INTERNATIONAL FORECOURT STANDARDS FORUM

STANDARD FORECOURT PROTOCOL

PART 3-27

FDC POS STANDARD INTERFACE

VERSION 01.00 - November 2010

This document is written by the IFSF Dispenser Working Group:

Name	Company	Telephone
Rainer Kramer	Wincor Nixdorf International GmbH	+49 4063603 205
Jan Psenicka	Radiant Systems	+420 246 09 1035
Massimiliano Polito	Gilbarco – Veeder Root.	+39 055 3094 926
Matthias Lürkens	Gesytec GmbH	+49 2408 944 262
Alexandre Vargas	Petrotec	+351 21 225 09 54
Volker Schulz	Scheidt & Bachmann GmbH	+49 2166 266 569

For further copies and amendments to this document please contact:

IFSF Technical Services.
E-mail: techsupport@ifsf.org

Their contact details and the latest revision of this document can be found on the Internet at the following address:

Internet address: www.ifsf.org

Contents

1 Document History

State	Date	By	Changes
Draft	12-20-2004	Rainer Kramer	FDC POS Standard Interface – Initial Version (0.1)
Draft	03-31-2005	Rainer Kramer	IFSF Draft – Version (1.0.1)
Draft	25-05-2005	Rainer Kramer	Updates & extensions
Draft	31-01-2008	Jan Psenicka	Comments
Draft	05-05-2008	Rainer Kramer	WG changes
Draft	05-19-2008	Jan Psenicka	Use cases, other changes
Draft	05-27-2008	Massimiliano Polito	
Draft	06-09-2008	Matthias Luerkens	
Draft	18-09-2008	Alexandre Vargas	Comments
Draft	06-19-2008	Volker Schulz	Comments
Draft	23-07-2008	Rainer Kramer	WG changes
Draft	06-08-2008	Rainer Kramer	WG changes
Draft	26-09-2008	Rainer Kramer	WG changes
Draft	30-10-2008	Rainer Kramer	WG changes
Draft	15-12-2008	Hans-Dieter Bender	Communication Interface changes
Draft	11-12-2009	N Bradshaw	WG changes
Draft	20-02-2010	N Bradshaw	WG changes
Draft	19-03-2010	N Bradshaw	WG changes
Draft	19-09-2010	N Bradshaw	WG changes
Final	23-11-2010	N Bradshaw	

2 References

Document	Version	Date
Standard Forecourt Protocol – Dispenser Application -	2.31	July 2008
IFSF ENGINEERING BULLETIN NO. 7 DISPENSER CRC SIGNATURE GENERATION	1.01	November 2000
Standard Forecourt Protocol- Price Pool Application	1.23	July 2008
Standard Forecourt Protocol-Tank Level Gauge Application	1.29	March 2008
Communication Specification over TCP/IP	1.02	June 2004

3 Definitions

Name	Description
ACK	Positive Acknowledge
ARTS	Association for Retail Technology Standards
CD	Controller Device also called FDC, Forecourt Device Controller
CWU	Carwash Unit
CWP	Carwash Point
DeviceClass	Collection of properties of a specific forecourt device like DSP, FP, OPT, PSN, TLG, TP etc.
DSP	Fuel Dispenser
FDC	Forecourt Device Controller also called CD, Controller Device
FP	Fuelling Point
IANA	Internet Assigned Number Authority
IX-Retail	IXRetail Release XML Price and Digital Receipt Schemas for Retail Industry
LAN	Local Area Network
LON	Local Operating Network
NACK	Negative Acknowledge or disconnected message
NACS	National Association of Convenience Stores
Nozzle	Part of a fuelling point, a spout at the end of a hose by which a stream of petrol can be controlled.
OPT	Outdoor Payment Terminal
PCATS	Petroleum Convenience Alliance for Technology Standards
ProductNo	Item number of a pure fuel product
ProductModeCode	Combination of fuel product number and fuel mode with an assigned fuel price
PP	Price Pole
TLG	Tank Level Gauge
TP	Tank Probe
UnitNo	Unique unit number for a device in a given device class. The number is static in a current forecourt configuration.
XML	Extensible Markup Language
XSD	XML Schema Definition, describes the structure of an XML document

4 Introduction

A Forecourt Controller Device (FDC) is a software application that controls the various forecourt devices of a petrol station using a physical LON or LAN network. Primarily it is focused on physical and logical device control. It controls the device states and makes sure, non unauthorised state changes are held and processes follow given IFSF regulations.
A Forecourt Device Controller also manages individual business processes on petrol stations. That includes fuel sales under consideration of given business rules and country specific requirements. Sales transactions are sent to POS systems from the FDC. The POS can also be used to enter pre-payments, price changes for fuel items and release pumps on demand.
Making the Forecourt Controller Device flexible, so it can support POS systems from different suppliers requires a detailed description of the commands and information flow interchanged between Forecourt Controller Device and POS. A standard interface between POS and Forecourt Controller Device must also simplify the complexity of the FDC commands for the POS system. The FDC-POS standard interface offers only a subset of the whole FDC IFSF commands between FDC and POS and hides the complexity of forecourt control from the POS application. However, the POS must be able to fulfil the required business processes on the petrol station with the given commands.

5 Objectives

The purpose of this document is to describe the necessary logically commands to communicate between an IFSF Forecourt Controller Device and one or more POS systems. It describes the contents of the data interchanged between POS and FDC. The target is to achieve the most common acceptance of the data and command descriptions by different FDC and POS suppliers.
The whole interface will be described and created as an XML interface.

6 System Architecture

The following chapter describes the main functions of the POS application and the role of the Forecourt Device Controller application.

6.1 POS Application

The most important functions supported by the POS are the following:

- Performing the sales process, e.g. scanning shop items, calculating discounts, getting input for card payment and printing the customer receipt.
- Release of pumps, in case of manual release using the FDC to forecourt interface (standard IFSF LON/LAN based interface).
- Visualisation of fuelling points (Volume, Value and Device status).

- Transaction payment with credit and debit cards. Black and white lists for cards.
- Electronic journal for all transactions and payments.
- Tank Level Gauge management including alarms.
- Fuel deliveries and transfers.
- Shop and fuel price maintenance.
- Shift management and day end reconciliation.
- Internationalisation and legal requirements.

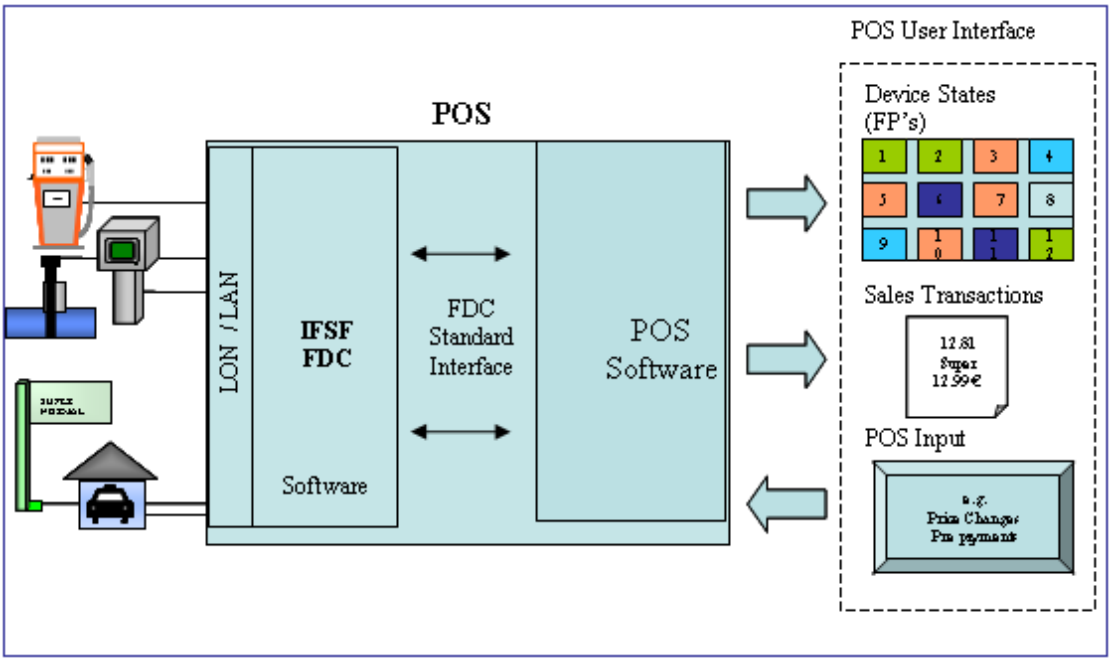
6.2 FDC Application

The FDC application provides the following functionality:

- Logical and physical forecourt configuration.
- Forecourt device control (start, stop, release, reserve) of forecourt devices (Dispenser, Fuelling Points, Automatic Tank Gauges, Tank Probes, Car Wash, Out-Door Payment Terminals etc.).
- Reading and writing device properties.
- Performing device commands.
- Notification of device errors and exceptions.
- Storage of logging information about all events, errors and exceptions for all forecourt devices.
- Control of the physical LON or LAN Network.
- Guarantee of IFSF standard data and rules for forecourt devices.
- Support of forcourt business logic.
- Weight and Measure regulations.

7 Implementation Concept

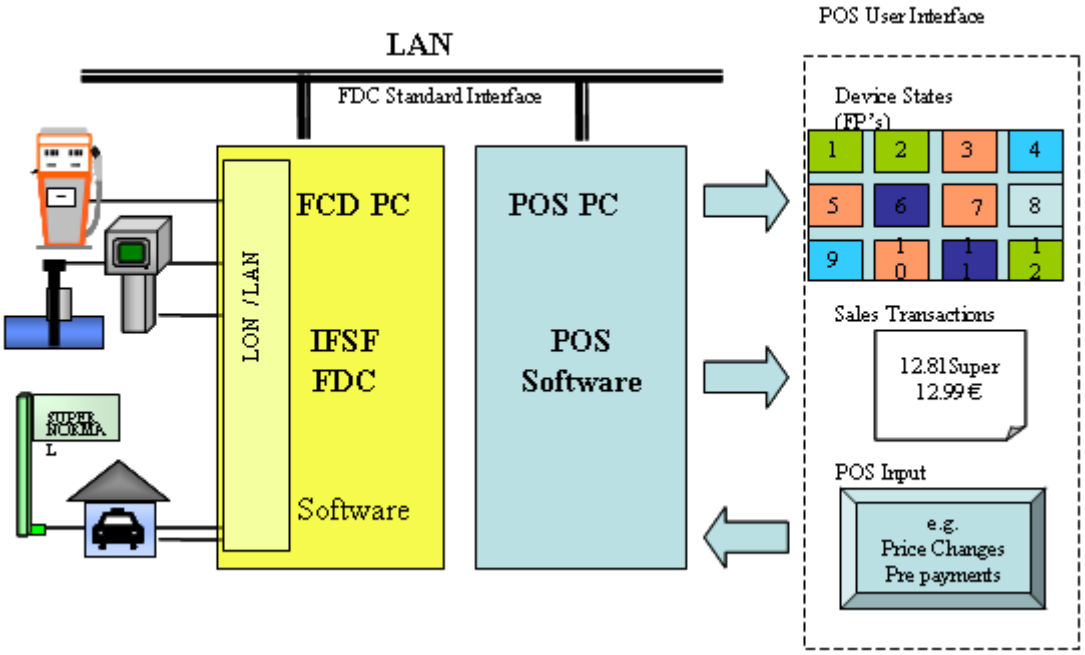
The following picture indicates where the FDC – POS standard interface is located.



This picture shows the FDC software and POS software running in one PC system, e.g. a POS. Communication between FDC application and POS application is using TCP/IP via sockets. This kind of communication is sometimes called Client-Server communication, where one application is called the server and the other one is called the client application. It is usually arbitrary and not important which application is called the client or the server, because there is an interchange of information in both directions.

It is important to remember, the IFSF FDC application software is able to run on the same computer as the POS software. If the IFSF FDC software is running on a separate computer, the POS software communicates through a LAN to process FDC commands and interchange data. In this way, several POS systems can communicate with one Forecourt Device Controller application at the same time.

POS and FDC application running on separated PC's.



The communication between a separate IFSF FDC-PC and the POS-PC software makes use of TCP/IP and socket communication. This technology is supported by many operating systems and allows program to program communication in one computer or via LAN.

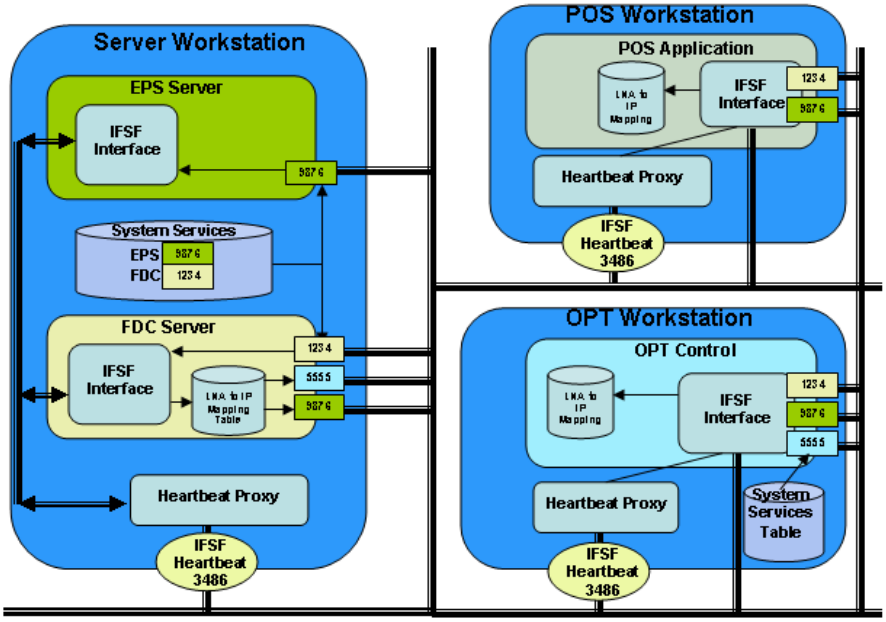
7.1 Communication Interface

The communication interface between the Forecourt Device Control Server (FDC) and its client (e.g. POS) is based on the following principles:

- Only one socket per application for request, response and unsolicited messages
- Independent configuration for server and clients
- Heartbeat management via existing IFSF components

The overall design is adopted from IFSF standard as given with "COMMUNICATION SPECIFICATION OVER TCP/IP" version 1.02 from June 2004. This specification describes the design of an IFSF network where the forecourt devices are all connected to a local area network (LAN). The methods, modules and technical approaches that are necessary to build up an IFSF LAN can be used to establish a TCP/IP based communication between high-level applications like EPS and FDC Server, too. The main difference is the architecture of the data records exchanged between the applications. This is XML based in case of server applications and standard IFSF format in case of forecourt applications (e.g. Dispenser, tank level gauge).

By the way this communication model is usable in the same way for the POSToEPS specification.



7.1.1 IFSF Interface

This module is responsible for the interface between the IFSF application (e.g. FDC Server) and other IP communication services. It will maintain a table of all Logical Node Addresses (LNA) and their corresponding IP/port addresses (a combination of IP address, protocol and port number corresponds to the socket address). This module will route all heartbeat messages to the **Heartbeat Proxy** and all other messages to the connected applications. The IFSF interface will receive heartbeats from other devices, add the LNA socket address to the table (if not already in the table), and then send the received IFSF heartbeat to all applications the interface is hosting. Existing IFSF interfaces must be extended to deal with XML based record types.

