared transactions that can be stored in the FP. Always the latest transactions are available (first in, first out). If a write action occurs to this Data_Id with a value greater than can be supported by the dispenser, the dispenser should reject the message with a Data_ACK value of 1 (Invalid value (too big/small)). Please note that dispensers that do not permit this Data_Id to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Data_Id* to the hard coded default value. *PCD Comment:* *It is desirable that the PCD supports as many historic transaction buffers as is allowed by the local W&M and can be stored in the device.*	Bin8 (1-15)	R(1-9) W(1-2)	M
4 (04H)	***Nb_Logical_Nozzle*** Number of logical nozzle on the FP. The acceptable range is 1 to 8. 0 = not configured If a write action occurs to this Data_Id with a value greater than can be supported by the dispenser, the dispenser should reject the message with a Data_ACK value of 1 (Invalid value (too big/small)). Please note that dispensers that do not permit this Data_Id to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). ■ Must set the *Data_Id* to the hard coded default value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (1-8)	R(1-9) W(1-2)	O

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
6 (06H)	***Loudspeaker_Switch*** To allow the fuelling point's loudspeaker to be switch on and off. 0 = off, 1 = on *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-1)	R(1-9) W(1-9)	O
7 (07H)	***Default_Fuelling_Mode*** The FM for the next fuelling transaction can be changed by the data element Fuelling_Mode (Data_Id 33). The Fuelling_Mode is set to the Default_Fuelling_Mode after the current transaction is stored in the transaction buffer. The acceptable range for the FM is 1 to 8. 0 = not configured *PCD Comment:* *As most proprietary pump protocols do not have the concept of fuelling modes the PCD will have to manage this issue. The most important things to consider are the correct prices & limits are sent to the proprietary pump whenever the FM is changed.*	Bin8 (1-8)	R(1-9) W(1-2)	M
8 (08H)	***Leak_Log_Noz_Mask*** To allow the CD to perform a leak test for the predefined logical nozzle: bit 1 = LogicalNozzle1Flag [Logical Nozzle 1] (LN_ID = 1) bit 2 = LogicalNozzle2Flag [Logical Nozzle 2] (LN_ID = 2) " " " " bit 8 = LogicalNozzle8Flag [Logical Nozzle 8] (LN_ID = 8) (The numbering of the logical nozzles is manufacturer specific and must be defined separately.) 1= Perform leak test. 0 = Do not perform leak test. All nozzles must be reset to 'perform leak test' after the current transaction has been stored in the transaction buffer. *PCD Comment:* *If the proprietary pump protocols doesn't support masking of the nozzles the PCD will have to reject any attempt to write a value other than 255/FFH with a MS_ACK=5 and a Data_ACK=2.*	Bin8	W(3)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
LIGHT CONTROL DATA				
10 (0AH)	***Drive_Off_Light_Switch*** To allow to switch the 'drive off light' when the Drive_Off_Light_Mode (Data_Id 8 in Calculator Database) is in external control mode: 0 = light off 1 = light on *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-1)	R(1-9) W(2-9)	O
11 (0BH)	***OPT_Light_Switch*** To allow to switch up to four 'OPT_Lights' when the OPT_Light_Mode (Data_Id 9 in Calculator Database) is in external control each light is controlled by a pair of adjoining bits were: bits 0-1 = Light1 [light 1] bits 2-3 = Light2 [light 2] bits 4-5 = Light3 [light 3] bits 6-7 = Light4 [light 4] and the bit values to set the light state are 00 = light off 01 = light on 10 = SLOW BLINK 11 = FAST BLINK The time for Slow and Fast blinking is dependent on the technology of the light. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-255)	R(1-9) W(2-9)	O

