PSS 5000

Application Note

Integrating 3rd Party Intelligent Terminal

Date December 15, 2009
Document number PSS5000/APNO/804680/00

Doms A/S Formervangen 28 DK-2600 Glostrup
Tel. +45 4329 9400 Fax +45 4343 1012
info@doms.dk www.doms.com

Table of Contents

1 Scope

Scope for this document is to describe how to build a system, where an intelligent payment terminal and a POS system from different manufactures can have a shared access to the pumps using Doms POS Protocol (DPP).

Some of the features described here require special DPP features/messages. These new features will be described in a coming release of Doms POS Protocol.

The term “Intelligent Payment Terminal” is here seen from the PSS point of view, meaning that the payment terminal intelligence (terminal application driving the user dialog and connection to the payment server or hosts) is outside the PSS.

With Intelligent Payment Terminals, PSS is isolated from the payment systems so payment related issues are outside the scope of this document.

If the terminal application is outside the PSS, it can be running at different places:

- One application in each terminal
- One or more applications (outside PSS) on a terminal server serving several terminals

The Terminal Application can be a separate application or integrated with a payment server application, again something that doesn’t relate to PSS and is outside the scope of this document.

Please note that all solutions where applications from different manufacturers must share a PSS need to be coordinated with the system integrator.

Building a solution with a Payment Terminal and a POS system from different manufactures, require coordination regarding the system architecture and functionality details.

It is a requirement, that the connected POS system has implemented the interface to the PSS in a way that allows multiple clients to be connected and access the pumps.

With this application note, we explain which areas need to be coordinated and show how the PSS can be used to do this.

Standard PSS application with support for Intelligent 3rd Party Terminals requires a **CPB509 boards** (PSS applications 411-xx-xxx).

2 References

Following references to external documents are used throughout this document.

[1] Doms POS Protocol, Chapter A

[2] Doms POS Protocol, Chapter B

[3] Doms POS Protocol, Chapter E

[4] Doms POS Protocol, Connection via TCP/IP

2.1 Abbreviations

Abbreviations used in this document.

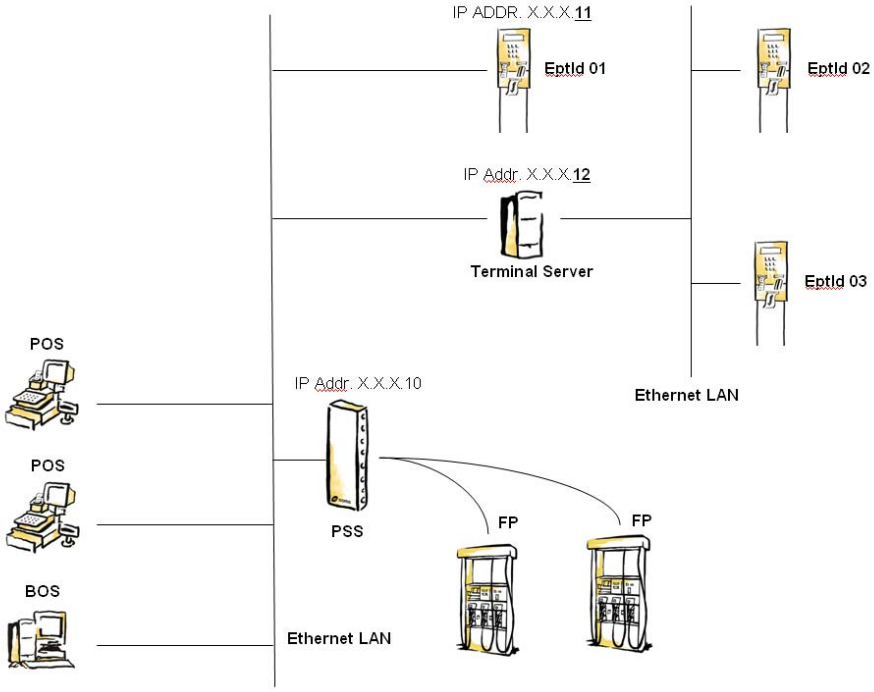
BOR	Back Office Record
CRIND	Card Reader In Dispenser
EPT	Electronic Payment Terminal Common name for all types or terminals, intelligent or unintelligent CRINDs and free standing terminals.
FP	Fuelling Point
LAN	Local Area Network
PSS	(Doms) Petro Site System
TCP/IP	Transmission Control Protocol / Internet Protocol
VAT	Value Added Tax

3 Overall Architecture

3.1 Diagrams

The diagram below shows the two ways a Payment Terminal can communicate to the PSS using Doms POS Protocol via Ethernet LAN:

- Directly
- Indirectly through a Terminal Server



3.2 System Design Issues

- Which information does the Terminal Application need from the Forecourt Controller to control its user interface and how can it get this information? This document describes which information PSS can provide and how.

- How to provide the Back Office System with information about the sales performed on the Payment Terminal. Doms Protocol provides a method, which can be used for this, and it is described in this document.