7.1.2 IFSF Heartbeat Proxy

The requirements on a heartbeat proxy module are described in the "IFSF Communication Specification over TCP/IP" dated from June 2004. A heartbeat proxy is needed at all if at least one member of an IFSF network depends on a communication via a local network. Therefore this does not create additional impact on the implementation of an IFSF compliant Forecourt Control Interface.
The following explanation is copied from the "IFSF Communication Specification over TCP/IP"

*"This module is responsible for packaging local application heartbeat messages and broadcasting them using UDP datagram. It is from here the UDP datagram is broadcast to the 'well known' IFSF heartbeat port. Incoming heartbeat messages come through this module, and are sent through the **IFSF interface**. The proxy will also send the heartbeats it receives from a local device to other locally hosted devices.*

An IFSF heartbeat contains the LNA and a device status bit. To be effective on the IP network this message needs to have augmented to it the IP address of the host of the local IFSF application and the port number on the local host that a remote device uses to connect to the local IFSF application. When a remote device receives this message it will strip off the IP and port number information, record the LNA of the sending device, and pass the IFSF heartbeat message on to the IFSF application in the standard IFSF protocol format. The remote device will take the data from the received message and make an entry in a table that maps the IP and port address to the LNA of a device that has announced itself to the network, which we will call the LNA to IP mapping table.
Each time a heartbeat message is received by the IIPC it will reset a timer for that remote device. The purpose of the timer is to notify the IIPC when a heartbeat has not been received for a period of time. When the IIPC gets that notice it assumes the remote device has gone off-line and removes it from the LNA to IP mapping table, sends a connection closed message to the remote device, and closes any local connections associated with the device other than the main service connection. The next time a heartbeat or TCP connection request comes in from the remote device then a new entry will be made in the LNA to IP mapping table and the timer is started again."

FDC application sends a heartbeat to the POS and vice versa, POS sends heartbeat to FDC. The heartbeat interval is by default set to ten seconds.

In applications, using the IFSF/IP standard, there must be means to distinguish from FDC and standard IFSF heartbeats. Both are using the same IP port and identical UDP messages, without taking care FDC and IFSF LNA/IP mapping will be mixed.
The standard IFSF heartbeat is formatted like this:

IFSF Heartbeat	
HOST IP	*4 byte*
Port	*2 bytes*
LNAO	*2 bytes*
IFSF_MC	*1 byte*
STATUS	*1 byte*

For IFSF Heartbeat communication the IFSF_MC is defined as 1. Other IFSF_MC are 0 and 2. Using different content for LNAO and message code allow a simple separation from IFSF Heartbeat and FDC Heartbeat. The FDC Heartbeat uses a LNAO (e.g. FDC LNAO could be 1B01) and a message code that is defined as 3.

FDC Heartbeat	
HOST IP	*4 byte*
Port	*2 bytes*
LNAO	*2 bytes*
FDC_MC	*1 byte = 3*
STATUS	*1 byte*

UDP and application Heartbeats are mandatory. If either Heartbeat disappears the device/ application is off-line.

7.1.3 LNA to IP mapping Table

This table contains a list of LNA/IP pairs that is needed to route messages via the IFSF LAN network. It is also used to notify the time when the last heartbeat from this address was received.

7.1.4 System Services Table

This table may be used for the definition of port numbers that a local server process (e.g. FDC Server) must use for its welcome socket. It is recommended to use the file named ***services*** that is located in the folder **system32/drivers/etc** of the Windows installation (e.g. ***C:\Windows***).

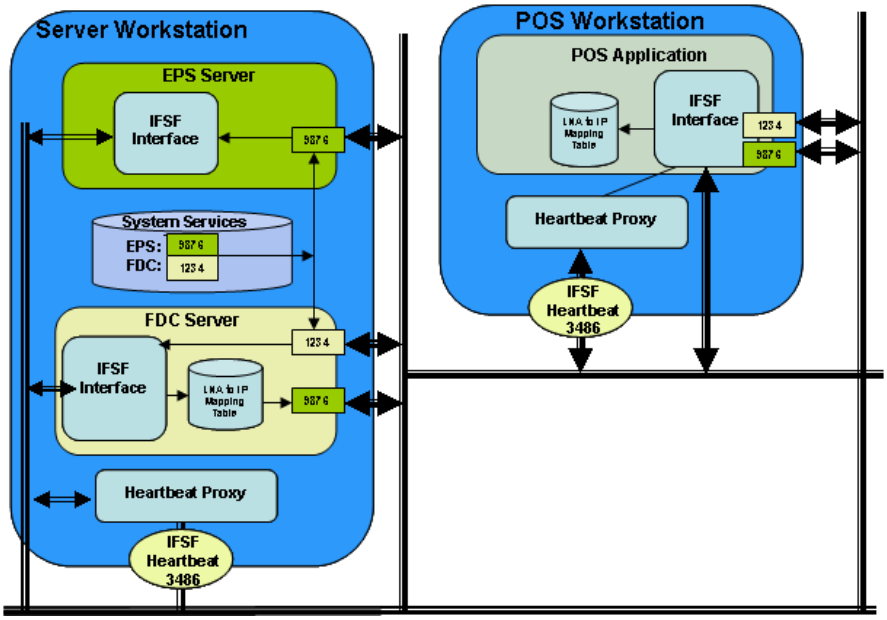
We recommend adding a line with

ifsf_fcserver 3487/tcp #IFSF Forecourt Server

to this file.

7.1.5 Exemplary Description of FDC Server Start-up

The picture below shows an example how the connection between a FDC Server and a POS client is established. In addition the picture shows an EPSToPOS interface (EPS Server) that might be managed in a similar way.



Port numbers:
1234: Server port of the FDC Server
9876: Server port of the EPS Server

When starting the FDC server evaluates the defined port number from the services table (=1234) and establishes a welcome socket on this port. After successful creation

of the socket the server signals his readiness by sending a broadcast message in the IFSF network by means of the local IFSF Heartbeat Proxy. The FDC server only must send an internal message to the heartbeat proxy to tell that the server now is ready for operation. Then the proxy now is sending the IFS formatted heartbeat message via the UDP port 3486 that is reserved for this purpose for all IFSF complient devices by IANA. This heartbeat message contains all information needed by clients to establish a connection to the server process. The structure of the message from the FDC server to the IFSF heartbeat proxy is free for a given implementation and not in the scope of this document. On the way back the heartbeat proxy is routing heartbeats from LAN connected IFSF devices and other high-level applications (e.g. EPS server) to the forecourt device controller.

When starting, the POS establishes a local connection to its IFSF Interface and IFSF Heartbeat Proxy and starts sending of heartbeat messages. Now the POS application is ready to receive heartbeats from any server. All incoming and outgoing heartbeat messages are routed through the IFSF Heartbeat Proxy and the IFSF Interface.

We assume that the POS application has received a heartbeat message from the FDC Server. This message contains the port number of the welcome socket of the server (1234) and the IP address of the Server Workstation. Now the POS application can establish a connection to the server which is in general accepted. From the subnet type received with the heartbeat message the POS application knows that all messages send to and received from the FDC Server are XML based and defined with the IFSF EPSToPOS interface specification. From now on the POS application is able to send a request of type ***Login*** to the server. This request contains the identification data of the application. The FDC server adds the POS application to its client table and sends finally a Login response to the POS application. The connection between FDC Server and POS Application is established.

7.1.6 Offline Detection

The standard IFSF procedure is used to detect offline situations. A given server application is set to offline status if no application heartbeat message was received for the interval as defined in the IFSF communication specification (e.g. 3 times 10 seconds). As already mentioned the heartbeat messages are send via UDP (broadcast!). If this feature is used each application is getting rid of the task to send individual heartbeat messages to each of its connected clients.
The FDC will log off all POS automatically in case of an offline detection. When a POS logs on to the FDC, the FDC stores the IP address of the POS together with the ApplicationSender ID and Workstation ID from the LogOn Request. If the FDC detects an offline condition by the IFSF heartbeat, which is just related to an IP address, all FDC clients from this IP address, e.g. POS applications, are logged off.

7.1.7 Application Ready

Because the IFSF heartbeat may be send from a different application than POS or FDC application and IFSF heartbeat is using just UDP not TCP, an additional life check method on TCP level is needed. It is common practise in TCP/IP application to

exchange a kind of empty application message to figure out dead applications or so called half dead TCP/IP sockets. For this purpose unsolicited messages, named FDCReady and POSReady ***(link to chapter)*** are defined. Both, FDC and POS must send these messages regularly. If these messages are not received in a given interval, the sender is assumed to be dead or the connection is broken. The FDC will automatically log off a POS in this case. Interval and numbers of repeats can be parameterized and must be consistent on the forecourt. The standard interval is 10 seconds, number of repeats 3 times. With this standard a broken connection is determined after 30 seconds.

7.1.8 Authentication

Because IT networks can be attacked easier compared with a LonWorks network, an authentication is defined with this standard. Using authentication ensures that messages can be interchanged just between trusted partners. The implementation of authentication is mandatory. This specification does not define encryption. If encryption is needed, the communication between FDC and POS can be established using IPSec or SSL which is an available feature in a couple of operating systems, including embedded operating systems. Even if SSL or IPSec is used, the mentioned authentication must be implemented. Using IPSec or SSL does not protect against untrusted applications, running on a POS or FDC.
Authentication is implemented by using a hash value on the payload of a message, based on a **passphrase** known by the FDC and the POS. For security reasons, the passphrase must be stored safe on the FDC and POS. There must be no means, to read the passphrase from one of the systems. To be open for future technical progress in hash algorithms, there is an ID for the used algorithm. Actually there is just one definition for MD5 128 Bit. The list of algorithms may be extended by IFSF engineering bulletins. Usage of authentication on a forecourt is optional and may be turned off. In this case an algorithm ID of zero is used. The hash field has a length of zero.

Hash Algorithm

ID	Type
0	No authentication
1	MD5 128bit

Authentication can be switched on and off, therefore implementation should decide how much security is required.
Authenication should be on all messages including Heartbeat Ready, but not on IFSF Heartbeat.
Client and Server use same authentication.

7.1.9 Frame Format

A message frame consists of three parts. It starts with the length of the payload. The length is a 32 bit value in network byte order. It is followed by the 128 bit MD5 hash for the payload. Third part of the message is the payload, which is a XML text.

length	Algorithm ID	Authentication hash	payload
32 bit network byte order	16 bit enumeration in network byte order	ID 0: 0 byte length (no authentication)	XML text
		ID 1: 128 bit length (MD5 128 bit)	
		Dependent of algorithm	

8 Device Classes

A device class describes a specific forecourt device and associated properties and events. Properties and events are defined by the IFSF standard regulations. The following classes are relevant for a standard interface definition of a forecourt device controller:

- COM - Communication entry point for each physical device; database for communication parameters.
- Fuel Dispenser (DSP)
- Fuelling Point (FP), sub device of DSP. One or more fuelling points are controlled by the same fuel dispenser.
- Tank Level Gauge (TLG), has sub devices of type TP, common database for all sub-ordinated tank probes.
- Tank Probe (TP); sub device of TLG.
- Price Pole (PP).
- Price Pole Point (PPP).
- Outdoor Payment Terminal (OPT)
- Car Wash (CW), sub device is CWP (Car Wash Point).
- Code Generating Device (CGD).

9 Device Hierarchy

Forecourt configuration information is stored in the configuration file(s) of the FDC application for each forecourt device. The forecourt controller divides forecourt devices in device classes. Not all classes are relevant for the POS. The following device classes are required in the POS: Dispenser (DSP), Fuel Points (FP), Tank Level Gauges (TLG), Tank Probes (TP), Price Poles (PP) and Price Pole Points (PPP). The FDC sends the essential information for each device class to the POS.

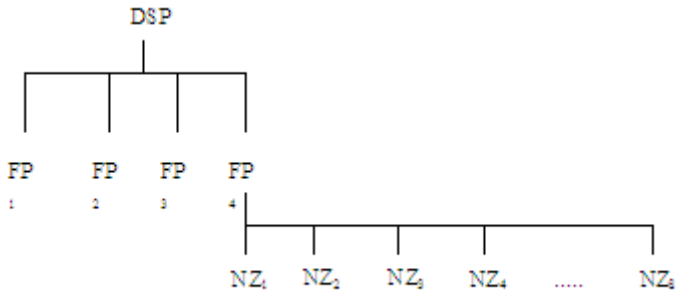
Devices are identified by a device ID. The device ID will be created, when the controller device starts. The Device ID is by device type across the whole forecourt.

9.1 Dispenser Hierarchy

A dispensing unit consists of one or more (maximum 4) *Fuelling Points*. A Fuelling point is 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”.

A dispenser device (DSP) controls fuelling points (FP). Each fuelling point has nozzles with a pure fuel product or a fuel product mix.

The XML structure for DSP, FP's and NZ's corresponds to the logical structure as following:



The structure will be repeated for each new dispenser. The pump number in a FP structure could be used to number FP's consecutively.

The fuel grade and possible fuel mix per nozzle is defined in the DSP meter ratio in the forecourt configuration file. The unique identifier for a FP is the Unit Number.

The following example shows Device ID numbering for two dispensers and fuelling points.

Dispenser 1 Device ID "1"		
	FP "1"	FP"2"
Device ID	"1"	"2"
Pump No	"1"	"2"

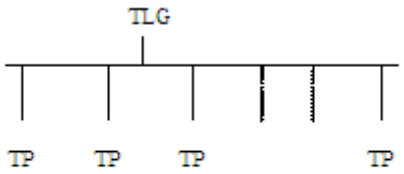
Dispenser 2 Device ID "2"		
	FP "1"	FP"2"
Device ID	"6"	"7"
Pump No	"3"	"4"

9.2 Tank Level Gauge Hierarchy

A Tank Level Gauge connects to one or more (maximum 31) *Tank Probes*. A Tank Probe is the item of forecourt equipment which is capable of measuring the level of product in a tank and hence volume can be calculated.

Each Tank Probe (TP) represents a tank. The unique identifier is the Unit Number. There are real tank probes and virtual tank probes. A virtual tank probe describes a tank without a physical probe installed.
Parent device of a TP device is a TLG.

The XML structure for a TLG and TP's' corresponds to the logical structure as following:



The structure will be repeated for each new tank level gauge. The tank number in a TP structure could be used to number TP's consecutively.

The following example shows Device ID numbering for two Tank Level Gauges and Tank Probes.

Tank Level Gauge 1 Device ID "1"			
	TP "1"	TP"2"	TP"3"
Device ID	"1"	"2"	"3"
Tank No	"1"	"2"	"5"

Tank Level Gauge 2 Device ID "2"		
	TP "1"	TP"2"
Device ID	"6"	"7"
Tank No	"3"	"4"

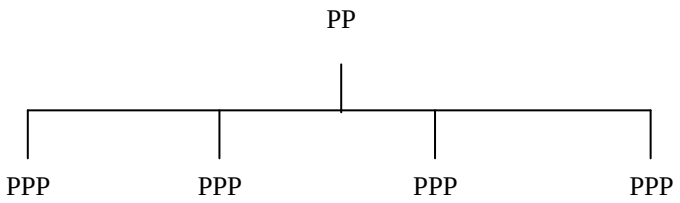
Very few sites have more than one Tank Level Gauge, nevertheless the above diagram shows the structure for two Tank Level Gauges.