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
CONTROL DATA				
20 (14H)	***FP_State*** Used to indicate the state of the FP. Please see the Fuelling Point State Diagram for details of the individual states (chapter 2.1 of this document). An unsolicited message (Data_Id 100) is generated by the FP for each change in the FP state. *PCD Comment:* *Please see the earlier sections dealing with the state behaviours related to the PCD.*	Bin8 (1-9)	R(1-9)	M
21 (15H)	***Log_Noz_State*** To allow to read the state of all logical nozzles. Bit 1 = LogicalNozzle1Flag [Logical nozzle 1] (LN_ID = 1) bit 2 = LogicalNozzle2Flag [Logical nozzle 2] (LN_ID = 2) .. bit 8 = LogicalNozzle8Flag [Logical nozzle]8 (LN_ID = 8) The numbering of the nozzles is manufacturer model specific and must be defined separate. 0 = Nozzle not removed 1 = Nozzle removed An unsolicited message (Data_Id 100) is generated by the FP for each change in the Logical Nozzle State. *PCD Comment:* *Where the proprietary pump protocol indicates which nozzle has been removed and hence allows the logical nozzle details to be passed on to the SC/CD the PCD should generate the respective correct unsolicited messages. If the proprietary pump protocol doesn't indicate the nozzle and the PCD can't establish which one it is by cross-referencing the grade Id or the unit price. Then it should set the Log_Noz_State to a default of 0 when no nozzles have been lifted and to 255/FFH when a nozzle has been lifted.*	Bin8	R(1-9)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
22 (16H)	***Assign_Contr_Id*** Used to indicate if and to whom the FP has been assigned. Only a release command coming from this controller is accepted. Note that it is possible to clear a transaction from any CD and not just the CD that released the FP. A Logical Node Address (LNA) is used for the Assign_Contr_Id. The LNA is specified by 2 bytes (S = Subnet, N = Node). For details see document "Part II.1, Communication Specification". 0,0 = not assigned, X,Y = Controller device that assigned the FP (X = S, Y = N), 255,255 = FP running in stand alone mode. A new assignment can only be received by a FP after a reset (not assigned, i.e. 0,0 is written) by the device that previously assigned the FP. In cases where the CD that assigned the FP has 'crashed' and is off-line the assignment can be cleared by another CD. This is achieved by setting the *Assign_Contr_Id* to the same as the Dispenser's own application Subnet & Node. The Dispenser then resets the *Assign_Contr_Id* to 0,0. If a CD releases a FP that it previously assigned the resulting transaction must be stored and immediately flagged as being 'Locked' by the CD that assigned it. **Please note that when a FP is assigned to a CD all write actions (data or commands) to the respective FP's Fuelling Point, Calculator, Product, Product per Fuelling Mode, Meter & Logical Nozzle Data Bases by other CDs must be rejected with a Data_ACK 6 (command not accepted or Data_Ack 2 data not writable). Communications databases can not be locked by a controlling device. Only databases related to a fuelling point can be locked by a device wishing to control that fuelling point. Common databases should not be changed without serious consideration.** An unsolicited message (Data_Id 100) is generated by the FP for each change in the FP's assignment. *PCD Comment:* *As the assignment concept is generally not supported in proprietary pump protocols the PCD will have to manage this assignment handling locally.*	Bin16	R(1-9) W(2-4)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
23 (17H)	***Release_Mode*** To allow to configure the release mode. 0 = a "release message" must be received from a CD to authorise any transaction 1 = the FP may authorise transactions as long as a free transaction buffer is available *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-1)	R(1-9) W(2-4)	O
24 (18H)	***ZeroTR_Mode*** Specifies if a transaction with a zero value (the displayed volume and the displayed amount are zero) must be stored in the transaction buffer. 0 = zero transaction must not be stored 1 = zero transaction must be stored The ZeroTR_Mode is set to 0 (transaction must not be stored) after the current fuelling transaction is stored in the transaction buffer. *PCD Comment:* *As the zero transaction handling is generally not supported in proprietary pump protocols the PCD will have to manage this handling locally.*	Bin8 (0-1)	R(1-9) W(2-4)	M
25 (19H)	***Log_Noz_Mask*** To allow the CD to authorize one or many logical nozzle(s): bit 1 = LogicalNozzle1Flag [Logical nozzle 1] (LN_ID = 1) bit 2 = LogicalNozzle2Flag [Logical nozzle 2] (LN_ID = 2) . bit 8 = LogicalNozzle8Flag [Logical nozzle 8] (LN_ID = 8) The numbering of the nozzles is manufacturer model specific and must be defined separate. 1 = Nozzle authorized 0 = Nozzle not authorized All nozzles must be authorised in the Log_Noz_Mask after the current fuelling transaction is stored in the transaction buffer. *PCD Comment:* *If the proprietary pump protocols doesn't support masking of the nozzles the PCD will have to reject any attempt to write a value with a MS_ACK=5 and a Data_ACK 2.*	Bin8	R(1-9) W(2-4)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
26 (1AH)	***Config_Lock*** Used to lock the communications of a dispenser to one controlling device while the dispenser is being configured. X,Y = Controller Device that locked the FP (X = Subnet, Y = Node) If the controlling device fails after being locked, a time out is applied. A new assignment can only be received by a CD after a reset (not assigned, i.e. 0,0 is written) by the device that previously assigned the FP. In cases where the CD that assigned the FP has 'crashed' and is off-line the assignment can be cancelled by another CD. This is achieved by setting the config_lock to the same as the application Subnet & Node. The Dispenser should then reset the config_lock to 0,0. MS_ACK 9 (configuration lock error) is sent in responses to other devices attempting to communicate with the dispenser during configuration. *PCD Comment:* *This action involves the SC/CD communicating directly with the PCD so the functionality has to be implemented as given.*	Bin 16	R(1,2) W(1,2)	M
27 (1BH)	***Remote_Amount_Prepay*** Specifies the money amount prepay limit for the potential pending transaction. 0 = no prepay Remark: If the Remote_Amount_Prepay and the Remote_Volume_Preset are used, the more restrictive is used. The prepay value is set to zero by the dispenser calculator when the current fuelling transaction is stored in the transaction buffer. *PCD Comment:* *If the proprietary pump protocols doesn't support any prepayment the PCD will have to reject any attempt to write a value with a MS_ACK=5 and a Data_ACK 2.*	Amount	R(1-9) W(3-4)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
28 (1CH)	***Remote_Volume_Preset*** Specifies the volume preset limit for the potential pending transaction. 0 = no preset Remark: If the Remote_Amount_Prepay and the Remote_Volume_Preset are used, the more restrictive is used. The preset value is set to zero by the calculator when the transaction is finished. *PCD Comment:* *If the proprietary pump protocol doesn't support volume prepayment but does support an amount preset the PCD will have to try and find a mechanism to over come this shortcoming. A method may be to calculate the corresponding amount and using an amount preset. If the proprietary pump protocols doesn't support any prepayment the PCD will have to reject any attempt to write a value with a MS_ACK=5 and a Data_ACK 2.*	Volume	R(1-9) W(3-4)	M
32 (20H)	***Release_Token*** Allows the controller device to assign a token when a transaction is started. This token is used by the controller to link a release command with the resulting transaction. *PCD Comment:* *The PCD will need to store the received release token and then attach it to the resulting transaction.*	Bin8 (0-255)	R(1-9) W(3-4)	M
33 (21H)	***Fuelling_Mode*** Fuelling mode (FM_ID) of the fuelling point. It cannot be modified when a transaction is started. The acceptable range is 1 to 8. After the current fuelling transaction is stored in the transaction buffer the FM is set to the default FM (specified in Data_Id 7). *PCD Comment:* *As most proprietary pump protocols don't support fuelling modes the PCD will have to provide this functionality internally. The main task is to make sure that the proprietary pump has the correct unit price and limits for the given fuelling mode.*	Bin8 (1-8)	R(1-9) W(3-4)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
41 (29H)	***Transaction_Sequence_Nb*** After storing the current transaction in the transaction buffer, a new sequence number is created by incrementing the previous one. *PCD Comment:* *As the transaction sequence number is unlikely to be provided by the proprietary pump. The PCD will have to maintain and update the transaction sequence number.*	Bcd4 (1-9999)	R(1-9) W(1-2)	M
CURRENT TRANSACTION DATA				
29 (1DH)	***Current_TR_Seq_Nb*** Indicate the sequence number for the running fuelling transaction. By authorising the fuelling, the sequence number is copied from Transaction_Sequence_Nb (Data_Id 41). Its value is reset to zero after storing the transaction in the transaction buffer. *PCD Comment:* *During the transaction this Data_Id must be set by the PCD to the transaction sequence number that the resulting transaction will have. It is most likely that the PCD will have to generate this transaction sequence number.*	Bcd4 (1-9999)	R(6-9)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
30 (1EH)	***Release_Contr_Id*** Specifies which Controller Device has released the FP for the running transaction. A Logical Node Address (LNA) is used for the Release_Contr_Id. The LNA is specified by 2 bytes (S = Subnet, N = Node). For details see document "Part II, Communication Specification". 0,0 = Controller Device is not specified, X,Y = Controller Device that released the FP (X = Subnet, Y = Node), 255,255 = FP running in stand alone mode. Its value is reset to zero after storing the current fuelling transaction in the transaction buffer or when the FP state changes to Idle. Please note that dispensers that do not permit the *Release_Contr_Id* to be changed remotely should: ■ Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). This is the preferred solution. *PCD Comment:* *During the transaction this Data_Id must be set by the PCD to the controller Id that released the pump.*	Bin16	R(3-9) W(3-4)	M
31 (1FH)	***Suspend_Contr_Id*** Specifies which Controller Device has suspended the running transaction. A Logical Node Address (LNA) is used for the Suspend_Contr_Id. The LNA is specified by 2 bytes (S = Subnet, N = Node). For details see document "Part II, Communication Specification". 0,0 = Controller Device not specified, X,Y = Controller Device that suspended the FP (X = Subnet, Y = Node). Its value is reset to zero after resuming the suspended transaction or after storing the current fuelling transaction in the transaction buffer. *PCD Comment:* *During the transaction this Data_Id must be set by the PCD to the controller Id that suspended the pump.*	Bin16	R(7,9) W(6,8)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
34 (22H)	***Current_Amount*** Indicates the money amount of the current fuelling transaction. Its value is reset to zero after storing the transaction in the transaction buffer. *PCD Comment:* *During the transaction this Data_Id must be set by the PCD to the current transaction amount. If the transaction amount can not be determined from the proprietary pump the PCD will have to set this value to 0. However, if the transaction volume is available the PCD should try and calculate the amount from that and the unit price (if known).*	Amount	R(6-9)	M
35 (23H)	***Current_Volume*** Indicates the volume of fuel dispensed in the current fuelling transaction. Its value is reset to zero after storing the transaction in the transaction buffer. *PCD Comment:* *During the transaction this Data_Id must be set by the PCD to the current transaction volume. If the transaction volume can not be determined from the proprietary pump the PCD will have to set this value to 0. However, if the transaction amount is available the PCD should try and calculate the volume from that and the unit price (if known).*	Volume	R(6-9)	M
36 (24H)	***Current_Unit_Price*** Indicates the unit price of the current fuelling transaction. Its value is reset to zero after storing the transaction in the transaction buffer. *PCD Comment:* *During the transaction this Data_Id must be set by the PCD to the current transaction unit price. If the transaction's unit price can not be determined from the proprietary pump the PCD will have to set this value to 0. However, if the transaction's grade is known then the unit price can be derived from the PC's internal price tables.*	Unit_ Price	R(6-9)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
37 (25H)	***Current_Log_Noz*** Indicates which logical nozzle is removed for the current fuelling transaction. Coming to state STARTED the data from Log_Noz_State (Data_Id 21 in this database) are copied into this data element. Its value is reset to zero after storing the transaction in the transaction buffer. *PCD Comment:* *Where the proprietary protocol allows the PCD to establish which logical nozzle is being used it should set this Data_Id accordingly. If not then this Data_Id must be set to 255/FFH.*	Bin8	R(6-9)	M
38 (26H)	***Current_Prod_Nb*** Selected product number for the current fuelling transaction. The Prod_Nb is defined in the Product Database (chapter 3.4). Its value is reset to zero after storing the transaction in the transaction buffer. *PCD Comment:* *Where the proprietary protocol allows the PCD to establish which product is being used it should set this Data_Id accordingly. If not then this Data_Id should be set to 0.*	Bcd8	R(6-9)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
39 (27H)	***Current_TR_Error_Code*** Indicates the error status of the transaction. If the error status = 0 then no error has occurred. If <> 0 then an error has occurred. Dependent on the error type the transaction could be treated accordingly. (Please see the FP_Error_Type in the Error Code Database). Its value is reset to zero after storing the transaction in the transaction buffer. *PCD Comment:* *The PCD will only be able to set the error code if the proprietary pump protocol supplies the error code. If this is the case then the PCD is responsible to convert the proprietary error code to the corresponding IFSF Dispenser error code. If there is an proprietary error code that can not be mapped to an IFSF Dispenser one then an error code in the Manufacturer/Oil company specific should be assigned. These new error codes must be documented.*	Bin8 (0-255)	R(6-9)	M
40 (28H)	***Current_Average_Temp*** Indicates the current temperature of the fuel being dispensed. Its value is reset to zero after storing the transaction in the transaction buffer. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Temp	R(6-9)	O
42 (2AH)	***Current_Price_Set_Nb*** Indicates the current Price_Set_Nb in use by this dispenser. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bcd 4 (0-9999)	R(6-9)	O