9.3 Price Pole Hierarchy

A Price Pole connects to one or more (maximum 4) *Price Pole Points*. A Price Pole is a large display device which advertises product, services, goods, prices or general information.

A price pole device (PP) controls price pole points (PPP).

The XML structure for PP's' and PPP's corresponds to the logical structure as following:



The following example shows Device ID numbering for two Price Poles and Price Pole Points.

Price Pole 1 Device ID “1”		
	PPP “1”	PPP ”2”
Device ID	“1”	“2”
Sign No	“1”	“2”

Price Pole 2 Device ID “2”		
	PPP “1”	PPP ”2”
Device ID	“6”	“7”
Sign No	“3”	“4”

Very few sites have more than one Price Pole, nevertheless the above diagram shows the structure for two Price Poles.

10 FDC Configuration Data

FDC configuration data is stored in an IFSF XML Site Configuration file. When the FDC starts it reads the config file and the POS sends a GetConfigurationData.

11 Message Control and Synchronisation

Between the FDC application and the POS program information is interchanged. Requests are sent from the POS to the FDC application. The FDC application sends responses and unsolicited messages to the POS client.
The minimum a request consists of is the name of the request type, the application sender, a workstation ID and a request ID. Additioally, a timestamp has to be sent by the requestor. The request ID is a consecutive number inserted by the requestor.
The FDC application repeats the information received from the requestor and adds the desired information.
The requestor is responsible for synchronisation of the required data, the FDC application does not take care of message control and synchronisation of requests.
It is possible to query at the same time all devices of a given device class by using “*” (an asterisk) in the DeviceID attribute. In this case the response message will have a “<DeviceClass>” section for each DeviceID.

12 FDC Business Logic

Forecourt business logic implements functions which are not part of the IFSF standardisation. On the other hand it covers typical necessary forecourt functions. If the forecourt device control does not support these functions, each POS application has to implement them individually. There are no standard rules how to implement these functions. The XML interface must describe the required elements and attributes and the FDC business logic must process the appropriate logic.

Typical business logic for a FP is e.g. Self Service Mode with automatic pump release when the fuel transaction is paid, manual release at POS, automatic release by an OPT, prepayment mode, device and nozzle locking, automatic release with card authorisation or pay in the shop etc.
The kind of business logic supported by the FDC is described in more detail in use-cases.

13 Device Requests and Responses (Overview)

The following tables give an overview of requests from POS to FDC and unsolicited messages send from FDC to POS. The functionality of the requests will be described in more detail in a later chapter in this document.

Before a request or response will sent a four byte message length information must sent. The message length information must be stored in binary net byte order format.

13.1 Common Request / Response Attributes and Elements

The following elements and attributes are always used in each request and response:

Request Elements:

Name	Type	Use
POSdata	Complex	required
POSTimeStamp	String	required

Request Attributes:

Name	Type	Use
RequestType	RequestType	required
ApplicationSender	Application Type	required
WorkstationID	Workstation Type	required
RequestID	Request Type	required

Response Elements:

Name	Type	Use
FDCdata	Complex	required
FDCTimeStamp	String	required

Response Attributes:

Name	Type	Use
ResponseType	ResponseType	required
ApplicationSender	Application Type	required
WorkstationID	Workstation Type	required
ResponseID	Request Type	required
OverallResults	Result Type	required

13.2 Requests POS to FDC

POS request for service to FDC. Possible requests are identified by the XML attribute RequestType.

POS to FDC	FDC Response	Remark
LogOn	ACK or NACK	Logical POS sign on

POS to FDC	FDC Response	Remark
LogOff	ACK or NACK	Logical POS sign off
VersionInfo	Send release information or NACK	Provides the current FDC software release information
GetCountrySettings	Send country settings or NACK	POS asks for configured country settings
GetProductTable	Send product table or NACK	POS asks for current FDC product information
GetModeTable	Send mode-price table or NACK	POS asks for current service modes and corresponding prices
GetDSPConfiguration	Send forecourt configuration or NACK	POS asks for current Dispenser configuration
GetTLGConfiguration	Send forecourt configuration or NACK	POS asks for current Tank Level Gauge configuration
GetPPConfiguration	Send forecourt configuration or NACK	POS asks for current Price Pole configuration
GetDSPLimits	Send limits or NACK	Sends Volume and Money limits to POS
ChangeDSPLimits	ACK or NACK	POS want to change current DSP limits
ChangeFuelPrice	ACK or NACK	POS demands a fuel price change
GetFPFuelMode	Current fuel mode or NACK	POS requests the current fuel mode of a FP
GetPPPFuelMode	Current fuel mode or NACK	POS requests the current fuel mode of a PPP
ChangeFPFuelMode	ACK or NACK	POS demands fuel mode change for Fuel Points
ChangePPPFuelMode	ACK or NACK	POS demands fuel mode change for Price Pole Points
LockFuelSaleTrx	ACK or NACK	POS reserves a fuel sales transaction
UnlockFuelSaleTrx (unreserve)	ACK or NACK	POS unlocks a reserved fuel sales transaction
ClearFuelSaleTrx	ACK or NACK	POS clears a previously locked fuel sales transaction.
GetAvailableFuelSaleTrxs	Send list of available transactions or NACK	POS reads all available fuel sale transactions.
GetFuelSalesTrxDetails	Send transaction details or NACK	POS reads all details of fuel sale transaction.
SuspendFuelling	ACK or NACK	A running fuelling will be interrupted.
ResumeFuelling	ACK or NACK	A previously interrupted fuelling will be resumed.

POS to FDC	FDC Response	Remark
TerminateFuelling	ACK or NACK	The command stops a running fuelling at FP.
GetCurrentFuellingStatus	Send status of current fuelling or NACK	POS asks for intermediate values of current fuelling.
AuthoriseFuelPoint	ACK or NACK	Releases a selected fuel point.
StopForecourt	ACK or NACK	Software emergency stop of all forecourt devices
StartForecourt	ACK or NACK	Restart after software emergency stop
LockNozzle (FDC)	ACK or NACK	Lock a nozzle of a fuelling point
UnlockNozzle (FDC)	ACK or NACK	Unlock a nozzle of a FP
LockTank (FDC)	ACK or NACK	Locks a selected tank
UnLockTank (FDC)	ACK or NACK	Locks a selected tank
GetTankData	TP data or NACK	Current tank probe information
ReserveFuelPoint	ACK or NACK	Reserves a fuelling point for authorisation.
FreeFuelPoint (opposite to reserve)	ACK or NACK	Frees reserved fuelling point
GetFPState	ACK or NACK	POS ask for current state of a Fuelling Point
GetTPState	ACK or NACK	POS ask for current state of a Tank Probe
GetPPState	ACK or NACK	POS ask for current state of a Price Pole
GetTotals	Send totals or NACK	POS asks for dispenser totals
OpenDevice	ACK or NACK	Open a device.
CloseDevice	ACK or NACK	Close a device.

13.3 Unsolicited Messages FDC to POS

Unsolicited messages from FDC to POS are one-way information; sent from the FDC application to the POS. There is no preceding request sent from the POS. The POS application software usually must react on the FDC messages with individual routines.

FDC to POS	Remark
FDCReady	Used to check communications.
POSReady	Used to check communications.
DSPLimitsChange	Dispenser configuration change
FuelSaleTrx	FDC sends fuel sales transaction
FPStateChange	FDC sends Fuelling Point state change

FDC to POS	Remark
TPStateChange	FDC sends tank probe state change
PPStateChange	FDC sends price pole state change
FuelPriceChange	A fuel price was changed by FDC
FPModeChange	A Fuelling Point fuel mode was changed by FDC
PPPModeChange	A Price Pole Point fuel mode was changed by FDC
DeviceAlarm	FDC notifies a device alarm
FDCException	Exceptional message from FDC
FuelPointCurrentFuellingStatus	POS gets intermediate values of current fuelling

13.4 FDC Overall Results to the POS request

The "OverallResult" is used to report format errors in the request message or reasons why the request could not be executed. It is not used to report the out come of executing the request by the FDC. The result of executing a request by the FDC is reported in element "ErrorCode".

Following results are Possible:

OverallResult	Remark
Success	Complete Success.
PartialFailure	The request could not be executed completely.
Failure	Complete failure.
TimeOut	Complete failure. No response from FDC
FormatError	Complete failure. The request cannot process or is wrong formatted.
ParsingError	Complete failure. XML is not well formed.
ValidationError	Complete failure. XML is not validated against the defined schema.
MissingMandatoryData	Complete failure. Mandatory data are missing in the request.
WrongConfiguration	FDC has detected a wrong forecourt configuration.
NoLogon	The client has not performed log-on or log-on fails
AuthentificationError	Authentification error detected
MandatoryConfigurationDataMissing	Data missing that is needed before sales can be made e.g. one product must be defined.

13.4.1 Use of "OverallResult".

If the "OverallResult" is not "Success" then an error has been found in the "Request".

The following example shows an invalid "Request Type"

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="Rubbish" ApplicationSender="POSsell1"
```

```
 WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LogOn_Request.xsd" >
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

The “Response” message will be as follows:

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="Rubbish" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="FormatError"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LogOn_Response.xsd" >
</ServiceResponse>
```

13.5 Error Codes

The FDC application provides error codes to give more detailed information to the POS application. The element “ErrorCode” is used to report whether the request message is valid and the out come of executing a request message by the FDC.

Sending a request message can result in one of two out comes. These are, the FDC sends a response message only or the FDC sends a response message followed at some time by an unsolicited message. An unsolicited message is sent, whenever a request makes or attempts to make a change to a forecourt device.

If the FDC responds with only a response message the element “ErrorCode” is used to report whether the request message is valid (understood/ can be executed) and the out come of executing the request.

If the FDC responds with a response and unsolicited message the element “ErrorCode” in the response message is used to report whether the request message is valid (understood/ can be executed). The element “ErrorCode” in the unsolicited message is used to report the out come of executing the request.

An “Unsolicited” message alone is sent, when there has been a change in the configuration or state of a device.

The following error codes are specified:

Definition	Code	Description
ERRCD_OK	0	No error while action. Positive acknowledge: data acceptable. This is equivalent to Data_Ack equals 0 in the Communications Specification Part 2.1.
ERRCD_NO	1	No error, no action

Definition	Code	Description
ERRCD_CTRLERR		Forecourt controller error
ERRCD_COMMERR		Communication error
ERRCD_NOTSUP		Not supported option
ERRCD_READERR		Cannot read
ERRCD_WRITEERR		Cannot write. Not Writable - in that state (or any state) - read only data This is equivalent to Data_Ack equals 2 in the Communications Specification Part 2.1.
ERRCD_NOTPOSSIBLE		Action not possible. Command refused in that state. This is equivalent to Data_Ack equals 3 in the Communications Specification Part 2.1.
ERRCD_NOTALLOWED		Action not allowed. Command not accepted. Valid State and valid Command, but application rejects Command. There are a number of examples in Dispenser standard. This is equivalent to Data_Ack equals 6 in the Communications Specification Part 2.1.
ERRCD_INOP		FDC inoperable
ERRCD_DEVLOCK		Device locked
ERRCD_DEVOFFL		Device offline
ERRCD_BADVAL		Bad property value
ERRCD_NOPERM		Action not allowed
ERRCD_LIMITERR		Some limit exceeded
ERRCD_NOZZLELOCK		Referenced nozzle locked
ERRCD_TANKLOCK		Referenced tank locked
ERRCD_PREPAYERR		Prepayment not allowed
ERRCD_BADCONF		Bad forecourt configuration
ERRCD_NOTRANS		No fuel transaction in DSP transaction buffer
ERRCD_NORESTRANS		Fuel transaction not reserved for POS
ERRCD_TRANSLOCKED		Fuel transaction yet locked for another POS
ERRCD_INVTRANS		Fuel transaction invalid
ERRCD_DISPHWERR		No fuel transaction handling ok message at hardware
ERRCD_TIMEOUT		Fuel transaction timed out
ERRCD_UPDFAILED		Data update on forecourt hardware failed
ERRCD_NOMONEYPRES		Amount-Prepay not allowed
ERRCD_NOVOLUMEPRES		Volume preset not allowed
ERRCD_GENAUTHLIMIT		Maximum number of possible authorisations exceeded (global)
ERRCD_POSAUTHLIMIT		Maximum number of possible

Definition	Code	Description
		authorisations exceeded (per POS)
ERRCD_OTHER		Unspecified error
ERRCD_MAXSTACKLIMIT		Maximum number of unpaid transactions reached (result of failed authorization due to reached limit).
ERRCD_FPLOCK		Referenced Fuelling point locked
ERRCD_RESUMEFUEL		Error resume Fuelling
ERRCD_NODATA		No data to return e. g. request for all Trx , no transactons available
ERRCD_BADDEVID		Bad Device ID. Device ID does not exist.
ERRCD_BADTYPE		Bad Type. Type does not exist e.g. Type=”FFP”
ERRCD_DEVICEUNAVAILABLE		Request not possible, because device is unavailable, e.g. wrong device number
ERRCD_DEVICEDISABLED		Complete failure. Requested device is disabled.
ERRCD_WRONGDEVICENO		Complete failure. Requested device number is not available.

13.5.1 Use of “ErrorCode”.

If the “ErrorCode” is not “ERRCD_OK” then an error has been found in the execution of the “Request” or there is no data to return.

Case 1
The following example shows an invalid “Type”.

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="OpenDevice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FDC_OpenDevice_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FFP" DeviceID="1"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="OpenDevice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_OpenDevice_Response.xsd">
<FDCdata>
   <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
   <DeviceClass Type="FFP" DeviceID="1" >
         <ErrorCode>ERRCD_BADTYPE</ErrorCode>
   </DeviceClass>
```

```
  </FDCdata>
</ServiceResponse>
```

Case 2
The following example shows an invalid “Device Id”. “Device Id” 45 does not exist.

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="OpenDevice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FDC_OpenDevice_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="45"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="OpenDevice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_OpenDevice_Response.xsd">
<FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="45" >
          <ErrorCode>ERRCD_BADDEVICEID</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

13.6 Logical Device States

The FDC application must provide some logical device states to inform the POS application about the current device condition. Some device states are common for all forecourt devices. Logical device states depend on the forecourt device type. Fuel dispensers provide other states than price pole or tank probes. The following list gives an overview about logical device states.

State	Code	Description
Common Device States		
FDC_CONFIGURE		Configuration active (eg Pump initialisation in progress)
FDC_DISABLED		Describes a device status for a device that is available in a configuration but not active. It is temporary disabled.
FDC_ERRORSTATE		Error state
FDC_INVALIDSTATE		Invalid state (Device structure not completed)
FDC_OFFLINE		Offline (Physical connection down). Device switched off
FDC_OUTOFORDER		Out of order (e.g. the pump is switched off;

State	Code	Description
		inoperative)
Pump States		
FDC_CLOSED		Device cannot be used. The pump is in the Closed or Inoperative state.
FDC_READY		Ready (e.g. the pump is activated)
FDC_REQUESTED		Requested (The pump has release requested)
FDC_STARTED		Started (The pump motor is started)
FDC_FUELLING		Fuelling (Fuelling is active)
FDC_SUSPENDED_ STARTED		Suspended (The fuelling process is suspended)
FDC_SUSPENDED_ FUELLING		Suspended (The fuelling process is suspended)
FDC_CALLING		Device (usually pump) is calling, i.e. requiring authorisation from POS.
FDC_AUTHORISED		Fuelling Point is pre-authorised waiting for customer to select a valid logical nozzle.
Tank Probe States		
FDC_CLOSED		Device cannot be used. The tank probe is in the Closed or Inoperative state.
FDC_READY		Ready (e.g. the tank probe is activated)
Price Pole States		
FDC_CLOSED		Device cannot be used. The price pole is in the Closed or Inoperative state.
FDC_READY		Ready (e.g. the price Pole is activated)

Some Logical Device States are common to a number of devices e.g. FDC_READY, these common Logical Device State are listed under each device.