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
CONFIGURATION				
43 (2BH)	***Multi_Nozzle_Type*** This data ID returns the type of physical nozzle associated with a removed physical nozzle as defined in Data_Id 43. It is enclosed as follows: bit 1 = Satallite master nozzle bit 2 = Satallite slave nozzle No. 1 bit 3 = Satallite slave nozzle No. 2 bit 4 = Satallite slave nozzle No. 3 bit 5 = 2 speed standard flow nozzle bit 6 = 2 speed high flow nozzle bit 7 = Multi product through one nozzle (i.e. blender) Note these bits are valid only when the corresponding nozzle type corresponds to one of the model types listed above. If the nozzle does not correspond to one of the models listed or if no physical nozzle is removed, this Data_Id has the value 00.	Bin8	R(1-9)	O
44 (2CH)	***Multi_Nozzle_State*** These bits correspond to the physical nozzle state 0 = Nozzle not removed 1 = Nozzle removed Note this Data Id will reflect the physical state of any nozzle, not just the types listed in Data_Id 43. Any changes in the Multi_Nozzle State must result in unsolicited FP Multi Nozzle_Status_Message (Data_Id 101)	Bin8	R (1-9)	O
45 (2DH)	***Multi_Nozzle_Status_Message*** These bits correspond to the nozzle definitions in Data_Id 44. 0 = Flow disabled through respective nozzle 1 = Flow enabled through respective nozzle Where the dispenser does not allow a flow from the requested nozzle to be controlled, the write is rejected with a Data_ACK = 2 (Not Writable). The default value is despenser specific. Some despensers may require flow to be enabled through all available nozzles by default, others may restrict flow to specific nozzles by default.	Bin8	R (6-9) W (6-9)	O

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
51 (33H)	***Local_Vol_Preset*** Allows the FP to inform the CD about a change to the local volume preset. The volume preset is reset to 0 after the current fuelling transaction is stored in the transaction buffer. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Volume	R(1-9)	O
52 (34H)	***Local_Amount_Prepay*** Allows the FP to inform the CD about a change to the local amount prepay. The amount prepay is reset to 0 after the current fuelling transaction is stored in the transaction buffer. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Amount	R(1-9)	O
59 (3BH)	***Running_Transaction_Message_Frequency*** Specifies the frequency at which the running transaction is sent, in tenths of a second. 0 = not active. 1-999 = delay in tenths of a second. e.g. 20 is 2 second interval.	Bcd4 (0-999)	R(1-9) W(1-9)	O
FP CONTROL				
60 (3CH)	***Open_FP*** To open a closed FP. Please note that an Unsolicited *FP_Status_Message* (Data_Id 100) must be transmitted as a result of this command . This action must occur even if the state has not changed as a result of the command. Please note that an acknowledgement to this command implies that the *FP_State* has changed to the open state (see Chapter 5). *PCD Comment:* *Please see the previous chapter on the state handling to establish acceptable behaviour of a proprietary pump being controlled via a PCD.*	CMD	W(2)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
61 (3DH)	***Close_FP*** To close a FP. Please note that an Unsolicited *FP_Status_Message* (Data_Id 100) must be transmitted as a result of this command . This action must occur even if the state has not changed as a result of the command. *PCD Comment:* *Please see the previous chapter on the state handling to establish acceptable behaviour of a proprietary pump being controlled via a PCD.*	CMD	W(3-9)	M
62 (3EH)	***Release_FP*** Authorise or pre-authorise to start a transaction. The releasing CD identifier could be stored separately from this command in the Release_Contr_Id (Data_Id 30). A Release_FP command must be rejected with a Data ACK 6 if there is no Unit_Price available. Please note that an Unsolicited *FP_Status_Message* (Data_Id 100) must be transmitted as a result of this command . This action must occur even if the state has not changed as a result of the command. *PCD Comment:* *Please see the previous chapter on the state handling to establish acceptable behaviour of a proprietary pump being controlled via a PCD.*	CMD	W(3-4)	M
63 (3FH)	***Terminate_FP*** Terminate the running transaction. Please note that an Unsolicited *FP_Status_Message* (Data_Id 100) must be transmitted as a result of this command . This action must occur even if the state has not changed as a result of the command. *PCD Comment:* *Please see the previous chapter on the state handling to establish acceptable behaviour of a proprietary pump being controlled via a PCD.*	CMD	W(4-9)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
64 (40H)	***Suspend_FP*** Provisory stop the running delivery. The suspending CD identifier could be stored separately from this command in the Suspend_Contr_Id (Data_Id 31). Please note that an Unsolicited *FP_Status_Message* (Data_Id 100) must be transmitted as a result of this command . This action must occur even if the state has not changed as a result of the command. *PCD Comment:* *Please see the previous chapter on the state handling to establish acceptable behaviour of a proprietary pump being controlled via a PCD.*	CMD	W(6,8)	M
65 (41H)	***Resume_FP*** Restart a provisory stopped delivery. Only that CD that has suspended the transaction (the controller device identification is stored in Data_Id 31 Suspend_Contr_Id) can restart it. If the Suspend_Contr_Id is not specified (= 0) the suspended fuelling transaction can be resumed by every CD. The check (resuming CD equal suspending CD) must be done by the resuming CD. Please note that an Unsolicited *FP_Status_Message* (Data_Id 100) must be transmitted as a result of this command . This action must occur even if the state has not changed as a result of the command. *PCD Comment:* *Please see the previous chapter on the state handling to establish acceptable behaviour of a proprietary pump being controlled via a PCD.*	CMD	W(7,9)	M
66 (42H)	***Clear_Display*** When a valid *Clear_Display* command is received the FP display will be cleared according to the criteria given in the Calculator Data base's *Clear_Display_Mode* (Data_Id 10). *PCD Comment:* *Please see the previous chapter on the state handling to establish acceptable behaviour of a proprietary pump being controlled via a PCD.*	CMD	W(3-5)	M

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
67 (43H)	***Leak_Command*** To perform a leak test on the respective FP with the defined logical nozzles (see *Leak_Log_Noz_Mask*, Data_Id 8). The command is only accepted if all nozzles are returned. If the command is issued to a dispenser in the wrong device state or when a nozzle has been removed it will reject the command with a Data_ACK 3 (Command refused in that state). Please note that an Unsolicited *FP_Status_Message* (Data_Id 100) must be transmitted as a result of this command . This action must occur even if the state has not changed as a result of the command. *PCD Comment:* *When the proprietary pump protocol does not support the leak test. The PCD should reject this command with a MS_ACK=5 and Data_ACK=5.*	CMD	W(3)	M
80 (50H)	***FP_Alarm*** Used to indicate the alarm state of the FP. The Error Code Data was designed to keep a count of the number of times an error has occurred. There is also a need to know the current state of minor errors e.g. Paper Out, has a printer paper or not. It is possible for a controller device to keep a record of the current state of a minor error by monitoring all the Unsolicited messages, but if a controller device is 'Cold Started' all historical information is lost. Hence the need for an ***Alarm*** data element in a device. When read this data element gives the current state of alarms. Alarms are warnings. Alarms do not create a state change in the device, but an unsolicited (without acknowledge) message is generated by the FP for each change in the ***FP_Alarm.*** These alarms should not appear in the list of minor errors. (Bit number in decimal). Bit 1 – EVR Timer Running Bit 2 – EVR Timer Expired Bit 3 – 8 To be defined Bit 9 – EVR System Defect Bit 10 – 48 To be defined Bit 49 – 64 Manufacturer specific 0 means normal, alarm condition not present. 1 means alarm condition present.	Bin64	R(*)	O

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
UNSOLICITED DATA				
100 (64H)	***FP_Status_Message*** A FP_Status_Message must be sent unsolicited (without acknowledge) by the FP whenever a change has occurred in the status of the FP_State, Log_Noz_State or the Assign_Contr_Id or FP_Alarm (Optional), or whenever the state cannot be changed following request by the CD to change state. The FP_Status_Message includes: - FP_State (Data_Id = 20) - Log_Noz_State (Data_Id = 21) - Assign_Contr_Id (Data_Id = 22) - FP_Alarm (Data_Id = 80) Please note that the FP_Status_Message Data_Id is built up as follows: 100,0,20,01,fps,21,01,ns,22,02,acd Where: fps is the Fuelling Point Sate ns is the logical nozzle status acd is the Assign Control device The Data_Lg of the FP_Status_Message is always 0. *PCD Comment:* *Obviously this unsolicited Data_Id must be generated by the PCD when ever a change occurs in the state, logical nozzle or assignment.*	Bin8, bin8, bin16 Bin 64		M O
101 (65H)	***FP_Multi_Nozzle_Status_Message*** A FP_Multi_Nozzle_Status_Message must be sent unsolicited (without acknowledge) by the FP when ever a change has occurred in the status of the Multi_Nozzle_State (Data_Id 44). The FP_Multi_Nozzle_Status_Message includes - Multi_Nozzle_State (Data_Id = 44) Please note that the FP_Multi_Nozzle_Status_Message Data_Id is built up as follows: 101,0,44,01,mns Where: mns is the multi nozzle status The Data_Lg of the FP_Multi_Nozzle_Status_Message is always 0.	Bin8		O

FUELLING POINT DATABASE

DB_Ad = FP_ID (21H-24H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
102 (66H)	***FP_Running_Transaction_Message*** A ***FP_Running_Transaction_Message*** must be sent unsolicited (without acknowledge) by the FP whenever ***Running_Transaction_Message_Frequency*** exists in the database and is non zero and at the frequency defined by ***Running_Transaction_Message_Frequency.*** This unsolicited message should only be sent when the FP is in the Fuelling (8) or Suspended Fuelling (9) states. The ***FP_Running_Transaction_Message*** Includes: - Current_Amount (Data_Id 34) - Current_Volume (Data_Id 35) Please note that the FP_Running_Transaction_Message is built up as follows: 102,0,34,05,amount,35,05,volume The Data_Lg of the FP_Running_Transaction_Message is always 0.	Bin8+ Bcd8 Bin8+ Bcd8		O

N.B.