14 Requests POS to FDC (Detail Description)

Where the message format in both request and response messages allows multiple elements to be sent, applications must support this functionality.

14.1 LogOn

Description:
Each POS that wants to use the FDC application must itself log-on before being able to perform any operation. *Additionally, log-on initiates the application heartbeat between FDC and POS.*
The POS application first must open the TCP/IP connection for the static and dynamic channel. The connection must be held open by the POS until the LogOff command is sent by the POS or if the POS detects, it is offline to the FDC service.
Furthermore, the connection has to be closed, if the POS receives an FDCStopped message from the FDC application or get a socket closed error from the operating system.

A second log-on without a prior log-off is accepted every time (e.g. POS crashes and starts again).

Message Type	Schema Name
Request	FDC_LogOn_Request.xsd
Response	FDC_LogOn_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="LogOn" ApplicationSender="POSsell1"
 WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LogOn_Request.xsd" >
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <InterfaceVersion>1.0</InterfaceVersion>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="LogOn" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LogOn_Response.xsd" >
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <InterfaceVersion>1.0</InterfaceVersion>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</ServiceResponse>
```

14.2 LogOff

Description:
The request log-off disconnects a POS system from the FDC application. Used to terminate operations between the FDC and POS e.g. in case of configuration, administration, shut down, reconciliation etc.
A second log-off without a prior log-on is accepted every time.

Message Type	Schema Name
Request	FDC_LogOff_Request.xsd
Response	FDC_LogOff_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="LogOff" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LogOff_Request.xsd">
```

```
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="LogOff" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LogOff_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</ServiceResponse>
```

14.3 VersionInfo

Description

The POS asks for the current installed FDC software version.
If the request from the POS to the FDC application is successful, the FDC application process sends the response elements supplier, release, version and hot fix together with the common response information to the POS application.

Message Type	Schema Name
Request	FDC_VersionInfo_Request.xsd
Response	FDC_VersionInfo_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="VersionInfo" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_VersionInfo_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="VersionInfo" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_VersionInfo_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <InterfaceVersion>1.0</InterfaceVersion>
    <FDCsupplier>Wincor Nixdorf International</FDCsupplier>
    <FDCrelease>34.5 alpha</FDCrelease>
    <FDCversion>4.5.3</FDCversion>
    <FDChotfix>5.4</FDChotfix>
```

```
    <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</ServiceResponse>
```

14.4 GetCountrySettings

For country codes, currency codes, measurement units etc. common definitions by W3C or IFSF have to be used.

Description
The POS asks for the country settings of the FDC application. Country settings are configuration parameter of the forecourt controller device. They are used inside the Forecourt Device Controller to define country code, currency code, calculation factors, comma separator, measurement units etc.

Message Type	Schema Name
Request	FDC_GetCountrySettings_Request.xsd
Response	FDC_GetCountrySettings_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetCountrySettings" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetCountrySettings_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetCountrySettings" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetCountrySettings_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <VolumeUnit>Litres</VolumeUnit>
    <CurrencyCode>EUR</CurrencyCode>
    <LevelUnit>mm</LevelUnit>
    <TemperatureUnit>Cel</TemperatureUnit>
    <CountryCode>0044</CountryCode>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</ServiceResponse>
```

Country Code is same as Dispenser standard, either telephone code or ISO3166.

14.5 GetProductTable

Description:

The POS requests the pure fuel products used in the dispensers, fuelling points and tanks. The product number and name of a fuel product is described in the configuration files of the FDC application.
It is possible to mix fuel products in a fuelling point. The mix ratio is described in the DSP configuration parameters.

Message Type	Schema Name
Request	FDC_GetProductTable_Request.xsd
Response	FDC_GetProductTable_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetProductTable" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetProductTable_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceResponse RequestType="GetProductTable" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01423" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetProductTable_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <FuelProducts>
      <Product ProductNo="1000" ProductName="Super 98" />
      <Product ProductNo="2000" ProductName="Diesel" />
      <Product ProductNo="3000" ProductName="Normal" />
      <Product ProductNo="4000" ProductName="Super Mix96" />
    </FuelProducts>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</ServiceResponse>
```

The following example shows a request for a product table with no entries.

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetProductTable" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetProductTable_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceResponse RequestType="GetProductTable" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01423"
OverallResult="MandatoryConfigurationDataMissing"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetProductTable_Response.xsd">
```

14.6 GetModeTable

Description
The POS requests the fuel mode table from the FDC configuration data. Fuel modes are self service or full service, for example. A fuel price is related to a fuel product and a fuel mode; there may be different prices for a fuel product if the product is sold in different modes. The POS application usually creates its own item numbers for product/mode relations and assigns prices.

Message Type	Schema Name
Request	FDC_GetModeTable_Request.xsd
Response	FDC_GetModeTable_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetModeTable" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetModeTable_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceResponse RequestType="GetModeTable" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetModeTable_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <FuelMode ModeNo="1" ModeName="FullServ"/>
    <FuelMode ModeNo="2" ModeName="SelfServ"/>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</ServiceResponse>
```

14.7 GetDSPConfiguration

Description
The POS requests the Dispenser configuration using the message “GetDSPConfiguration”.

The IFSF standard provides fuel blending properties in the form of volume of product one and volume of product two. Some pump suppliers do not support this description but support a blend ration, two product numbers and a total volume. The blend ratio describes the relation between product one and two. A blended fuel sale will be mapped to a BlendProductNo. A blend product number has a description and price available.

Message Type	Schema Name
Request	FDC_GetDSPConfiguration_Request.xsd
Response	FDC_GetDSPConfiguration_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetDSPConfiguration" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetDSPConfiguration_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceResponse RequestType="GetDSPConfiguration" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetDSPConfiguration_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="DSP" DeviceID="1">
      <Product ProductNo="1000" ProductName="Normal">
        <FuelPrice ModeNo="1">1.999</FuelPrice>
        <FuelPrice ModeNo="2">1.995</FuelPrice>
      </Product>
      <Product ProductNo="2000" ProductName="GasOil">
        <FuelPrice ModeNo="1">1.799</FuelPrice>
        <FuelPrice ModeNo="2">1.775</FuelPrice>
      </Product>
      <Product ProductNo="3000" ProductName="Super98">
        <FuelPrice ModeNo="1">1.569</FuelPrice>
        <FuelPrice ModeNo="2">1.549</FuelPrice>
      </Product>
      <Product ProductNo="4000" ProductName="SuperPlus">
        <FuelPrice ModeNo="1">1.445</FuelPrice>
      </Product>
      <Product ProductNo="7000" ProductName="Mix95">
        <FuelPrice ModeNo="1">1.659</FuelPrice>
      </Product>
    <DeviceClass Type="FP"  DeviceID="1" PumpNo="1">
          <Nozzle NozzleNo="1">
           <ProductID ProductNo="1000" TankN1o="1"/>
          </Nozzle>
          <Nozzle NozzleNo="2">
```

```
        <ProductID ProductNo="2000" TankNo1="2"/>
       </Nozzle>
       <Nozzle NozzleNo="3">
        <ProductID ProductNo="3000" TankNo1="3"/>
       </Nozzle>
       <Nozzle NozzleNo="4">
        <ProductID ProductNo="4000" TankNo1="4"/>
       </Nozzle>
       <Nozzle NozzleNo="5">
        <ProductID ProductNo="7000" TankNo1="2" TankNo2="3" BlendRatio="50"/>
       </Nozzle>
    <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
    <ErrorCode>ERRCD_OK</ErrorCode>
   </DeviceClass>
   </FDCdata>
</ServiceResponse>
```

Blend Ratio applies to first tank.

Should it be necessary to define the length of time for a fuelling process, before the dispenser motor is stopped this element will be part of the FDC configuration. If different timeouts are required, then different Fuel Modes should be used.

14.8 GetTLGConfiguration

Description
The POS requests the Tank Level Gauge configuration using the message“GetTLGConfiguration”.
Because it is possible that a tank is not equipped with a TP an additional property ManualMode was inserted. ManualMode=”false” means there is a tank probe installed, “true” means there is no tank probe available. To achieve tank data if there is no tank probe installed manual dipping is necessary.

Message Type	Schema Name
Request	FDC_GetTLGConfiguration_Request.xsd
Response	FDC_GetTLGConfiguration_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetTLGConfiguration" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetTLGConfiguration_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceResponse RequestType="GetTLGConfiguration" ApplicationSender="POSsell1"
```

```
 WorkstationID="POS01" RequestID="01254" OverallResult="Success"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="FDC_GetTLGConfiguration_Response.xsd">
   <FDCdata>
     <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
<DeviceClass Type="TLG" DeviceID="1">

 <DeviceClass Type="TP" DeviceID="1" TankNo="1" ProductNo="1000"
 ProductName="Normal" ManualMode="false">
      <ShellCapacity>0</ShellCapacity>
      <MaxSafeFillCap>14500</MaxSafeFillCap>
      <LowCapacity>0</LowCapacity>
      <MinOperatingCapacity>0</MinOperatingCapacity>
      <TankManifoldPartners />
  <SetPoints>
       <HiHiLevel>0</HiHiLevel>
       <HiLevel>0</HiLevel>
       <LoLevel>0</LoLevel>
       <LoLoLevel>0</LoLoLevel>
       <HiWater>25</HiWater>
   </SetPoints>
     <ErrorCode>ERRCD_OK</ErrorCode>
     </DeviceClass>

<DeviceClass Type="TP" DeviceID="2" TankNo="2" ProductNo="2000"
 ProductName="GasOil" ManualMode="false">
    <ShellCapacity>0</ShellCapacity>
    <MaxSafeFillCap>14500</MaxSafeFillCap>
    <LowCapacity>0</LowCapacity>
    <MinOperatingCapacity>0</MinOperatingCapacity>
    <TankManifoldPartners />
<SetPoints>
    <HiHiLevel>0</HiHiLevel>
    <HiLevel>0</HiLevel>
    <LoLevel>0</LoLevel>
    <LoLoLevel>0</LoLoLevel>
    <HiWater>25</HiWater>
  </SetPoints>
    <ErrorCode>ERRCD_OK</ErrorCode>
   </DeviceClass>

<DeviceClass Type="TP" DeviceID="3" TankNo="3" ProductNo="3000"
 ProductName="Super98" ManualMode="false">
    <ShellCapacity>0</ShellCapacity>
    <MaxSafeFillCap>14500</MaxSafeFillCap>
    <LowCapacity>0</LowCapacity>
    <MinOperatingCapacity>0</MinOperatingCapacity>
 <TankManifoldPartners>
    <TankNo>4</TankNo>
     </TankManifoldPartners>
<SetPoints>
   <HiHiLevel>0</HiHiLevel>
   <HiLevel>0</HiLevel>
   <LoLevel>0</LoLevel>
   <LoLoLevel>0</LoLoLevel>
   <HiWater>25</HiWater>
 </SetPoints>
  <ErrorCode>ERRCD_OK</ErrorCode>
 </DeviceClass>
```

```
<DeviceClass Type="TP" DeviceID="4" TankNo="4" ProductNo="3000"
  ProductName="Super98" ManualMode="false">
   <ShellCapacity>0</ShellCapacity>
 <MaxSafeFillCap>14500</MaxSafeFillCap>
   <LowCapacity>0</LowCapacity>
   <MinOperatingCapacity>0</MinOperatingCapacity>
 <TankManifoldPartners>
    <TankNo>3</TankNo>
     </TankManifoldPartners>
  <SetPoints>
     <HiHiLevel>0</HiHiLevel>
     <HiLevel>0</HiLevel>
     <LoLevel>0</LoLevel>
     <LoLoLevel>0</LoLoLevel>
     <HiWater>25</HiWater>
  </SetPoints>
    <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>

   <ErrorCode>ERRCD_OK</ErrorCode>
   </DeviceClass>
 </FDCdata>
</ServiceResponse>
```

14.9 GetPPConfiguration

Description

The POS requests the Price Pole configuration using the message "GetPPConfigurations".

Message Type	Schema Name
Request	FDC_GetPPConfiguration_Request.xsd
Response	FDC_GetPPConfiguration_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetPPConfiguration" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetPPConfiguration_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceResponse RequestType="GetPPConfiguration" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetPPConfiguration_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
```

```
   <DeviceClass Type="PP" DeviceID="1">

     <DeviceClass Type="PPP" DeviceID="1" SignNo="1">
      <Segments>
          <Line SegmentNo="1" ProductNo="1000" ProductName="Normal" ProductPrice="1.999"
ModeNo="1">
          </Line>
          <Line SegmentNo="2" ProductNo="3000" ProductName="Super98" ProductPrice="1.479"
ModeNo="1">
          </Line>
          <Line SegmentNo="3" ProductNo="2000" ProductName="GasOil" ProductPrice="1.799"
ModeNo="1">
          </Line>
     </Segments>
    <ErrorCode>ERRCD_OK</ErrorCode>
      </DeviceClass>

    <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
 </FDCdata>
</ServiceResponse>
```

14.10 GetDSPLimits

Description

The POS requests the configured limits for each DSP (for all Fuel Points) There are volume limits, value limits and time limits. It is important for the POS application to know the adjusted limits in the FDC configuration to avoid wrong device requests, e.g. for pre-payment.

Request consists of the DeviceID alternatively set to '*' for all devices. Money setting must have two digits and volume three digits.