1) Flow control by nozzle is not to be used as an alternative to suspend/resume. It is specifically used where product flow selection through two or more nozzles is required.

2) Multi nozzle flow control is actioned dynamically during the course of a transactions and response time is 1 second.

3) Multi nozzle flow control produces no change in dispenser state.

MANUFACTURER / OIL COMPANY SPECIFIC				
200 to 255	Free to the manufacturer / oil company			

3.8 Logical Nozzle Data

This data allows the CD to configure and control the logical nozzle at a FP.
The access to the logical nozzle data is done by the database address FP_ID (fuelling point identification) + LN_ID (logical nozzle identification).
The LN_ID = 10H is used to ask for all logical nozzle at a fuelling point.

Please note that to allow dispensers to operate in 'stand alone' mode, the dispenser must have default settings for some of the Data_Id's contained in this database. I.e. the dispenser must configure these Data_Id's itself after a master reset/cold start.

LOGICAL NOZZLE DATABASE

DB_Ad = FP_ID (21H-24H) + LN_ID (11H-18H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
CONFIGURATION				

LOGICAL NOZZLE DATABASE

DB_Ad = FP_ID (21H-24H) + LN_ID (11H-18H)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
1 (01H)	***PR_Id*** Identifier of the product dispensed by this logical nozzle. The PR_Id (value 1-8) specifies the product the product which is stored in the Product Database PR_ID (address 41H-48H). 0 = not product assigned 1 = product in Product Database with address 41H 2 = product in Product Database with address 42H . 8 = product in Product Database with address 48H Please note that the PR_Id referenced here may differ from the PR_Id that is linked to the respective meter in the Meter database. If the logical nozzle is a blended product, then this PR_Id will defiantly be different. Please note that dispensers that do not permit this Data_Id to be changed remotely should: - Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). - Must set the *Data_Id* to the hard coded default value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *As with standard IFSF dispensers the PCD may or may not need product, nozzle & meter configuration from the SC/CD. However, it will be beneficial if the PCD can have a hardcoded value for many of these parameters.*	Bin8 (1-8)	R(1-9) W(1-2)	M

5 (05H)	***Physical_Noz_Id*** Indicates the physical nozzle identifier for this logical nozzle. The numbering of the physical nozzles is manufacturer model specific and must be defined separate. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (1-8)	R(1-9) W(1-2)	O
7 (07H)	***Meter_1_Id*** Indicates the meter identifier of the first base product. The numbering of the meters is manufacturer model specific and must be defined separate. 0 = not configured Please note that dispensers that do not permit this Data_Id to be changed remotely should: - Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). - Must set the *Data_Id* to the hard coded default value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *As with standard IFSF dispensers the PCD may or may not need product, nozzle & meter configuration from the SC/CD. However, it will be beneficial if the PCD can have a hardcoded value for many of these parameters.*	Bin8 (0-16)	R(1-9) W(1-2)	M

8 (08H)	***Meter_1_Blend_Ratio*** Indicates the blend ratio in percentage of the first base grade. The blend ratio of the second base grade is assumed to be the remaining percentage. 0 = no blending Please note that when the FP supports blending, this Data_Id must be implemented. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bcd2 (0-99)	R(1-9) W(1-2)	O
9 (09H)	***Meter_2_Id*** Indicates the meter identifier of the second base product. The numbering of the meters is manufacturer model specific and must be defined separate. 0 = no 2nd meter used Please note that when the FP supports blending, this Data_Id must be implemented. Please note that dispensers that do not permit this Data_Id to be changed remotely should: - Reject any write attempts with a Data_ACK value of 2 (Read Only/Not Writable). - Must set the *Data_Id* to the hard coded default value. When a master reset/cold start occurs on the dispenser device the dispenser should reset this Data_Id to its default value. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-16)	R(1-9) W(1-2)	O

10 (0AH)	***Logical_Nozzle_Type*** Indicates the type of nozzle: 0 = normal 1 = blender 2 = high speed 3 = satellite *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-3)	R(1-9) W(1-2)	O
3 (03H)	***Hose_Expansion_Vol*** Indicates the expansion volume in centilitres of the hose attached to this logical nozzle. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-255)	R(1-9) W(1-2)	O
4 (04H)	***Slow_Flow_Valve_Activ*** Indicates the number of centilitres when the FP's slow flow valve need to be activated. The point when the slow flow valves need to be activated is determined by the point when the: dispensed volume > = (the maximum volume limit – Slow_Flow_Valve_activ). *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-255)	R(1-9) W(1-2)	O
11 (0BH)	***Preset_Valve_Activation*** Number of centilitres needed to stop the preset valve. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bin8 (0-255)	R(1-9) W(1-2)	O
PERMANENT TOTALS				

20 (14H)	***Log_Noz_Vol_Total*** Volume total for the respective logical nozzle. The total update is done after the fuelling is stored in the transaction buffer. *PCD Comment:* *If the proprietary pump protocol doesn't supply the respective nozzle volume total the PCD will have to calculate the total itself.*	Long_ Volume	R(1-9) W(1-2)	M
21 (15H)	***Log_Noz_Amount_Total*** Amount total of the respective logical nozzle. The total update is done after the fuelling is stored in the transaction buffer. *PCD Comment:* *If the proprietary pump protocol doesn't supply the respective nozzle amount total the PCD will have to calculate the total itself.*	Long_ Amount	R(1-9) W(1-2)	M
22 (16H)	***No_TR_Total*** Number of transactions provided by this logical nozzle. *PCD Comment:* *The PCD will have to maintain this transaction count total.*	Long_ Number	R(1-9) W(1-2)	M
STAND ALONE TOTALS				
30 (1EH)	***Log_Noz_SA_Vol_Total*** Specifies the resetable volume tote of transactions done in stand alone mode by this logical nozzle. The total update is done after the fuelling is stored in the transaction buffer. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Long_ Volume	R(1-9) W(1-4)	O

31 (1FH)	***Log_Noz_SA_Amount_Total*** Specifies the resetable amount tote of transactions done in stand alone mode by this logical nozzle. The total update is done after the fuelling is stored in the transaction buffer. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Long_ Amount	R(1-9) W(1-4)	O
32 (20H)	***No_TR_SA_Total*** Specifies the resetable number of transactions provided in stand alone mode by this logical nozzle. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Long_ Number	R(1-9) W(1-4)	O

MANUFACTURER / OIL COMPANY SPECIFIC				
200 to 255	Free to the manufacturer / oil company			

3.9 Fuelling Transaction Data

This data allows the CD to handle the transaction data from a FP.
Access to the fuelling transaction data is done by database address FP_ID (fuelling point identification) + TR_DAT (transaction data) + TR_Seq_Nb (transaction sequence number).
Use TR_DAT = 20H and TR_Seq_Nb = "0000" to ask for all transactions on a fuelling point that are in the Payable (state 2) or Locked (state 3) state. The resultant database address (DB_Ad) is 2x200000H - where x takes value 1-4 depending on FP.
In this section (chapter 3.9) only, the "Read/ Write in State" column refers to the Transaction Buffer State Diagram because the Transaction Buffer State is independent of the Fuelling Point State.