Message Type	Schema Name
Request	FDC_GetDSPLimits_Request.xsd
Response	FDC_GetDSPLimits_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetDSPLimits" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetDSPLimits_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="DSP" DeviceID="1"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetDSPLimits" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
 xsi:noNamespaceSchemaLocation="FDC_GetDSPLimits_Response.xsd">
 <FDCdata>
  <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
  <DeviceClass Type="DSP" DeviceID="1">
   <Product ProductNo="1000">
    <FuelMode ModeNo="1">
     <MaxTrxAmount>150.00</MaxTrxAmount>
     <MaxTrxVolume>80.000</MaxTrxVolume>
    </FuelMode>
    <FuelMode NodeNo="2">
     <MaxTrxAmount>150.00</MaxTrxAmount>
     <MaxTrxVolume>80.000</MaxTrxVolume>
    </FuelMode>
   </Product>
   <Product ProductNo="2000">
    <FuelMode ModeNo="1">
     <MaxTrxAmount>120.00</MaxTrxAmount>
     <MaxTrxVolume>70.000</MaxTrxVolume>
    </FuelMode>
    <FuelMode NodeNo="2">
     <MaxTrxAmount>100.00</MaxTrxAmount>
     <MaxTrxVolume>60.000</MaxTrxVolume>
    </FuelMode>
   </Product>
  <ErrorCode>ERRCD_OK</ErrorCode>
  </DeviceClass>
 </FDCdata>
</ServiceResponse>
```

14.11 ChangeDSPLimits

Description:
The POS wants to change the current limit values in a DSP database for a given product number and fuel mode.
Use (*) for all DSP in the same Request. Max money property must have two digits, max volume three digits.
The values of maximum amount and volume can be changed at any time. The new values will be applied to all new transactions.

If maximum amount or volume is left out there is no limit on the value.

An unsolicited “DSPLimitsChange” message will be sent by the FDC when the configuration has changed, thus confirming the request has been executed or when the limits cannot be changed.

In a multi POS configuration this will alert other POS’s’ to the change.

Message Type	Schema Name
Request	FDC_ChangeDSPLimits_Request.xsd
Response	FDC_ChangeDSPLimits_Response.xsd
Unsolicited	FDC_DSPLimitsChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="ChangeDSPLimits" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ChangeDSPLimits_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="DSP" DeviceID="1">
          <Product ProductNo=”1000”>
                <FuelMode ModeNo= “1”>
                      <MaxTrxAmount>180.00</MaxTrxAmount>
                      <MaxTrxVolume>120.000</MaxTrxVolume>
                </FuelMode>
          </Product>
    </DeviceClass>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceResponse RequestType="ChangeDSPLimits" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FDC_ChangeDSPLimits_Response.xsd">
  <FDCdata>
          <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
          <DeviceClass Type="DSP" DeviceID="1" >
             <ErrorCode>ERRCD_OK</ErrorCode>
          </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.12 ChangeFuelPrice

Description

The POS demands a fuel price change for a given fuel product and fuel mode.
A price change could be performed by date and time or immediately. Price changes will performed at once if the DateTime tag is empty. Fuel Price changes with DateTime in the past will be executed immediately
The FDC application makes sure that all connected price poles will be adjusted, following the legal regulations and all connected dispensers will be updated. If a dispenser is disconnected, the FDC application does the update later. The POS application must not supervise this process if it receives positive response from the FDC.

Synchronisation of clocks is performed by another system.

An unsolicited “FuelPriceChange” message will be sent by the FDC when the price change is complete or when the Price cannot be changed, thus confirming the request has been executed.

Message Type	Schema Name
Request	FDC_ChangeFuelPrice_Request.xsd

Response	FDC_ChangeFuelPrice_Response.xsd
Unsolicited	FDC_FuelPriceChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="ChangeFuelPrice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ChangeFuelPrice_Request.xsd">
<POSdata>
   <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
   <Product ProductNo="1000">
     <FuelMode ModeNo="1">
         <PriceNew>1.039</PriceNew>
         <EffectiveDateTime>2009-11-20T17:30:50</EffectiveDateTime>
     </FuelMode>
   </Product>
 </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="ChangeFuelPrice" ApplicationSender=" POSsell1"
 WorkstationID="CD-01" RequestID="073322" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ChangeFuelPrice_Response.xsd">
  <FDCdata>
   <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
   <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</ServiceResponse>
```

14.13 GetFPFuelMode

Description

This request returns current fuel mode for selected fuelling point.

Use "*" for all devices.

Message Type	Schema Name
Request	FDC_GetFPFuelMode_Request.xsd
Response	FDC_GetFPFuelMode_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetFPFuelMode" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetFPFuelMode_Request.xsd">
  <POSdata>
   <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
   <DeviceClass Type="FP" DeviceID="1"/>
  </POSdata>
```

```
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetFPFuelMode" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetFPFuelMode_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" PumpNo="1">
     <FuelMode ModeNo=”1”/>
     <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.14 GetPPPFuelMode

Description

This request returns current fuel mode for selected price pole point.

Use “*” for all devices.

Message Type	Schema Name
Request	FDC_GetPPPFuelMode_Request.xsd
Response	FDC_GetPPPFuelMode_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetPPPFuelMode" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetPPPFuelMode_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="PPP" DeviceID="1" SegmentNo="3" />
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetPPPFuelMode" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetPPPFuelMode_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
          <DeviceClass Type="PPP" DeviceID="1" SegmentNo="3" SignNo="1">
                <FuelMode ModeNo="1"/>
                <ErrorCode>ERRCD_OK</ErrorCode>
          </DeviceClass>
  </FDCdata>
```

```
</ServiceResponse>
```

14.15 ChangeFPFuelMode

Description
The POS demands a fuel mode change for a selected fuelling point (FP). The requested fuel mode must be known by the FDC and configured in the configuration procedure. If the requested mode does not exist, an error will be send to the POS application in the unsolicited. Fuel mode changes will be activated immediately or on completion of the current fuelling.

Use "*" to change all FP's.

An unsolicited "FPModeChange" message will be sent by the FDC when the fuel mode has changed, thus confirming the request has been executed executed or when the fuel mode cannot be changed.

Message Type	Schema Name
Request	FDC_ChangeFPFuelMode_Request.xsd
Response	FDC_ChangeFPFuelMode_Response.xsd
Unsolicited	FDC_FPModeChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="ChangeFPFuelMode" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FDC_ChangeFPFuelMode_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1">
          <FuelMode ModeNo="1"/>
    </DeviceClass>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="ChangeFPFuelMode" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ChangeFPFuelMode_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
          <DeviceClass Type="FP" DeviceID="1" >
             <ErrorCode>ERRCD_OK</ErrorCode>
          </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.16 ChangePPPFuelMode

Description
The POS demands a fuel mode change for a selected price pole point (PPP). The requested fuel mode must be known by the FDC and configured in the configuration procedure. If the requested mode does not exist, an error will be send to the POS application in the unsolicited message.

Use "*" to change all PPP's. The "SegmentNo" should also be set to "*".

An unsolicited "PPPModeChange" message will be sent by the FDC when the fuel mode has changed, thus confirming the request has been executed or when the fuel mode cannot be changed.

Message Type	Schema Name
Request	FDC_ChangePPPFuelMode_Request.xsd
Response	FDC_ChangePPPFuelMode_Response.xsd
Unsolicited	FDC_PPPModeChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceRequest RequestType="ChangePPPFuelMode" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ChangePPPFuelMode_Request.xsd">
        <POSdata>
                <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
                <DeviceClass Type="PPP" DeviceID="1" SegmentNo="3">
                        <FuelMode ModeNo="1"/>
                </DeviceClass>
        </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceResponse RequestType="ChangePPPFuelMode" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ChangePPPFuelMode_Response.xsd">
        <FDCdata>
                <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
                <DeviceClass Type="PPP" DeviceID="1" SegmentNo="3" >
                        <ErrorCode>ERRCD_OK</ErrorCode>
                </DeviceClass>
        </FDCdata>
</ServiceResponse>
```

14.17 LockFuelSaleTrx

Description
The payable fuel transactions will be send immedataly from the FDC application to the POS application through the dynamic communication channel.

The fuel transaction contains the necessary information for the POS (see Non Requested Messages FDC to POS – FuelSaleTrx).
The POS can reserve a desired fuel sales transaction for further processing. The assigned transaction number must be declared. The transaction will be locked for access by other POS systems. Next step is usually, the POS sets the transaction status to 'clear' or it unlocks the reserved transaction for use by another POS. Typically this happens when the wrong transaction was selected. Another POS can then reserve the fuel transaction.
If the transaction was paid by the POS it will be cleared in the dispenser database.
A Fuel transaction can be automatically locked, if that option was specified in AuthoriseFuelPoint command.
As long as the locking POS sends Heartbeats a locked transaction must not be unlocked or assigned to another POS.
When the locking POS stops sending Heartbeats the FDC must unlock the locked transactions and send an unsolicited message with the new transaction state (Cleared) to all POS 's', but not the off-line POS.

An unsolicited "FuelSaleTrx" message will be sent by the FDC when the state of the transaction has been changed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_LockFuelSaleTrx_Request.xsd
Response	FDC_LockFuelSaleTrx_Response.xsd
Unsolicited	FDC_FuelSaleTrx_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="LockFuelSaleTrx" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LockFuelSaleTrx_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" TransactionSeqNo="120301"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="LockFuelSaleTrx" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LockFuelSaleTrx_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
          <DeviceClass Type="FP" DeviceID="23" TransactionSeqNo="120301" >
            <ErrorCode>ERRCD_OK</ErrorCode>
          </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

An unsolicited message will also be sent.

The following example shows a response to a request to “Lock” a transaction that is already locked.

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="LockFuelSaleTrx" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LockFuelSaleTrx_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
          <DeviceClass Type="FP" DeviceID="23" TransactionSeqNo="120301" >
          <ErrorCode>ERRCD_OK</ErrorCode>
          </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

Unsolicited:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="FuelSaleTrx" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FuelSaleTrx_Unsolicited.xsd">
<FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" PumpNo="1" NozzleNo="1"
TransactionSeqNo="120301" >
      <State>Locked</State>
      <ReleaseToken></ReleaseToken>
      <FuelMode ModeNo=”2”></FuelMode>
      <Amount>76.96</Amount>
      <Volume>38.521</Volume>
      <UnitPrice>1.999</UnitPrice>
      <VolumeProduct1>38.52</VolumeProduct1>
      <VolumeProduct2></VolumeProduct2>
      <ProductNo1>1000</ProductNo1>
      <ProductNo2></ProductNo2>
      <BlendRatio>100</BlendRatio>
      <LockingApplicationSender>POSsell1</ LockingApplicationSender>
      <AuthorisationApplicationSender>POSsell1</ AuthorisationApplicationSender>
      <DSPFields>Field1 Field2 …Fieldn CheckSum</DSPFields>
      <CRCMode>ProcedureNo="1"</CRCMode>
      <ErrorCode>ERRCD_NOTPOSSIBLE</ErrorCode>
    </DeviceClass>
  </FDCdata>
</FDCMessage>
```

The following example shows a request to “Lock” a transaction that does not exist.

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="LockFuelSaleTrx" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LockFuelSaleTrx_Response.xsd">
```

```
 <FDCdata>
  <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
        <DeviceClass Type="FP" DeviceID="23" TransactionSeqNo="120301" >
         <ErrorCode>ERRCD_NOTRANS</ErrorCode>
        </DeviceClass>
 </FDCdata>
</ServiceResponse>
```

No unsolicited message will be sent by the FDC.

14.18 UnlockFuelSaleTrx (unreserve)

Description
The POS unlocks a previously reserved fuel sales transaction. This makes the transaction accessible to other POS systems again. A dispenser holds a number of transactions as long as they are not set to the status ‘clear’. Non paid transactions can block a corresponding fuelling point.
A transaction can be unlocked by POS that initially locked it unless it is offline, in which case any node can unlock a transaction.

An unsolicited “FuelSaleTrx” message will be sent by the FDC when the state of the transaction has been changed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_UnlockFuelSaleTrx_Request.xsd
Response	FDC_UnlockFuelSaleTrx_Response.xsd
Unsolicited	FDC_FuelSaleTrx_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="UnlockFuelSalesTrx" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_UnlockFuelSaleTrx_Request.xsd">
 <POSdata>
  <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  <DeviceClass Type="FP" DeviceID="23" TransactionSeqNo="120301"/>
 </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="UnlockFuelSaleTrx" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_UnlockFuelSaleTrx_Response.xsd">
 <FDCdata>
  <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
        <DeviceClass Type="FP" DeviceID="23" TransactionSeqNo="120301">
         <ErrorCode>ERRCD_OK</ErrorCode>
        </DeviceClass>
 </FDCdata>
```

```
</ServiceResponse>
```

14.19 ClearFuelSaleTrx

Description
The POS sets a fuel sales transaction to ‘clear’. The selected DSP transaction buffer will be cleared for a new transaction. The corresponding FP becomes able to process new fuel sales if the number of transactions is set to one.

An unsolicited “FuelSaleTrx” message will be sent by the FDC when the state of the transaction has been changed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_ClearFuelSaleTrx_Request.xsd
Response	FDC_ClearFuelSaleTrx_Response.xsd
Unsolicited	FDC_FuelSaleTrx_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="ClearFuelSaleTrx" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ClearFuelSaleTrx_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" TransactionSeqNo="120301" />
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="ClearFuelSaleTrx" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ClearFuelSaleTrx_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
          <DeviceClass Type="FP" DeviceID="23" TransactionSeqNo="120301">
            <ErrorCode>ERRCD_OK</ErrorCode>
          </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.20 GetAvailableFuelSaleTrxs

Description
POS asks for all payable or locked fuel sales transactions.

Use “*” in ID to request all transactions.

Message Type	Schema Name

Request	FDC_GetAvailableFuelSaleTrxs_Request.xsd
Response	FDC_GetAvailableFuelSaleTrxs_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetAvailableFuelSaleTrxs" ApplicationSender="POSsell1"
 WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetAvailableFuelSaleTrxs_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="23"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetAvailableFuelSaleTrxs" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetAvailableFuelSaleTrxs_Response.xsd">
<FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" PumpNo="1" TransactionSeqNo="120301" >
        <State>Locked</State>
        <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
    <DeviceClass Type="FP" DeviceID="23" PumpNo="1" TransactionSeqNo="120303" >
        <State>Payable</State>
        <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

The following example shows a response for all transactions on a fuelling point that has no transactions.

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetAvailableFuelSaleTrxs" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetAvailableFuelSaleTrxs_Response.xsd">
<FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" PumpNo="1">
        <ErrorCode>ERRCD_NOTRANS</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.21 GetFuelSalesTrxDetails

Description

The POS asks for details of transactions. This can be for a single transaction or all transactions on a fuelling point

Message Type	Schema Name
Request	FDC_GetFuelSalesTrxDetails_Request.xsd
Response	FDC_GetFuelSalesTrxDetails_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetFuelSaleTrxDetails" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetFuelSalesTrxDetails_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" TransactionSeqNo="*" />
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetFuelSaleTrxDetails" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetFuelSalesTrxDetails_Response.xsd">
<FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" PumpNo="1" NozzleNo=”1”
TransactionSeqNo="120301">
          <State>Payable</State>
          <ReleaseToken>01</ReleaseToken>
          <FuelMode ModeNo=”2”></FuelMode>
          <Amount>76.96</Amount>
          <Volume>38.521</Volume>
          <UnitPrice>1.999</UnitPrice>
          <VolumeProduct1>38.52</VolumeProduct1>
          <VolumeProduct2></VolumeProduct2>
          <ProductNo1>1000</ProductNo1>
          <ProductNo2></ProductNo2>
          <BlendRatio>100</BlendRatio>
          <LockingApplicationSender> POSsell1</ LockingApplicationSender>
          <AuthorisationApplicationSender> POSsell1</ AuthorisationApplicationSender>
          <DSPFields>Field1 Field2 …Fieldn CheckSum</DSPFields>
          <CRCMode>ProcedureNo="1"</CRCMode>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

Note: The CRC is not part of implementation and CRC implementation is country specific.
The dispenser configuration (data id “ZeroTR_Mode”) will determine whether or not zero value transactions are created. The FDC must make zero value transactions available to the POS.

The following example shows a response to a request for all transactions when there are none.

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetFuelSaleTrxDetails" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetFuelSalesTrxDetails_Response.xsd">
<FDCdata>
   <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
   <DeviceClass Type="FP" DeviceID="23" PumpNo="1" >
     <ErrorCode> ERRCD_NOTRANS </ErrorCode>
   </DeviceClass>
 </FDCdata>
</ServiceResponse>
```

14.22 SuspendFuelling

Description
The POS requests an interruption of a running filling. The FDC stops the corresponding fuelling point and sends a response message to the POS application.