FUELLING TRANSACTION DATABASE
DB_Ad = FP_ID (21H-24H) + TR_DAT (21H) + TR_Seq_Nb (0001-9999)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
TRANSACTION DATA				
1 (01H)	***TR_Seq_Nb*** Every transaction has a unique sequence number created by the FP. This number is the same number as used in the address of this database. *PCD Comment:* *The PCD will have to store in this Data_Id the transaction numbers that it used when this transaction was a current transaction.*	Bcd4 (1-9999)	R(1-3)	M
2 (02H)	***TR_Contr_Id*** Indicates the Controller Device that has released the transaction. A Logical Node Address (LNA) is used for the Release_Contr_Id. The LNA is specified by 2 bytes (S = Subnet, N = Node). For details see document "Part II, Communication Specification". 0,0 = Controller Device is not specified, X,Y = Controller Device that released the FP (X = Subnet, Y = Node), 255,255 = FP running in stand alone mode. At the end of the fuelling transaction the Release_Contr_Id (Data_Id 30 in FP Database) is stored here. *PCD Comment:* *The PCD will have to store in this Data_Id the controller Id that it used when this transaction was a current transaction.*	Bin16	R(1-3)	M

FUELLING TRANSACTION DATABASE DB_Ad = FP_ID (21H-24H) + TR_DAT (21H) + TR_Seq_Nb (0001-9999)				
Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
3 (03H)	***TR_Release_Token*** Indicates the Release_Token used when the transaction was started. At the end of the fuelling transaction the Release_Token (Data_Id 32 in FP Database) is stored here. *PCD Comment:* *The PCD will have to store in this Data_Id the release token that it used when this transaction was a current transaction.*	Bin8 (0-255)	R(1-3)	M
4 (04H)	***TR_Fuelling_Mode*** Indicates the fuelling mode used for this transaction. At the end of the fuelling transaction the Fuelling_Mode (Data_Id 33 in FP Database) is stored here. *PCD Comment:* *The PCD will have to store in this Data_Id the fuelling mode that it used when this transaction was a current transaction.*	Bin8 (1-8)	R(1-3)	M
5 (05H)	***TR_Amount*** Indicates the money amount of the transaction. At the end of the fuelling transaction the Current_Amount (Data_Id 34 in FP Database) is stored here. *PCD Comment:* *The PCD will have to store in this Data_Id the transaction amount. It is possible that the PCD will have to calculate this value from the received volume and unit price.*	Amount	R(1-3)	M
6 (06H)	***TR_Volume*** Indicates the dispensed volume of the transaction. At the end of the fuelling transaction the Current_Volume (Data_Id 35 in FP Database) is stored here. *PCD Comment:* *The PCD will have to store in this Data_Id the transaction volume. It is possible that the PCD will have to calculate this value from the received amount and unit price.*	Volume	R(1-3)	M
7 (07H)	***TR_Unit_Price*** Indicates the unit price of the dispensed fuelling product. At the end of the fuelling transaction the Current_Unit_Price (Data_Id 36 in FP Database) is stored here. *PCD Comment:* *The PCD will have to store in this Data_Id the transaction unit price. It is possible that the PCD will have to calculate this value from the received volume and amount.*	Unit_ Price	R(1-3)	M

FUELLING TRANSACTION DATABASE DB_Ad = FP_ID (21H-24H) + TR_DAT (21H) + TR_Seq_Nb (0001-9999)				
Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
8 (08H)	***TR_Log_Noz*** Indicates the logical nozzle that dispensed the fuel. At the end of the fuelling transaction the Current_Log_Noz (Data_Id 37 in FP Database) is stored here. *PCD Comment:* *The PCD will have to store in this Data_Id the logical nozzle used in the transaction. It is possible that the PCD will have to establish the logical nozzle from the product/grade Id returned by the proprietary pump protocol.*	Bin8	R(1-3)	M
9 (09H)	***TR_Price_Set_Nb*** Indicates the Price Set Number active at the time the transaction occurred. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 2 (Data not writable) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Bcd4 (0-9999)	R(1-3)	O
10 (0AH)	***TR_Prod_Nb*** Indicates the product number of the dispensed grade. At the end of the fuelling transaction the Current_Prod_Nb (Data_Id 38 in FP Database) is stored here. *PCD Comment:* *The PCD will have to store in this Data_Id the product number used in the transaction. It is possible that the PCD will have to establish the product number from the product/grade Id returned by the proprietary pump protocol.*	Bcd8	R(1-3)	M
11 (0BH)	***TR_Prod_Description*** Indicates the product description of the dispensed fuelling product. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 2 (Data not writable) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Asc16	R(1-3)	O

FUELLING TRANSACTION DATABASE
DB_Ad = FP_ID (21H-24H) + TR_DAT (21H) + TR_Seq_Nb (0001-9999)

Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
12 (0CH)	***TR_Error_Code*** Indicates the error code which may stopped the fuelling transaction. If the error codes = 0 then no error has occurred. (See the FP_Err_Type in the Error Code Database). At the end of the fuelling transaction the Current_TR_Error_Code (Data_Id 39 in FP Database) is stored here. *PCD Comment:* *The PCD will have to set this Data_Id to the IFSF dispenser error code value if an error occurred during the transaction.*	Bin8 (0-255)	R(1-3)	M
13 (0DH)	***TR_Average_Temp*** Indicates the average temperature of the dispensed fuel. At the end of the fuelling transaction the Current_Average_Temp (Data_Id 40 in FP Database) is stored here. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 2 (Data not writable) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Temp	R(1-3)	O
14 (0EH)	***TR_Security_Chksum*** This data element is used to sent a security checksum under the rules specified by the local W&M authority. The type of W&M security checksum is specified W&M_Security_Type (Data_Id 57 in the Calculator Database). *PCD Comment:* *The PCD will be responsible of calculating the W&M security checksum based on the agreed method for the country/region where the PCD is implemented.*	Bin24	R(1-3)	M
15 (0FH)	***M1_Sub_Volume*** Sub volume measured by the first meter (only present if blended fuel). *PCD Comment:* *The PCD will have to calculate this value if the proprietary pump protocol doesn't supply it.*	Volume	R(1-3)	M
16 (10H)	***M2_Sub_Volume*** Sub volume measured by the second meter (only present if blended fuel). *PCD Comment:* *The PCD will have to calculate this value if the proprietary pump protocol doesn't supply it.*	Volume	R(1-3)	M

FUELLING TRANSACTION DATABASE DB_Ad = FP_ID (21H-24H) + TR_DAT (21H) + TR_Seq_Nb (0001-9999)				
Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
17 (11H)	***TR_Tax_Amount*** The amount of tax for a given transaction. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 2 (Data not writable) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Amount	R(1-3)	O
TRANSACTION BUFFER STATUS				
21 (15H)	***Trans_State*** Used to indicate the state of a particular transaction buffer. Please see the Transaction Buffer State Diagram for details of the individual states (chapter 2.2 of this document). An unsolicited message (Data_Id 100) is generated by the FP for each change in the transaction buffer state. *PCD Comment:* *Please see the earlier chapter detailing how a PCD should handle the transaction states.*	Bin8 (1-3)	R(1-3)	M
20 (14H)	***TR_Buff_Contr_Id*** Indicates which CD has locked the transaction. This can be read in any FP state and written **ONLY** under fatal error conditions (see 2.2.3). 0,0 = transaction is unlocked and available to any CD X,Y = locked 255,255 = stand alone *PCD Comment:* *The PCD maintains the details of the SC/CD device that locks a transaction.*	Bin16	R(1-3) W(1-3)	M
TRANSACTION COMMAND				
30 (1EH)	***Clear_Transaction*** To clear a payable fuelling transaction in the transaction buffer. A transaction does not have to have been locked before it can be cleared. This command is allowed when Transaction Buffer is in state 2 or 3. Please note that an Unsolicited *Trans_State* (Data_Id 100) must be transmitted as a result of this command. This action must occur even if the state has not changed as a result of the command. Clear all transactions should not be implemented. *PCD Comment:* *The PCD must support this transaction command. See the earlier chapter dealing with transaction states.*	CMD	W(2,3)	M

FUELLING TRANSACTION DATABASE DB_Ad = FP_ID (21H-24H) + TR_DAT (21H) + TR_Seq_Nb (0001-9999)				
Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
31 (1FH)	***Lock_Transaction*** To lock an unlocked payable fuelling transaction in the transaction buffer. Dispenser should write the CD's Subnet & Node address to the *TR_Buff_Contr_Id.* This command is allowed in state 2 of Transaction Buffer. Please note that an Unsolicited *Trans_State* (Data_Id 100) must be transmitted as a result of this command. This action must occur even if the state has not changed as a result of the command. *PCD Comment:* *The PCD will have to support this transaction command. See the earlier chapter dealing with transaction states.*	CMD	W(2)	M
32 (20H)	***Unlock_Transaction*** To unlock a locked payable fuelling transaction in the transaction buffer. This command is allowed when Transaction Buffer is in state 3. The transaction can only be unlocked by the CD that locked it. An exception to this is when a transaction is unlocked using the TR_Buff_Contr_Id to unlock transactions that have been previously locked by an off line CD. See section 4.7 in the Communication Specification standard on how to determine if a CD is off line. In this way the Assign_Contr_Id is de-assigned. Unlocking a transaction which has been locked by an off line CD is achieved by setting the TR_Buff_Contr_Id to the same as the dispensers own application subnet and node. The dispenser should then reset the TR_Buff_Contr_Id to 0, 0 and change the Trans_State to PAYABLE. The *TR_Buff_Contr_Id* should be reset to 0,0 when the transaction is unlocked. Please note that an Unsolicited *Trans_State* (Data_Id 100) must be transmitted as a result of this command. This action must occur even if the state has not changed as a result of the command. *PCD Comment:* *The PCD will have to support this transaction command. See the earlier chapter dealing with transaction states.*	CMD	W(3)	M