An unsolicited “FPStateChange” message will be sent by the FDC when the device changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_SuspendFuelling_Request.xsd
Response	FDC_SuspendFuelling_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="SuspendFuelling" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_SuspendFuelling_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="SuspendFuelling" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_SuspendFuelling_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" >
```

```
   <ErrorCode>ERRCD_OK</ErrorCode>
   </DeviceClass>
   </FDCdata>
</ServiceResponse>
```

An unsolicited message will also be sent.

The following example shows a request to suspend fuelling when FP is in “Ready” state.

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="SuspendFuelling" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_SuspendFuelling_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="SuspendFuelling" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_SuspendFuelling_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" >
    <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
    </FDCdata>
</ServiceResponse>
```

Unsolicited:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="FPStateChange" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FPStateChange_Unsolicited.xsd">
  <FDCdata>
    <FDCTimeStamp> 2009-11-20T17:30:50 </FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" PumpNo="1">
      <DeviceState>FDC_READY</DeviceState>
      <LockingApplicationSender></ LockingApplicationSender>

      <Nozzle  NozzleNo=”1”>
        <LogicalNozzle>NozzleDown</ LogicalNozzle>
        <LogicalState>Unlocked</LogicalState>
        <TankLogicalState>Unlocked</TankLogicalState>
       <ErrorCode>ERRCD_OK</ErrorCode>
      </ Nozzle>

      <Nozzle NozzleNo=”2”>
```

```
      <LogicalNozzle>NozzleDown</ LogicalNozzle>
      <LogicalState>Unlocked</LogicalState>
      <TankLogicalState>Unlocked</TankLogicalState>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </ Nozzle>

    <Nozzle NozzleNo=”3”>
      <LogicalNozzle>NozzleDown</ LogicalNozzle>
      <LogicalState>Unlocked</LogicalState>
      <TankLogicalState>Unlocked</TankLogicalState>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </ Nozzle>

    <Nozzle NozzleNo=”4”>>
      <LogicalNozzle>NozzleDown</ LogicalNozzle>
      <LogicalState>Unlocked</LogicalState>
      <TankLogicalState>Unlocked</TankLogicalState>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </ Nozzle>

    <ErrorCode>ERRCD_NOTPOSSIBLE</ErrorCode>
   </DeviceClass>
  </FDCdata>
</FDCMessage>
```

14.23 ResumeFuelling

Description
The POS requests a restart of a suspended filling. The FDC tries to resume a suspended fuelling and sends a response with OverallResult ‘Success’ or an error code to the POS application if the command fails.

An unsolicited “FPStateChange” message will be sent by the FDC when the device changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_ResumeFuelling_Request.xsd
Response	FDC_ResumeFuelling_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="ResumeFuelling" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FDC_ResumeFuelling_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="ResumeFuelling" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ResumeFuelling_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" >
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.24 TerminateFuelling

Description
The POS requests a stop of a running filling for the selected fuelling point. If a wrong device ID was selected an error message is sent to the POS application. If there is no running fuelling at the selected FP no action will be taken.

An unsolicited “FPStateChange” message will be sent by the FDC when the device changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_TerminateFuelling_Request.xsd
Response	FDC_TerminateFuelling_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="TerminateFuelling" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_TerminateFuelling_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="TerminateFuelling" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_TerminateFuelling_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" >
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.25 GetCurrentFuellingStatus

Description
The POS requests intermediate values for transaction currently in progress. Authorisation Application Sender is similar to Release Control Id.

Message Type	Schema Name
Request	FDC_GetCurrentFuellingStatus_Request.xsd
Response	FDC_GetCurrentFuellingStatus_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetCurrentFuellingStatus" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetCurrentFuellingStatus_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="23"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetCurrentFuellingStatus"
ApplicationSender="POSsell1" WorkstationID="POS01" RequestID="01254"
OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetCurrentFuellingStatus_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" PumpNo="1" TransactionSeqNo="120305">
      <CurrentAmount>76.96</CurrentAmount>
      <CurrentVolume>38.521</CurrentVolume>
      <CurrentNozzle>1</CurrentNozzle>
      <CurrentUnitPrice>1.999</CurrentUnitPrice>
      <ReleaseToken></ReleaseToken>
      <AuthorisationApplicationSender> POSsell1</ AuthorisationApplicationSender>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.26 AuthoriseFuelPoint

Description
The POS releases a fuelling point controlled in manual mode. This command serves for all type of authorisations (postpay, prepay, OPT sale). The element 'ReleaseToken' is used to link the authorisation command to the related transaction data that will be sent with the FuelSaleTrx message (see FuelSaleTrx).

An unsolicited “FPStateChange” message will be sent by the FDC when the device changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_AuthoriseFuelPoint_Request.xsd
Response	FDC_AuthoriseFuelPoint_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="AuthoriseFuelPoint" ApplicationSender="POSsell1"
 WorkstationID="POS01" RequestID="01254"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="FDC_AuthoriseFuelPoint_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1">
      <MaxTrxAmount>50.00</MaxTrxAmount>
      <MaxTrxVolume></MaxTrxVolume>
      <FuelMode ModeNo="1"/>
    <ReleasedProducts>
      <Product ProductNo="1000"/>
      <Product ProductNo="4000"/>
    </ReleasedProducts>
      <ReleaseToken>01</ReleaseToken>
      <LockFuelSaleTrx>true</LockFuelSaleTrx>
    </DeviceClass>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="AuthoriseFuelPoint" ApplicationSender="POSsell1"
 WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_AuthoriseFuelPoint_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" >
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

Note: If “Release Product” is empty all products are authorised.

14.27 StopForecourt

Description

The POS requests all forecourt devices to be closed.
Running fuellings will not be interrupted unless “Emergency Stop” is true, in which case running fuellings will be stopped and it is not possible to continue (resume) the transaction.

After a “Stop Forecourt” the FDC cannot open devices, until it has received a “Start Forecourt”.

Message Type	Schema Name
Request	FDC_StopForecourt_Request.xsd
Response	FDC_StopForecourt_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="StopForecourt" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_StopForecourt_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <EmergencyStop>false</EmergencyStop>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="StopForecourt" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_StopForecourt_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</ServiceResponse>
```

14.28 StartForecourt

Description

The POS requests all closed devices to be opened. Devices already open will not change.

Message Type	Schema Name
Request	FDC_StartForecourt_Request.xsd
Response	FDC_StartForecourt_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="StartForecourt" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_StartForecourt_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="StartForecourt" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="FDC_StartForecourt_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</ServiceResponse>
```

14.29 LockNozzle

Description:
The POS requests a lock for a selected nozzle at a fuelling point.The nozzle is no longer available for operation. If the nozzle is locked already, the request will be refused and an error message will be reported.

An unsolicited “FPStateChange” message will be sent by the FDC when the nozzle changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_LockNozzle_Request.xsd
Response	FDC_LockNozzle_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="LockNozzle" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LockNozzle_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" NozzleNo="3"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="LockNozzle" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LockNozzle_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" NozzleNo="3">
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
    </FDCdata>
```

```
</ServiceResponse>
```

14.30 UnlockNozzle

Description:
The POS requests an unlock of a previously locked nozzle at a fuelling point. If the selected nozzle is not locked, no action is taken but an error will be reported to the POS.

An unsolicited "FPStateChange" message will be sent by the FDC when the nozzle changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_UnlockNozzle_Request.xsd
Response	FDC_UnlockNozzle_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

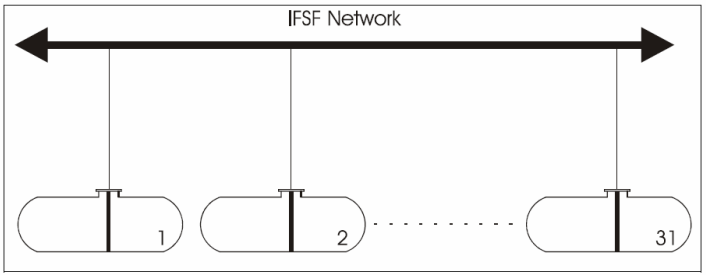
```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="UnlockNozzle" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_UnlockNozzle_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" NozzleNo="3"/>
    </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="UnlockNozzle" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_UnlockNozzle_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" NozzleNo="3">
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
    </FDCdata>
</ServiceResponse>
```

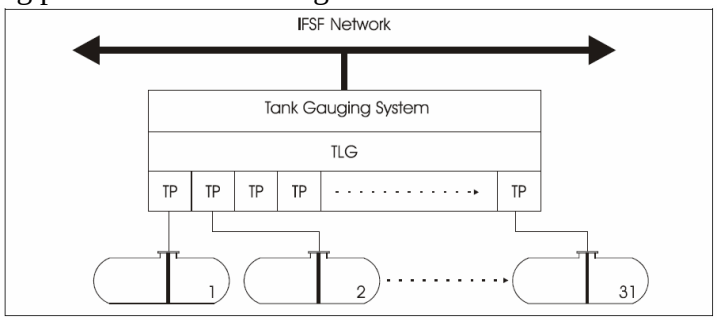
14.31 LockTank

Automatic tank data reading requires a tank probe installed in a tank. A tank probe can be direct linked to the IFSF network or each probe is connected to a tank level gauge (TLG).



The picture above shows an IFSF tank gauging system configuration were each tank probe (TP) is linked to the IFSF network.

Following picture shows a configuration with a TLG concentrator.



Tank data management must consider both configurations.

Description

The POS requests a lock for a selected tank. If the tank is locked already, the request will be refused with an error message. Locking a tank locks all dispenser nozzles linked to that tank.

An unsolicited “TPStateChange” message will be sent by the FDC when the tank probe changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_LockTank_Request.xsd
Response	FDC_LockTank_Response.xsd
Unsolicited	FDC_TPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="LockTank" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LockTank_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="TP" DeviceID="1" />
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="LockTank" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_LockTank_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="TP" DeviceID="1" >
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.32 UnlockTank

Description

The POS requests unlock of a previously locked tank. If the selected tank is not locked, no action will be taken but an error will be reported to the POS.

An unsolicited “TPStateChange” message will be sent by the FDC when the tank probe changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_UnlockTank_Request.xsd
Response	FDC_UnlockTank_Response.xsd
Unsolicited	FDC_TPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="UnlockTank" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_UnlockTank_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="TP" DeviceID="1" />
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="UnlockTank" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_UnlockTank_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="TP" DeviceID="1" >
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
```

```
</ServiceResponse>
```

14.33 GetTankData

Description
The POS requests the tank probe (TP) data for a selected tank. The FDC application provides the physical tank data from the FDC configuration file. If the TP is a real TP device and the FDC application can read the current measurement data from the TP, the read data will be sent in the XML tag <MeasuredData>. The device status will be set to "ERRCD_OK" if the data could be provided successful. A device error will be reported in the ErrorCode if the FDC service cannot reach the TP device.
The measured data provided by the TP varies from model to model.

Use (*) to request all TP's.

Message Type	Schema Name
Request	FDC_GetTankData_Request.xsd
Response	FDC_GetTankData_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetTankData" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetTankData_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="TP" DeviceID="1" />
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetTankData" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3_u111 ?rg/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetTankData_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="TP" DeviceID ="1" TankNo="1" ManualMode="False">
      <MeasurementData>
        <ProductLevel>1243</ProductLevel>
        <TotalObservedVolume></TotalObservedVolume>
        <GrossStandardVolume></GrossStandardVolume>
        <AverageTemp></AverageTemp>
        <WaterLevel></WaterLevel>
        <ObservedDensity></ObservedDensity>
      </MeasurementData>
    <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

Note: Tank Logical State can be “Locked” or “Unlocked”.

The following example shows a request for all tank probe data, but one tank cannot be reached.

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetTankData" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetTankData_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="TP" DeviceID="*" />
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="GetTankData" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi=" http://www.w3.org/2001/XMLSchema-instance "
xsi:noNamespaceSchemaLocation="FDC_GetTankData_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>

    <DeviceClass Type="TP" DeviceID ="1" TankNo="1" ManualMode="False">
      <MeasurementData>
      </MeasurementData>
    <ErrorCode>ERRCD_READERR</ErrorCode>
    </DeviceClass>

    <DeviceClass Type="TP" DeviceID ="2" TankNo="1" ManualMode="False">
      <MeasurementData>
        <ProductLevel>1243</ProductLevel>
        <TotalObservedVolume>1234</TotalObservedVolume>
        <GrossStandardVolume>1234</GrossStandardVolume>
        <AverageTemp>23</AverageTemp>
        <WaterLevel>2</WaterLevel>
        <ObservedDensity>1</ObservedDensity>
      </MeasurementData>
    <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>

  </FDCdata>
</ServiceResponse>
```

14.34 ReserveFuelPoint

A Fuelling Point (FP) is an 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”.

Description

A POS can lock (assign) a fuel point to itself to make sure no one else can authorise it. A “ReserveFuelPoint” can be sent to the FDC at any time.

As long as the POS that reserved the Fuel Point sends Heartbeats no other POS can authorise that Fuelling Point.
When the POS stops sending Heartbeats the FDC must free the Fuelling Point from the assignment.

An unsolicited “FPStateChange” message will be sent by the FDC when the device changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_ReserveFuelPoint_Request.xsd
Response	FDC_ReserveFuelPoint_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="ReserveFuelPoint" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ReserveFuelPoint_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="23"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="ReserveFuelPoint" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_ReserveFuelPoint_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" >
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.35 FreeFuelPoint

Description
POS frees fuelling point. Any other POS can now authorise it. . A “FreeFuelPoint” can be sent to the FDC at any time. An automatic FreeFuelPoint is issued by the FDC if it is detected that the POS was down.

An unsolicited “FPStateChange” message will be sent by the FDC when the device changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_FreeFuelPoint_Request.xsd
Response	FDC_FreeFuelPoint_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="FreeFuelPoint" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FreeFuelPoint_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="2"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="FreeFuelPoint" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FreeFuelPoint_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="2" >
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.36 GetFPState

Description

The POS asks for the current state of a fuelling point e.g. after a reboot of a POS or any other complication the POS is able to synchronize to FDC as soon as possible. Otherwise the POS has to wait for the next unsolicited state-message.

Message Type	Schema Name
Request	FDC_GetFPState_Request.xsd
Response	FDC_GetFPState_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetFPState" ApplicationSender="POSsell1"
 WorkstationID="POS01" RequestID="01254"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
 xsi:noNamespaceSchemaLocation="FDC_GetFPState_Request.xsd">
  <POSdata>
   <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
   <DeviceClass Type="FP" DeviceID="2"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType=" GetFPState " ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetFPState_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="2" >
      <DeviceState>FDC_READY</DeviceState>
      <LockingApplicationSender> POSsell1</ LockingApplicationSender>

      <Nozzle  NozzleNo=”1”>
        <LogicalNozzle>NozzleUp</ LogicalNozzle>
        <LogicalState>Unlocked</LogicalState>
        <TankLogicalState>Unlocked</TankLogicalState>
      </ Nozzle>

      <Nozzle NozzleNo=”2”>
        <LogicalNozzle>NozzleDown</ LogicalNozzle>
        <LogicalState>Unlocked</LogicalState>
        <TankLogicalState>Unlocked</TankLogicalState>
      </ Nozzle>

      <Nozzle NozzleNo=”3”>
        <LogicalNozzle>NozzleDown</ LogicalNozzle>
        <LogicalState>Unlocked</LogicalState>
        <TankLogicalState>Unlocked</TankLogicalState>
      </ Nozzle>

      <Nozzle NozzleNo=”4”>>
        <LogicalNozzle>NozzleDown</ LogicalNozzle>
        <LogicalState>Unlocked</LogicalState>
        <TankLogicalState>Unlocked</TankLogicalState>
      </ Nozzle>

      <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.37 GetTPState

Description

The POS asks for the current state of a tank probe e.g. after a reboot of a POS or any other complication the POS is able to synchronize to FDC as soon as possible. Otherwise the POS has to wait for the next unsolicited state-message.

Message Type	Schema Name
Request	FDC_GetTPState_Request.xsd
Response	FDC_GetTPState_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetTPState" ApplicationSender="POSsell1"
 WorkstationID="POS01" RequestID="01254"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="FDC_GetTPState_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="TP" DeviceID="2"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType=" GetTPState " ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetTPState_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="TP" DeviceID="2" />
       <DeviceState>FDC_READY</DeviceState>
       <TankLogicalState>Unlocked</TankLogicalState>
       <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.38 GetPPState

Description

The POS asks for the current state of a price pole e.g. after a reboot of a POS or any other complication the POS is able to synchronize to FDC as soon as possible. Otherwise the POS has to wait for the next unsolicited state-message.

Message Type	Schema Name
Request	FDC_GetPPState_Request.xsd
Response	FDC_GetPPState_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetPPState" ApplicationSender="POSsell1"
 WorkstationID="POS01" RequestID="01254"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="FDC_GetPPState_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
```

```
    <DeviceClass Type="PP" DeviceID="2"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType=" GetPPState " ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetPPState_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="PP" DeviceID="2" >
      <DeviceState>FDC_READY</DeviceState>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.39 GetTotals

Description

The POS asks for totals e.g. pump totals/ accumulators.

For all devices DeviceID and NozzleNo are set to “*”.

Message Type	Schema Name
Request	FDC_GetTotals_Request.xsd
Response	FDC_GetTotals_Response.xsd
Unsolicited	n/a

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="GetTotals" ApplicationSender="POSsell1"
 WorkstationID="POS01" RequestID="01254"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="FDC_GetTotals_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" NozzleNo="3"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType=" GetTotals" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_GetTotals_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" PumpNo="1" NozzleNo="3">
```

```
        <Amount>76.96</Amount>
        <Volume>38.521</Volume>
        <VolumeProduct1>1.999</VolumeProduct1>
        <VolumeProduct2>1.999</VolumeProduct2>
        <ProductNo1>1.999</ProductNo1>
        <ProductNo2>1.999</ProductNo2>
       <ErrorCode>ERRCD_OK</ErrorCode>
   </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

14.40 OpenDevice

Description
The POS can "Open" a device.

An unsolicited "FP or TP or PP StateChange" message will be sent by the FDC when the device changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_OpenDevice_Request.xsd
Response	FDC_OpenDevice_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="OpenDevice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FDC_OpenDevice_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="OpenDevice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_OpenDevice_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" >
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

An unsolicited message will also be sent.

The following example shows the response to “Open” a device that is in the “Open” state.

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="OpenDevice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_OpenDevice_Response.xsd">
 <FDCdata>
  <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
  <DeviceClass Type="FP" DeviceID="1" >
        <ErrorCode>ERRCD_OK</ErrorCode>
  </DeviceClass>
 </FDCdata>
</ServiceResponse>
```

Unsolicited:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="FPStateChange" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FPStateChange_Unsolicited.xsd">
 <FDCdata>
  <FDCTimeStamp> 2009-11-20T17:30:50 </FDCTimeStamp>
  <DeviceClass Type="FP" DeviceID="1" PumpNo="1">
   <DeviceState>FDC_READY</DeviceState>
   <LockingApplicationSender> POSsell1</ LockingApplicationSender>

   <Nozzle  NozzleNo=”1”>
    <LogicalNozzle>NozzleUp</ LogicalNozzle>
    <LogicalState>Unlocked</LogicalState>
    <TankLogicalState>Unlocked</TankLogicalState>
   <ErrorCode>ERRCD_OK</ErrorCode>
   </ Nozzle>

   <Nozzle NozzleNo=”2”>
    <LogicalNozzle>NozzleDown</ LogicalNozzle>
    <LogicalState>Unlocked</LogicalState>
    <TankLogicalState>Unlocked</TankLogicalState>
    <ErrorCode>ERRCD_OK</ErrorCode>
   </ Nozzle>

   <Nozzle NozzleNo=”3”>
    <LogicalNozzle>NozzleDown</ LogicalNozzle>
    <LogicalState>Unlocked</LogicalState>
    <TankLogicalState>Unlocked</TankLogicalState>
    <ErrorCode>ERRCD_OK</ErrorCode>
   </ Nozzle>

   <Nozzle NozzleNo=”4”>>
    <LogicalNozzle>NozzleDown</ LogicalNozzle>
    <LogicalState>Unlocked</LogicalState>
    <TankLogicalState>Unlocked</TankLogicalState>
    <ErrorCode>ERRCD_OK</ErrorCode>
   </ Nozzle>
```

```
    <ErrorCode> ERRCD_NOTPOSSIBLE </ErrorCode>
   </DeviceClass>
  </FDCdata>
</FDCMessage>
```

14.41 CloseDevice

Description
The POS can “Close” a device.

An unsolicited “FP or TP or PP StateChange” message will be sent by the FDC when the device changes state, thus confirming the request has been executed or whenever the state cannot be changed following a request to change state.

Message Type	Schema Name
Request	FDC_CloseDevice_Request.xsd
Response	FDC_CloseDevice_Response.xsd
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

Request:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceRequest RequestType="CloseDevice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FDC_CloseDevice_Request.xsd">
  <POSdata>
    <POSTimeStamp>2009-11-20T17:30:50</POSTimeStamp>
    <DeviceClass Type="FP" DeviceID="1"/>
  </POSdata>
</ServiceRequest>
```

Response:

```
<?xml version="1.0" encoding="utf-8" ?>
<ServiceResponse RequestType="CloseDevice" ApplicationSender="POSsell1"
WorkstationID="POS01" RequestID="01254" OverallResult="Success"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_CloseDevice_Response.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="1" >
          <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</ServiceResponse>
```

15 Unsolicited Messages FDC to POS

These messages from the FDC application to the POS will be sent without further response. The MessageID is numbered consecutively by the FDC application.

15.1 Ready Messages

Description
The FDC application sends a “Ready” message to the POS and vice versa to verify socket is open and working in both directions. The “Ready” message interval is by default set to ten seconds.

15.1.1 FDC Ready

Message Type	Schema Name
Unsolicited	FDC_FDC_Ready_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="FDCReady" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322">
  xmlns="http://www.nrf-arts.org/IXRetail/namespace"
  xmlns:IFSF="http://www.ifsf.org/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="FDC_FDC_Ready_Unsolicited.xsd">
  <FDCdata>
    <FDCTimeStamp>2004-20-12 08:39:09</FDCTimeStamp>
  </FDCdata>
</FDCMessage>
```

15.1.2 POS Ready

Message Type	Schema Name
Unsolicited	FDC_POS_Ready_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="POSReady" ApplicationSender="POSsel1"
WorkstationID="POS-01" MessageID="073322">
  xmlns="http://www.nrf-arts.org/IXRetail/namespace"
  xmlns:IFSF="http://www.ifsf.org/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="FDC_POS_Ready_Unsolicited.xsd">
  <POSdata>
    <POSTimeStamp>2004-20-12 08:39:09</POSTimeStamp>
  </POSdata>
</FDCMessage>
```

15.2 DSPLimitsChange

Description
The FDC application has changed the DSP Limits. This means, the POS must request the new limits data and update the forecourt device information held in the POS.

Message Type	Schema Name

Unsolicited	FDC_DSPLimitsChange_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="DSPLimitsChange" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_DSPLimitsChange_Unsolicited.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="DSP" DeviceID="1">
          <Product ProductNo=”1000”>
                <FuelMode ModeNo= “1”>
                      <MaxTrxAmount>180.00</MaxTrxAmount>
                      <MaxTrxVolume>120.000</MaxTrxVolume>
                </FuelMode>
          </Product>
      <ErrorCode>ERRCD_OK</ErrorCode>
     </DeviceClass>
  </FDCdata>
</FDCMessage>
```

15.3 FuelSaleTrx

Description

The FDC application sends a new fuel sale transaction to all connected and logged on POS systems. The element ‘ReleaseToken’ links the sale transaction data to the command that authorised it (see message AuthoriseFuelPoint).

The FDC application sends a FuelSaleTrx unsolicited message to all connected and logged on POS systems, when the state (Payable, Locked or Cleared) of a transaction changes or whenever the state cannot be changed following a request to change state.

To avoid manipulation of the fuel transactions a checksum is used (element “<CRC>”). To create the check sum an agreed encryption algorithm between fuelling point dispenser and POS is used. The dispenser fuel transaction consists of a number of fields in prescribed sequence order. The check sum is the last field in the sequence. A device that prints the fuel transaction must check the check sum and create an error if it differs from the dispenser check sum. (see Welmec Norm for Europe).

If W&M encryption is required in a country, the CCITT polynome is usually used as general key and the SEED as private key. For more information see the IFSF document ‘Standard forecourt protocol Part III.I, Dispenser Application Version 2.20 June 2004.
To calculate the W&M checksum, the POS must know the polynome and the SEED. These data are stored in the DSP database.

Message Type	Schema Name
Unsolicited	FDC_FuelSaleTrx_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="FuelSaleTrx" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FuelSaleTrx_Unsolicited.xsd">
<FDCdata>
   <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
   <DeviceClass Type="FP" DeviceID="23" PumpNo="1" NozzleNo="1"
TransactionSeqNo="120301" >
     <State>Locked</State>
     <ReleaseToken></ReleaseToken>
     <FuelMode ModeNo=”2”></FuelMode>
     <Amount>76.96</Amount>
     <Volume>38.521</Volume>
     <UnitPrice>1.999</UnitPrice>
     <VolumeProduct1>38.52</VolumeProduct1>
     <VolumeProduct2></VolumeProduct2>
     <ProductNo1>1000</ProductNo1>
     <ProductNo2></ProductNo2>
     <BlendRatio>100</BlendRatio>
     <LockingApplicationSender>POSsell1</ LockingApplicationSender>
     <AuthorisationApplicationSender>POSsell1</ AuthorisationApplicationSender>
     <DSPFields>Field1 Field2 …Fieldn CheckSum</DSPFields>
     <CRCMode>ProcedureNo="1"</CRCMode>
     <ErrorCode>ERRCD_OK</ErrorCode>
   </DeviceClass>
 </FDCdata>
</FDCMessage>
```

15.4 FPStateChange

Description
The FDC application sends a device status change for a specific fuelling point. Only the nozzle state for existing nozzles is returned.

Message Type	Schema Name
Unsolicited	FDC_FPStateChange_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="FPStateChange" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FPStateChange_Unsolicited.xsd">
 <FDCdata>
   <FDCTimeStamp> 2009-11-20T17:30:50 </FDCTimeStamp>
   <DeviceClass Type="FP" DeviceID="23" PumpNo="1">
     <DeviceState>FDC_READY</DeviceState>
     <LockingApplicationSender> POSsell1</ LockingApplicationSender>

     <Nozzle  NozzleNo=”1”>
       <LogicalNozzle>NozzleUp</ LogicalNozzle>
       <LogicalState>Unlocked</LogicalState>
       <TankLogicalState>Unlocked</TankLogicalState>
      <ErrorCode>ERRCD_OK</ErrorCode>
     </ Nozzle>
```

```
    <Nozzle NozzleNo=”2”>
      <LogicalNozzle>NozzleDown</ LogicalNozzle>
      <LogicalState>Unlocked</LogicalState>
      <TankLogicalState>Unlocked</TankLogicalState>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </ Nozzle>

    <Nozzle NozzleNo=”3”>
      <LogicalNozzle>NozzleDown</ LogicalNozzle>
      <LogicalState>Unlocked</LogicalState>
      <TankLogicalState>Unlocked</TankLogicalState>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </ Nozzle>

    <Nozzle NozzleNo=”4”>>
      <LogicalNozzle>NozzleDown</ LogicalNozzle>
      <LogicalState>Unlocked</LogicalState>
      <TankLogicalState>Unlocked</TankLogicalState>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </ Nozzle>

    <ErrorCode>ERRCD_OK</ErrorCode>
   </DeviceClass>
  </FDCdata>
</FDCMessage>
```

15.5 TPStateChange

Description
The FDC application sends a device status change for a specific tank probe.

Message Type	Schema Name
Unsolicited	FDC_TPStateChange_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="TPStateChange" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_TPStateChange_Unsolicited.xsd">
  <FDCdata>
    <FDCTimeStamp> 2009-11-20T17:30:50 </FDCTimeStamp>
    <DeviceClass Type="TP" DeviceID="1"  TankNo="1"/>
      <DeviceState>FDC_READY</DeviceState>
      <TankLogicalState>Unlocked</TankLogicalState>
      <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</FDCMessage>
```

15.6 PPStateChange

Description
The FDC application sends a device status change for a specific price pole.

Message Type	Schema Name
Unsolicited	FDC_PPStateChange_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="PPStateChange" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_PPStateChange_Unsolicited.xsd">
  <FDCdata>
    <FDCTimeStamp> 2009-11-20T17:30:50 </FDCTimeStamp>
    <DeviceClass Type="PP" DeviceID="1" >
     <DeviceState>FDC_READY</DeviceState>
     <ErrorCode>ERRCD_OK</ErrorCode>
    </DeviceClass>
  </FDCdata>
</FDCMessage>
```

Note: Price Pole state is at Price Pole level, not Price Pole Point level. See Price Pole standard.

15.7 FuelPriceChange

Description

The FDC application informs the POS application that a fuel price change was performed.

Message Type	Schema Name
Unsolicited	FDC_FuelPriceChange_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="FuelPriceChange" ApplicationSender="CD001"
          WorkstationID="CD-01" MessageID="073322"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="FDC_FuelPriceChange_Unsolicited.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <Product ProductNo=”1000” >
      < FuelMode ModeNo=”1”>
        <NewPrice>1.039</NewPrice>
       <OldPrice>1.022</OldPrice>
       <EffectiveDateTime>2009-11-20T17:30:50</EffectiveDateTime>
      </FuelMode>
    </Product>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </FDCdata>
</FDCMessage>
```

15.8 FPModeChange

Description

The FDC application informs the POS application that a fuelling point fuel mode change was performed.

Message Type	Schema Name
Unsolicited	FDC_FPModeChange_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="FPModeChange" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FPModeChange_Unsolicited.xsd">
 <FDCdata>
  < FDCTimeStamp>2009-11-20T17:30:50</ FDCTimeStamp>
  <DeviceClass Type="FP" DeviceID="23"  PumpNo="1">
    <FuelMode ModeNo=”1”>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </DeviceClass>
 </FDCdata>
 </FDCMessage>
```

15.9 PPPModeChange

Description
The FDC application informs the POS application that a price pole point fuel mode change was performed.