FUELLING TRANSACTION DATABASE DB_Ad = FP_ID (21H-24H) + TR_DAT (21H) + TR_Seq_Nb (0001-9999)				
Data _Id	***Data Element Name*** Description	Field Type (Value)	Read/Write in State	M/O
UNSOLICITED DATA				
100 (64H)	***TR_Buff_Status_Message*** A TR_Buff_Status_Message must be sent unsolicited (without acknowledge) whenever the status of a transaction buffer has changed (transaction is created, locked, unlocked or cleared), or whenever the state cannot be changed following request by the CD to change state. This message includes the following data: - TR_Seq_Nb (Data_Id = 1) - Trans_State (Data_Id = 21) - TR_Buff_Contr_Id (Data_Id = 20) - TR_Amount (Data_Id = 5) - TR_Volume (Data_Id = 6) Please note that the TR_Buff_Status_Message Data_Id is built up as follows: 100,0,1,2,trn,21,1,trs,20,2,trcd,5,5,tram,6,5,vol Where: trn is the transaction sequence number trs is the transaction status trcd is the transaction buffer controller Id tram is the six bytes used to store the transaction amount vol is the six bytes used to store the transaction volume The Data_Lg of the TR_Buff_Status_Message is always 0. *PCD Comment:* *The PCD will have to generate this unsolicited message when ever any of the respective embedded Data_Id's change.*	Bcd4, bin8, bin16, Amount, Volume		M
MANUFACTURER / OIL COMPANY SPECIFIC				
200 to 255	Free to the manufacturer / oil company			

3.10 Error Code Data

This data allows the CD to handle the error data from a FP.
The access to the error data is done by the database address FP_ID (fuelling point identification) + ER_DAT (error data) + ER_ID (error identification).

The ER_DAT = 40H is used to ask for all error code data. Please note the dispenser should return all defined error codes in the below list (01H to 12H and 20H to 33H), even if the respective error event has not occurred. It is preferred Manufacturer Specific error codes are not returned, when all error code data is requested.

All error types listed below must be supported (01H to 40H).

Protocol Converter Device Comment
The PCD will have to convert any error codes it receives from the proprietary pump protocol to an IFSF Dispenser error code. Where the proprietary error code doesn't exist in the IFSF Dispenser error code table the PCD supplier will have to use the Manufacture/oil company free fields to reflect the error. Obviously these new manufacturer specific error codes will have to be documented.

ERROR CODE DATABASE

DB_Ad = FP_ID (21H-24H) + ER_DAT (41H) + ER_ID (01H-40H)

Data _Id	***Data Element Name*** Description	Field Type	Read/Write in State	M/O
ERROR DATA				
1 (01H)	***FP_Error_Type*** Every error has a unique error code. This number is the same number as used in the address ER_ID of this database. A list of all errors is at the end of this table. An unsolicited message is generated by the FP when a major or minor error occurs. *PCD Comment:* *The PCD will have to convert any error codes it receives from the proprietary pump protocol to an IFSF Dispenser error code. Where the proprietary error code doesn't exist in the IFSF Dispenser error code table the PCD supplier will have to use the Manufacture/oil company free fields to reflect the error.*	Bin8 (1-64)	R(1-9)	M

ERROR CODE DATABASE

DB_Ad = FP_ID (21H-24H) + ER_DAT (41H) + ER_ID (01H-40H)

Data _Id	***Data Element Name*** Description	Field Type	Read/Write in State	M/O
2 (02H)	***FP_Err_Description*** Description of the error. *PCD Comment:* *As this is an optional data field the PCD can NAK any write requests to this Data_Id with a Data_ACK code of 4 (Data does not exist in this device) or reply to any read request with an answer message with the respective Data_Id's length set to 0.*	Asc20	R(1-9) W(1-2)	O
3 (03H)	***FP_Error_Total*** Total of error having that code. If more that 255 errors are counted, the value remains 255. When a 0 value is written in this FP_Error_Total, the total is cleared. *PCD Comment:* *The PCD will need to maintain the error total count.*	Bin8 (0-255)	R(1-9) W(1-2)	M
5 (05H)	***FP_Error_State*** Specifies the FP State during which the latest error (with the selected ER_ID) occurred. The FP state numbering described in the "FP State Diagram, chapter 2.1" is used. *PCD Comment:* *The PCD will need to maintain the FP error state.*	Bin8 (1-9)	R(1-9)	M
UNSOLICITED DATA				
100 (64H)	***FP_Error_Type_Mes*** A FP_Error_Type_Mes message must sent unsolicited (without acknowledge) when ever a the error occurs. This message includes the following data: - FP_Error_Type (Data_Id = 1) - FP_Error_State (Data_Id = 5) *PCD Comment:* *The PCD will have to generate this unsolicited message whenever an error is logged in this DB.*	Bin8, bin8		M

MANUFACTURER / OIL COMPANY SPECIFIC

ERROR CODE DATABASE

DB_Ad = FP_ID (21H-24H) + ER_DAT (41H) + ER_ID (01H-40H)

Data _Id	***Data Element Name*** Description	Field Type	Read/Write in State	M/O
200 to 255	Free to the manufacturer / oil company			

The errors have different priorities. In the following table the classification is done. For details in the behaviour of the FP see chapter 2 (Fuelling Point Behaviour Model).

Classification	ER_ID	Description
MAJOR ERROR	1H	RAM defect
	2H	ROM defect
	3H	Configuration or Parameter Error
	4H	Power supply out of order
	5H	Main Communication error
	6H	Display error
	7H	Pulser error
	8H	Calculation error
	9H	Blender error
	0AH	Download error
	0BH	Checksum error
	0CH	Leak Error
	0DH	PCD RAM defect
	0EH	PCD ROM defect
	0FH	PCD Configuration or Parameter Error
	10H	PCD Power supply out of order
	11H	PCD Main Communication error
	12H	Vapour Recovery Error
	13H-1FH	Spare
MINOR ERROR	20H	Battery error
	21H	Communication error
	22H	Customer_Stop_Pressed
	23H	Spare
Fuelling Errors	24H	Authorise_Time_Out
	25H	Fill_Time_Out
	26H	No_Progress
	27H	Limit_Reached
	28H	Fuelling suspended
	29H	Fuelling resumed
State Error	2AH	Vapour Recovery Timer Started
	2BH	Vapour Recovery Timer Reset
	2CH	Vapour Recovery Module Defect

Classification	ER_ID	Description
	2DH	State error 1: FP is state INOPERATIVE
	2EH	State error 2: FP in state CLOSED
	2FH	State error 3: FP is already opened
	30H	State error 4: Transaction not in progress
	31H	State error 5: Transaction already started
	32H	State error 6: Parameter/Configuration change not possible
	33H	CD identifier not correct (assign, release, resume, clear)
	34H	Urea temperature low, heater failed. This is an optional minor error and is only required by Urea dispensers. It should not be supported by other dispensers e.g. Diesel, LPG, petroleum, etc.
	35H-37H	Spare
Manufacturer Specific	38H-40H	Spare

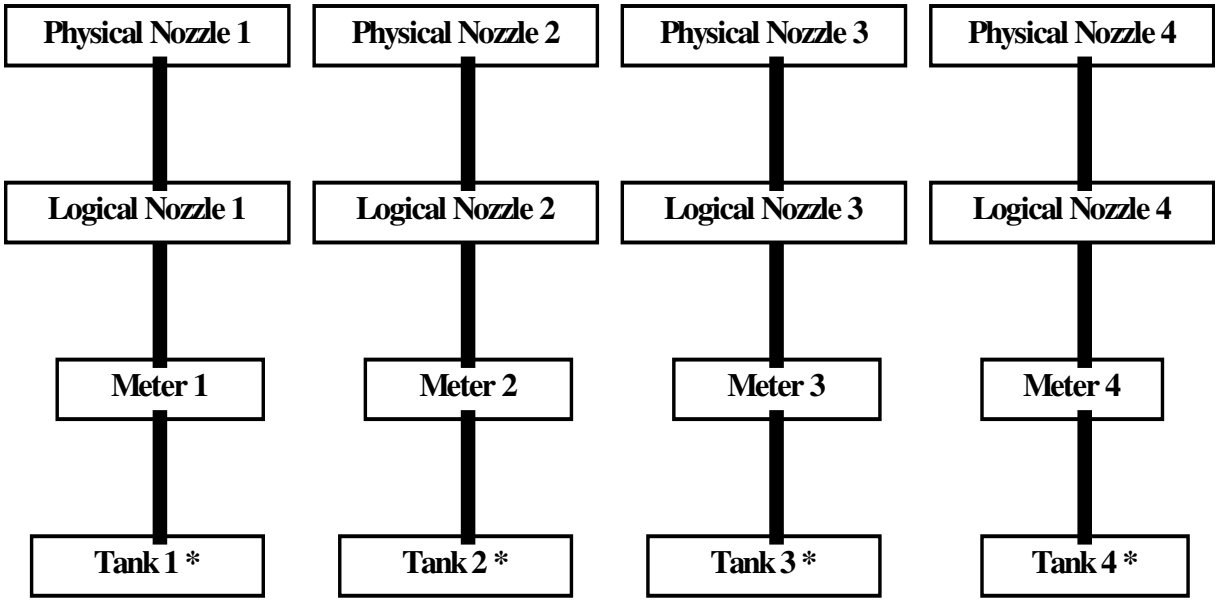
3.11 Data Download

In Version 2.13 standard tools will be used. This section is deleted.