Message Type	Schema Name
Unsolicited	FDC_PPPModeChange_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="PPPModeChange" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_PPPModeChange_Unsolicited.xsd">
 <FDCdata>
  < FDCTimeStamp>2009-11-20T17:30:50</ FDCTimeStamp>
  <DeviceClass Type="PPP" DeviceID="23" SignNo="1" SegmentNo="3">
    <FuelMode ModeNo=”1”>
    <ErrorCode>ERRCD_OK</ErrorCode>
  </DeviceClass>
 </FDCdata>
 </FDCMessage>
```

15.10 DeviceAlarm

Description
The FDC application informs the POS application that an alarm was raised. An unsolicited message is sent whenever there is a change in state of an alarm. This means an unsolicited message is sent when an alarm is cleared.

Message Type	Schema Name
Unsolicited	FDC_DeviceAlarm_Unsolicited.xsd

XML Data (Example for TP alarm):

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="DeviceAlarm" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_DeviceAlarm_Unsolicited.xsd">
  <FDCdata>
     <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
     <DeviceClass Type="TP" DeviceID="1">
          <AlarmMsg Number="2" Text="Overfill"></AlarmMsg>
          <AlarmMsg Number="5" Text="High-High Level"></AlarmMsg>
    </DeviceClass>
  </FDCdata>
</FDCMessage>
```

15.11 FDCException

Description

The FDC application informs the POS application that an exception event has occurred. Exceptions can be used to handle special system events in the POS system. Typical exceptions are for example:

- Pump meter has been changed
- Pump has been operated in authark??? mode

Message Type	Schema Name
Unsolicited	FDC_FDCException_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="FDCException" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FDCException_Unsolicited.xsd">
  <FDCdata>
    <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <ExceptionNo>0001</ExceptionNo>
    <Description></Description>
  </FDCdata>
</FDCMessage>
```

Device identification will be identified in the “Description”.

At present it is not possible to define the exception events. To allow for standarisation at a later date “Exception Numbers” 0001 to 0500 are reserved for IFSF use.

15.12 FuelPointCurrentFuellingStatus

Description

For transaction in progress FDC will keep sending these unsolicited messages with current volume and amount.
To avoid this facility over loading the network, there will be a repetition timer element in the FDC configuration that controls the frequency of unsolicited messages. It should be possible to turn this facility off.

Message Type	Schema Name
Unsolicited	FDC_FuelPointCurrentFuellingStatus_Unsolicited.xsd

XML Data:

```
<?xml version="1.0" encoding="utf-8" ?>
<FDCMessage MessageType="CurrentFuellingStatus" ApplicationSender="CD001"
WorkstationID="CD-01" MessageID="073322"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FDC_FuelPointCurrentFuellingStatus_Unsolicited.
xsd">
  <FDCdata>
   <FDCTimeStamp>2009-11-20T17:30:50</FDCTimeStamp>
    <DeviceClass Type="FP" DeviceID="23" PumpNo="1" NozzleNo="1"
TransactionSeqNo="120301" >
      <ReleaseToken></ReleaseToken>
      <FuelMode ModeNo=”2”></FuelMode>
      <Amount>76.96</Amount>
      <Volume>38.521</Volume>
      <UnitPrice>1.999</UnitPrice>
      <VolumeProduct1>38.52</VolumeProduct1>
      <VolumeProduct2></VolumeProduct2>
      <ProductNo1>1000</ProductNo1>
      <ProductNo2></ProductNo2>
      <BlendRatio>100</BlendRatio>
    </DeviceClass>
  </FDCdata>
</FDCMessage>
```

16 Use cases

This section describes command flow for different types of fuelling.

16.1 Normal fuel sale (postpay)

- POS is waiting for *FDC_CALLING* pump state
- POS sends *AuthoriseFuelPoint* command to FDC
- POS is waiting for *FuelSaleTrx* or *DeviceStateChange* unsolicited message
- POS sends *LockFuelSaleTrx* command to FDC
- POS processes new fuel sale transaction
- POS sends *ClearFuelSaleTrx* command to FDC

16.2 Prepaid fuel sale (prepay)

It is assumed that there can be multiple POS units. One can authorise the prepay transaction and another can process it afterwards. Therefore it is not desirable to have the transaction automatically locked by FDC.

- POS sends *AuthoriseFuelPoint* command to FDC
- POS is waiting for *FuelSaleTrx* or *DeviceStateChange* unsolicited message
- POS sends *LockFuelSaleTrx* command to FDC
- POS processes new fuel sale transaction
- POS sends *ClearFuelSaleTrx* command to FDC

16.3 Outdoor Payment Terminal fuel sale (OPT-sale)

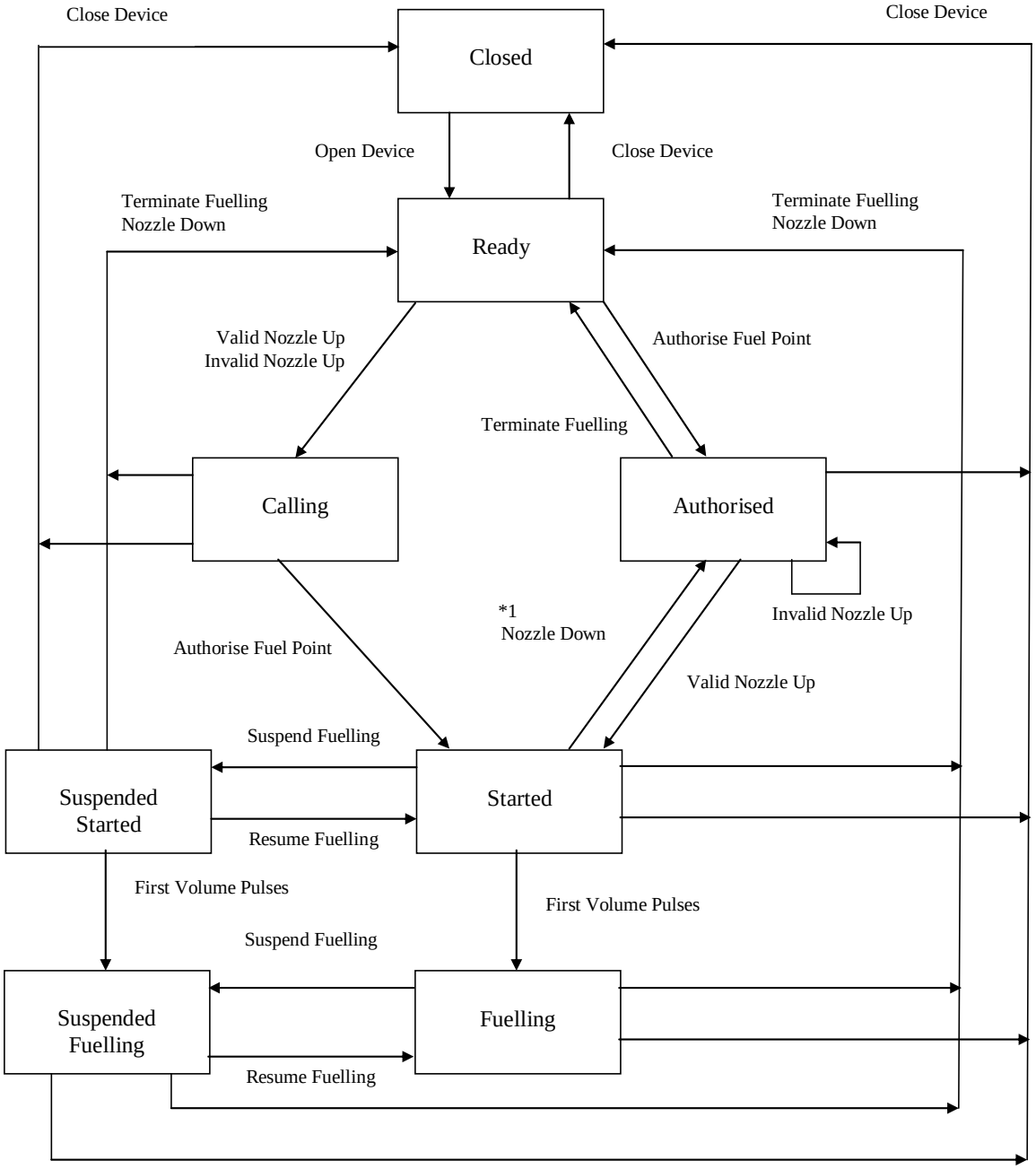
It is assumed that OPT transactions need to be automatically locked for authorizing unit to make sure that other POS units will not process the transaction by mistake.

- OPT sends *ReserveFuelPoint* command to FDC
- OPT sends *AuthoriseFuelPoint* command to FDC
- OPT is waiting for *FuelSaleTrx* or *DeviceStateChange* unsolicited message
- Transaction is automatically locked for OPT
- OPT processes new fuel sale transaction
- OPT sends *ClearFuelSaleTrx* command to FDC

17 State Diagrams

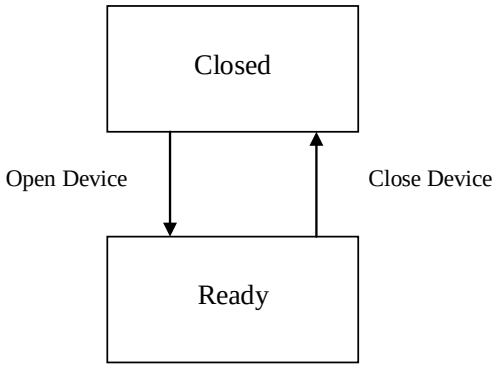
This section contains the State Diagrams for a Fuelling Point, Tank Probe and Price Pole presented by the FDC to the POS.

17.1 Fuelling Point Sate Diagram

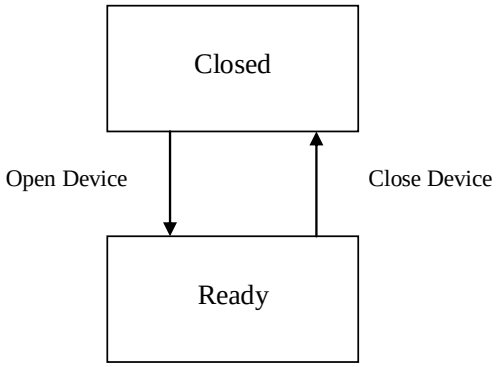


*1 Nozzle Down only moves to the Authorised state when Authorised state is allowed by configuration , otherwise it moves to Ready.

17.2 Tank Probe State Diagram



17.3 Price Pole State Diagram



18 Appendix

18.1 List of Elements

FDC

Element Name	Description
CurrencyCode	Currency code used in fuel transactions
DateTime	Requested Date and Time for PriceNew price change.
EffectiveDateTime	The date and time, when the price change took place.
ExceptionNo	Exception number created by FDC
FDChotfix	FDC hot fix release information.
FDCrelease	FDC software release information.
FDCsupplier	FDC software supplier.
FDCversion	FDC software version information.
CommunicationVersion	Communications version
CountryCode	*Country code*
LevelUnit	The units in which levels are reported eg mm
NewPrice	Price changed to NewPrice value.
OldPrice	Price before change.
PriceNew	Requested new Price.
TemperatureUnit	The units in which temperature is reported eg Centigrade, Fahrenheit.
VolumeUnit	The units in which volume is reported eg litres, gallons.

Common to Dispenser, TLG and PP

Element Name	Description
DeviceId	Identifies a specific device.
ManufacturerName	Manufacturer name.
DeviceModel	Model name.
DeviceType	Device Type.
ErrorType	*Error Type.*
Err_Description	*Error Description.*
ProdPrice	*Product Price.*
LogicalState	Logical device state
ModeName	Source in standards *Fuelling_Mode_Name.*
ModeNo	Derived from FM_ID.
ProductID	Product identification linked to a nozzle
ProdDescription	*Product Description.*
ProdNo	*Product Number.*
ProductPrice	Source in standards *Prod_Price*.
Type	Identifies a device type eg dispenser.

Dispenser

Element Name	Description
Amount	Source Dispenser standard *TR_Amount.*
BlendProductNo	Source Dispenser standard *PR_Id* in the Logical Nozzle Database.
BlendRatio	Source Dispenser standard *Meter_1_Blend_Ratio. An optional information about product mix, not provided by all dispensers.*
CurrentAmount	Source Dispenser standard *Current_Amount.*
CurrentNozzle	Source Dispenser standard *Current_Log_Noz.*
CurrentUnitPrice	Source Dispenser standard *Current_Unit_price.*
CurrentVolume	Source Dispenser standard *Current_Volume.*
FuelMode	Source Dispenser standard *TR_Fuelling_Mode.*
LockFuelSaleTrx	Causes a transaction to be locked to a POS.
MaxTrxAmount	Remote Amount Prepay also known as Max Auth Amount
MaxTrxVolume	Remote Volume Preset also known as Max Auth Volume
NozzleNo	Derived from Dispenser standard LN_ID.
ProductNo1	The Product Number is found via *Meter_1_Id*, then looking up Product Number (*PR_Id*) in the Meter Database.
ProductNo2	The Product Number is found via *Meter_2_Id*, then looking up Product Number (*PR_Id*) in the Meter Database.
VolumeProduct1	Source Dispenser standard *M1_Sub_Volume.*
VolumeProduct2	Source Dispenser standard *M2_Sub_Volume.*
ReleaseToken	Source Dispenser standard *Release_Token.*
ReleaseToken	Additional service information
State	Source Dispenser standard *Trans_State.*
TransactionSeqNo	Source Dispenser standard *TR_Seq_Nb.* Note value is 1 to 9999. See Dispenser standard.
Type	In transaction data??
UnitPrice	Source Dispenser standard *TR_Unit_Price.*
Volume	Source Dispenser standard *TR_Volume.*

Tank Level Gauge

Element Name	Description
AlarmMsg	Source TLG standard expansion of *TP_Alarm.*
AverageTemp	Source TLG standard *Average_Temp.*
GrossStandardVolume	Source TLG standard *Gross_Standard_Volume.*
HiHiLevel	Source TLG standard *HiHi_Level_Setpoint.*
HiLevel	Source TLG standard *Hi_Level_Setpoint.*
HiHiWater	
HiWater	Source TLG standard *Hi_Water_Setpoint.*
LoLoLevel	Source TLG standard *LoLo_Level_Setpoint.*

Element Name	Description
LoLevel	Source TLG standard *Lo_Level_Setpoint.*
LowFuel	Source TLG standard *Low_Capacity.*
ManualMode	The tank is not equipped with a TLG (True = no TLG, False=TLG installed)
MaxSafeFillCapacity	Source TLG standard *Max_Safe_Fill_Capacity.*
MinOperatingCapacity	Source TLG standard *Min_Operating_Capicity.*
ObservedDensity	Source TLG standard *Observed_Density.*
ProductLevel	Source TLG standard *Product_Level.*
ShellCapacity	Source TLG standard *Shell_Capacity.*
TankDiameter	Source TLG standard *Tank_Diameter.*
TankNo	Derived from TLG standard TP_ID.
TimeStamp	Source TLG standard *Current_Date*
TotalObservedVolume	Source TLG standard *Total_Observed_Volume.*
TPStatus	Source TLG standard *TP_Status*
WaterLevel	Source TLG standard *Water_Level.*

Price Pole

Element Name	Description
SegmentNo	
ManualMode	

18.2 Device Alarms and Errors

18.2.1 Alarm Messages

The Dispenser alarm messages are to be found in the Dispenser standard (version 2.32 January 2009 section 3.7 Data_ID 80).

The Tank Level Gauge alarm messages are to be found in the Tank Level Gauge standard (version 1.29 March 2008 section 3.5.1 Data_ID 33).

The Price Pole alarm messages are to be found in the Price Pole standard (version 1.23 July 2008 section 3.3 Data_ID 80).

18.2.2 Error States

The following device errors should be reflected in the “Error Codes”.

The Dispenser error states are to be found in the Dispenser standard (version 2.32 January 2009 section 3.10).

The Tank Level Gauge error states are to be found in the Tank Level Gauge standard (version 1.29 March 2008 section 3.6).

The Price Pole error states are to be found in the Price Pole standard (version 1.23 July 2008 section 3.8).

18.3 Appendix A - Schemas

A set of schemas is supplied with the standard.