4 Example Configuration Diagrams

This section gives an example of the configuration of a Multi Product Dispenser (MPD) and a typical blender dispenser. The purpose is to show the relationship between the Physical Nozzles, Logical Nozzles and the Meters.

4.1 Multi Product Dispensers

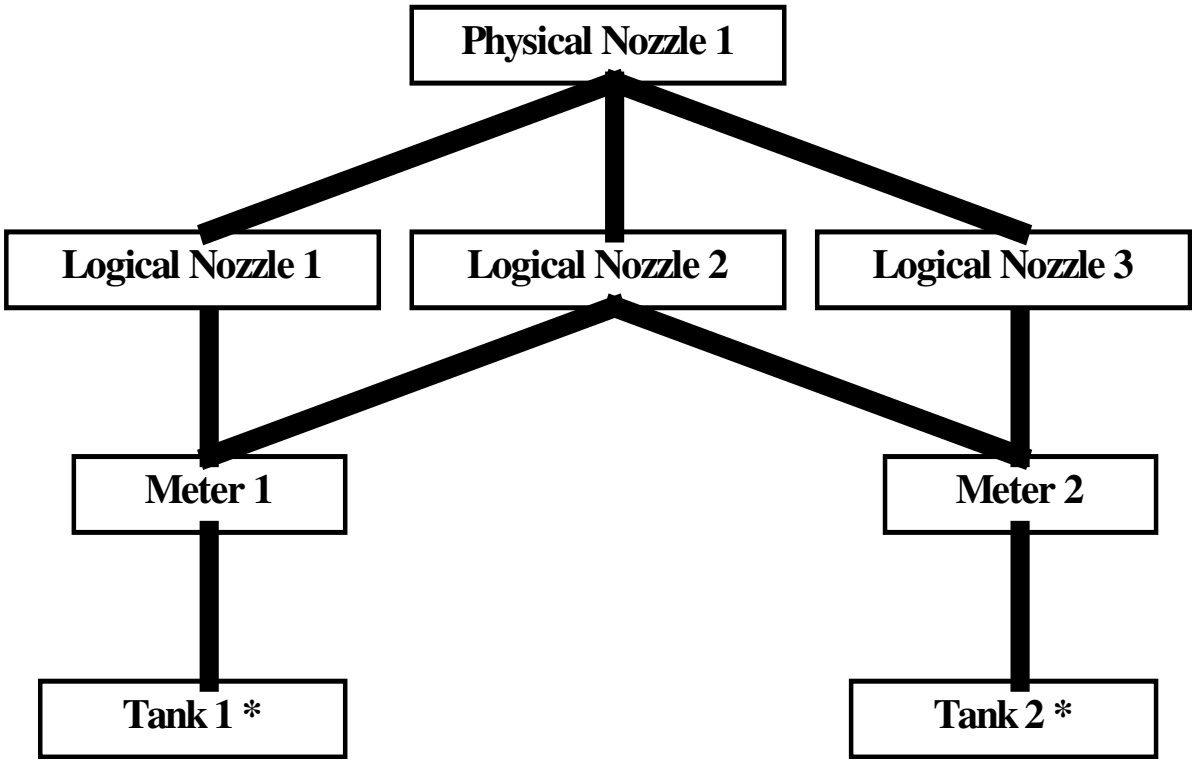


In this example a MPD is configured with 4 grade options. Each grade option has a direct relationship with its own meter, logical nozzle and physical nozzle. This dispenser does not support blending.

With this set-up, it is possible to establish the volume of fuel dispensed per logical nozzle and establish the volume of fuel dispensed through each meter, which usually have a direct relationship with the tanks that source the fuel.

* = Please note that the Tanks featuring in this diagram are not configured via the Dispenser Application Protocol.

4.2 Blender Dispenser



In this example a blending dispenser is configured with 3 logical nozzles/grade options.

- The first logical nozzle has a direct relationship with 'meter 1' (i.e. all fuel dispensed by it is totalised against the meter 1 totals), The logical nozzle shares a physical nozzle with the other 2 logical nozzles. This logical nozzle is not a blend product.
- The second logical nozzle has a relationship with 'meter 1' & 'meter 2' (i.e. all fuel dispensed by it is totalised against the meter 1 & meter 2 totals. The ratio is given specified by the *Meter_1_Blend_Ratio* Data_Id), The logical nozzle shares a physical nozzle with the other 2 logical nozzles. This logical nozzle is a blend product.
- The third logical nozzle has a direct relationship with 'meter 2' (i.e. the fuel dispensed by it is totalised against the meter 2 totals), The logical nozzle shares a physical nozzle with the other 2 logical nozzles. This logical nozzle is not a blend product.

With this set-up, it is possible to establish the volume of fuel dispensed per logical nozzle and establish the volume of fuel dispensed through each meter, which usually have a direct relationship with the tanks that source the fuel.

* = Please note that the Tanks featuring in this diagram are not configured via the Dispenser Application Protocol.

5 Implementation Guidelines & Recommendations

This section gives guidelines & recommendation for implementations of the IFSF Dispenser Application Protocol.

5.1 Handling after a Device Master Reset/Cold Start or Initial Start-up

After a master reset, cold start, initial start-up or discovery that the device's configuration is corrupted, the dispenser should:

- Initialise the Communication Specification's Heartbeat_Interval to 10 seconds.
- Start generating Heartbeat messages with a Device_Status indicating that configuration is required.
- Reset the Communication Specification's Recipient Address Table.
- Clear out all current & historic transactions and initialise all other fields.
- Where a default value exists for a Data_Id, the dispenser should set up the Data_Id's value accordingly.

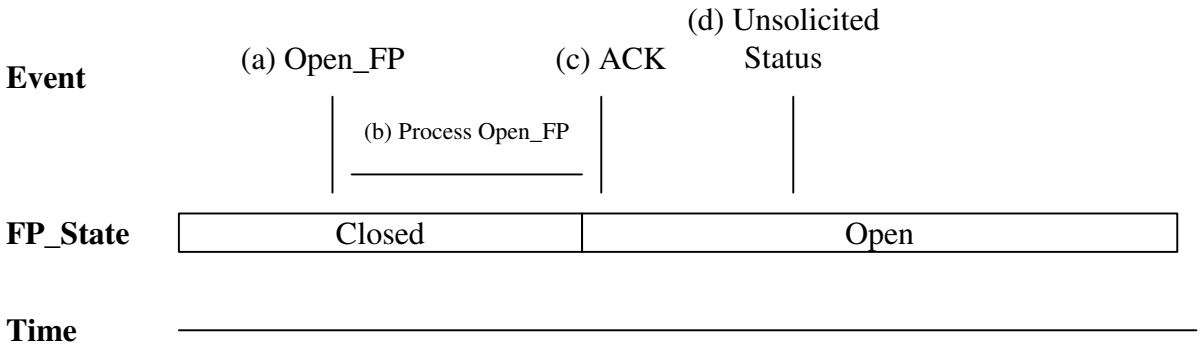
5.2 Handling After a Reset or Power Off

After a master Reset or Power Off of the dispenser the device should:

- Send a Configuration Needed heartbeat, since it cannot know if it has the correct prices.
- **Do not** clear the Communication Specification's Recipient Address Table .
- **Do not** clear current & historic transactions.
- **Do not** unlock locked transactions.
- **Do not** reset Data_Id's to their default values.

5.3 Dispenser Behaviour after an Acknowledgement of a Command

When a dispenser receives a command from the CD (i.e. Open_FP, Close_FP, Release_FP, etc.) and the dispenser acknowledges the command positively (i.e. with a MS_ACK=0). The dispenser must change to the new state immediately before sending the Acknowledge reply. Please see the diagram/example below showing the required steps.



Where:
a = Open_FP command sent by the Control Device.
b = Dispenser validates and processed the Open_FP command and decides that the FP can be opened. Dispenser changes its state to 'Open'.
c = Dispenser Replies to the Open_FP command by sending a positive acknowledgement.
d = Dispenser generates the unsolicited FP_Status_Message to all devices entered in its Recipient Address table.

5.4 Dispenser & Site Controller On-Line & Off-Line Handling

5.4.1 Actions when a Dispenser recognises that a SC is off-line

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

DO:

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

DO NOT:

- If currently fuelling do not stop the transaction. Continue to dispense fuel until the end of the transaction or a previous defined limit has been reached. Basically the off-line situation has no affect on the operation of the Dispenser.
- Do not remove the off-line SC device from the Recipient Table.

5.4.2 Actions when a Dispenser recognises that a SC comes back on-line

The Dispenser can recognise that a SC device that has been entered into its recipient table is back on line when it receives a heartbeat from the SC device.

DO:

- Send the unsolicited status and transaction messages to the SC that has come on-line. Please note that this event will cause every dispenser to do the same action, so the network could get ‘busy’.
- Start sending unsolicited messages to the SC device as normal.

5.4.3 Actions when a SC recognises that a Dispenser is off-line

The SC recognises that the Dispenser device has gone off-line when it does not receive a heartbeat within the duration of 3 times the heartbeat interval (normally 3 times 10 seconds = 30 seconds).

DO:

- Indicate to the system operator (cashier) that the Dispenser device has gone off-line.

5.4.4 Actions when a SC recognises that a Dispenser comes back on-line

The SC recognises that the Dispenser device has come back on-line when it receive a heartbeat from the Dispenser Device.

DO:

- Request the Dispenser's status and transaction details.

5.4.5 Correct Manner of removing a SC from the Network

When a SC is to be removed from the network or taken off-line the following actions should be carried out:

- Remove the SC's address from all dispensers' Recipient Table.

5.4.6 Actions when a dispenser recognises that the line is cut

The dispenser recognises that a SC device does not answer to a message on the physical level (no ACKNOWLEDGEMENT on LONTalk level also after the retries).

DO:

- Stop sending any more unsolicited messages to the SC device

Repeat all unanswered and outstanding messages when the Dispenser recognises that the line is back again (the dispenser recieves a heartbeat from the SC device before heartbeat timeout).

DO NOT:

- Remove the SC device from the Recipient Table
- Send any more messages to the SC.

5.5 Dispenser Stand Alone Behaviour

Definition - A dispenser is in Stand Alone model when it is not controlled and there is no reliable read/write activity - this can be for a number of reasons. When a dispenser recovers out of Stand Alone mode it must be re-configured.

When a Dispenser is put into stand alone mode it must carry out the following actions:

- For all the Fuelling Points controlled by the dispenser it must send out the unsolicited message *FP_Status_Message* (Data_Id 100) to all CD's entered in the Dispenser's Recipient_Table. The *FP_Status_Message* must indicate that the FP has been assigned to stand alone mode (*Assign_Contr_Id*=255,255).
- Continue communications (including the sending of Heartbeats) to all devices logged in the Recipient_Table. This will allow Control Devices *i.e.* Tank Level Gauges that need to know what transactions have been occurring on the site to still go and get the transaction details.

When a Dispenser is removed from stand alone mode it must carry out the following actions:

- For all the Fuelling Points controlled by the dispenser it must send out the unsolicited message *FP_Status_Message* (Data_Id 100) to all CD's entered in the Dispenser's Recipient_table. The *FP_Status_Message* must indicate that the FP is no longer assigned (*Assign_Contr_Id*=0,0).

5.6 Units Of Measurement

The IFSF Dispenser Application Protocol works on the bases that the units of measurement are implied. I.E. If the environment where the device is installed works in Litres, then all the volume fields will also be Litres. Alternatively, If the environment where the device is installed works in Gallons, then all the volume fields will also be Gallons.

Should an IFSF Dispenser be placed in an environment where both Litres & Gallons are used. It will be the responsibility of the CD/Site controller to correctly configure the dispenser with the correct prices for each product and provide transaction & meter details to other applications (i.e. Pump controller/POS) running at the site in their required unit format.

5.7 Transaction Terminated - Nozzle not returned

The standard 8 second time-out for responses to messages can be ignored in the case of a state transition from Fuelling to Idle. This is due to the need for extra time to take account of inertia within the fuelling termination process.

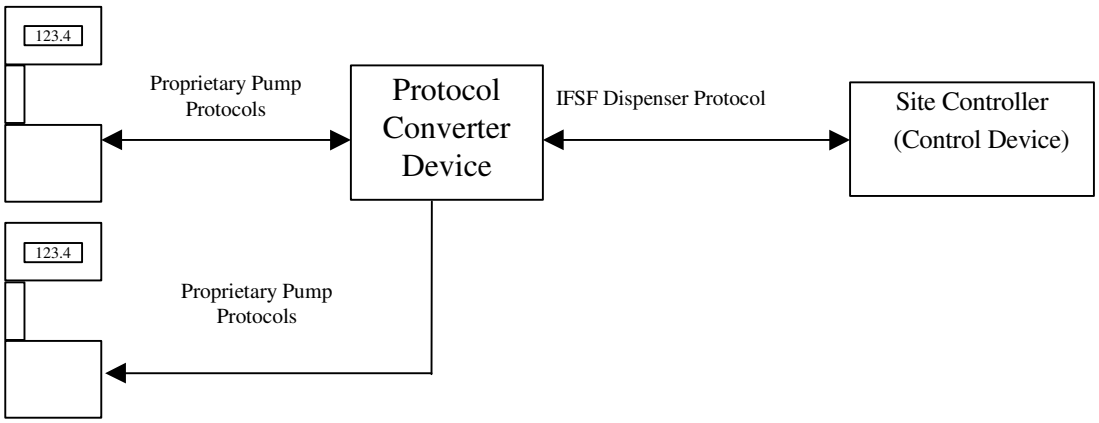
6 Protocol Converter Device Implementation Guidelines

This section gives guidelines & recommendation for implementations of Protocol Converter Devices (PCD) that control non IFSF Dispenser.

6.1 Overview of a Protocol Converter Device (PCD)

A Protocol Converter Device (PCD) is a hardware device that is located between non IFSF dispensers/pumps and converts their proprietary pump protocols to the IFSF Dispenser Application.

Please note that a PCDs may also control devices other than dispensers.



The PCD may have a one to one relationship with a dispenser/pump or may be capable of controlling several dispensers. In some circumstances the PCD may actually reside physically in the dispenser. It is also foreseen that more than one PCD may be connected to the SC/CD.

The task that a PCD has to accomplish is to successfully make the SC/CD believe that it is communicating directly with a fully IFSF compatible device.

6.2 Configuration of the PCD

The PCD can not expect to receive configuration from the IFSF SC/CD as the SC/CD will not have all the required configuration information required for a device controlling different types of dispenser/pumps. Hence the PCD supplier will have to provide the means of configuring their device separately.

Some PCD Parameters that will not be known by the IFSF SC/CD are:

- Link between Proprietary device address and the IFSF logical (LNA) address.
- Proprietary protocol used by the proprietary devices.
- Default values for varies IFSF and non IFSF parameters.
- Etc.

6.3 Device Addressing

The PCD may be controlling more than one proprietary dispenser. Hence, there is a requirement that the PCD is capable of supporting more than one logical device on one physical address. This intern implies that the IFSF SC/CD devices must be capable of communicating with devices that have a different physical address (PNA) to the logical address (LNA).

To allow more than one PCD to be active on the site the PCD will have to allow its physical address (PNA) to be configured. The PCD also needs to allow the logical (LNA) address for each of the proprietary dispensers/pumps to be set up.

A subnet address for PCDs will need to be defined in the IFSF Communication Subnet table.

6.4 Heartbeat Handling

The PCD must provide an IFSF communications database for each proprietary device connected to it. This is important, as the PCD will also have to generate the IFSF heartbeats as if these devices were native IFSF devices. Hence each device will need to have its own heartbeat interval and recipient table.

The PCD and the IFSF SC/CD must also be aware that the heartbeat messages can be transmitted and received from devices where the physical (PNA) and logical (LNA) addresses may be different.

6.5 General Rules

The PCD must make every effort to implement the fullest IFSF implementation possible. However, there are some circumstances where 100% compatibility is not possible. In these cases the PCD will have to indicate clearly to the SC/CD that a command or Data Id read or write operation could not be carried out. The IFSF protocols have a comprehensive set of standard reject codes that can be used to indicate why some action was not carried out.

For any command that can not be carried out the PCD can unless otherwise documented reject the write message with a MS_ACK=5 and a Data_ACK=5 (Command not understood / implemented) and stay in the same device state.

For any read attempt that can not be supported by the PCD, the PCD will unless documented otherwise generate an answer message with the respective Data_Id data length set to 0.

Where a proprietary pump protocol doesn't supply the data needed in the IFSF database the PCD should make every attempt to generate the missing data.

6.6 Known Limitations

It needs to be stated that a PCD can translate the proprietary pump protocols to the IFSF Dispenser Application protocol, but will not be able to:

- Change measurement or transaction data received from the proprietary pump. The data accuracy is the responsibility of the dispenser/pump.

- Support some functions when the proprietary dispenser doesn't have the hardware to support them i.e. where no slow flow valves are present to allow the dispenser to accurately stop the delivery on a supplied volume or amount limit.

Where a PCD can not support a function that can be supported by a native IFSF compatible device the PCD supplier **must** document this shortcomings so that customers are aware of it.