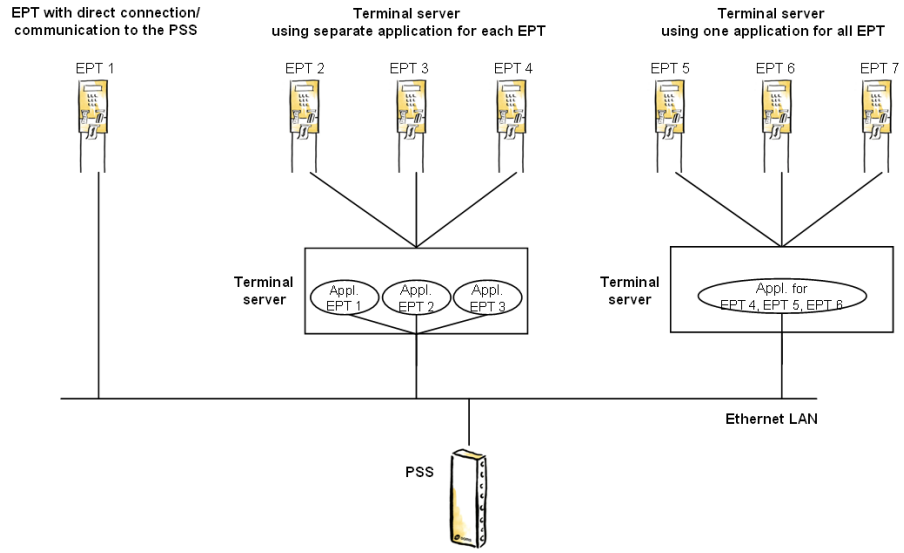
- Which information from the Terminal Application needs to forward to other applications, and how should the information get from the Terminal Application to these various applications processing the information? Doms POS Protocol provides a way to support this information flow.

If the system design process shows that something is missing, please contact Doms to discuss an enhancement to the interface.

4 Communication with PSS

4.1 Socket Handling

A Terminal Application can communicate either directly to the PSS or indirectly through a terminal server.



A terminal server will typically be built in one of two ways:

- One application per EPT it controls.
- One application that controls all EPTs.

The PSS supports both implementations, but the number of TCP/IP sockets needed is different in the two scenarios.

Terminal communicating directly with PSS

A terminal that connects and communicates directly to the PSS via TCP/IP must create a socket for each APC used in the Doms POS Protocol. Typically, a terminal application will need:

APC 1 / Port 5001 for solicited commands / requests
APC 2 / Port 5002 for unsolicited status messages, supervised operations
APC 5 / Port 5005 for unsolicited status messages, unsupervised operations

Number of sockets needed = (number of APCs) x (number of Terminals)

Terminal communicating indirectly with PSS through a Terminal Server

If the terminal does not communicate directly to the PSS, but communicates with a terminal server that communicates with the PSS, the terminal server can connect to the PSS in two ways:

1. Use separate sockets for each Terminal Application handled; one socket for each APC used in the Doms POS Protocol. Number of sockets needed = (number of APCs) x (number of Terminals)

2. Use one socket for each APC used in the Doms POS Protocol, and use these as common sockets for all the Terminals that the Terminal Server handles. Number of sockets needed = (number of APCs)

4.2 Logon to PSS

When a Terminal Application connects to PSS it starts the dialog by sending an *FcLogon2_req* message, which has some additional features compared to the basic *FcLogon* message.

4.2.1 Unsolicited Data from PSS

With *FcLogon2_req* message the client can list the status messages it wants unsolicited on the APC it sends the logon to. In the reply from PSS is a list holding the messages it has accepted to send unsolicited.

With this option, the unsolicited communication can be limited to the messages each client really is interested in. Limiting the unsolicited communication to what is really needed has a positive impact on system performance.

4.2.2 Application Identifier (APPL_ID)

For an Intelligent Terminal:

In the FcAccessCode parameter of the *FcLogon2_req* message, the Terminal Application must include an application identifier tag (APPL_ID) as described in document “Doms POS Protocol, Connection via TCP/IP” section FcLogon.

The APPL_ID binds together the EPT connections on different APC’s.

The APP_ID for a Terminal must follow the rule:

"APPL_ID=EPT_ + application number"

E.g. "APPL_ID=EPT_3" for Terminal Application with number 3.

For a POS:

Each POS must also logon with a unique APPL_ID and it must not start with EPT_. We recommend POS used “POS_1”, “POS_2” etc.

5 PSS Configuration

5.1 Install EPT

An Intelligent Payment Terminal must, as any other Payment Terminal, be installed as an EPT in the PSS using a Doms POS Protocol *install_Ept* message or with the use of the Doms PSS Configurator application.

The *install_Ept* message supports the following parameters:

EptId

EptId = the terminal number a customer will see.

For CRIND's, the EptId must be the same as the FpId it is linked to. From a customer point of view, the CRIND Terminal and Dispenser is one device with one ID, despite that, from a system point of view, it is two independent devices.

EptInterfaceType

This included flags to indicated

- Terminal is a CRIND
- CALL filter must be activated in PSS

EptSubDeviceId's

Not relevant for intelligent terminals.

PssChannelNo

PSS 5000 Port Number (41 = Ethernet port).

PhysicalAddress

NoEptDeviceSelectOptions

Specifies the FPs selectable from this terminal.

5.2 Operation Mode

5.2.1 FP Service Modes

For a CRIND, it might be relevant to see if the connected FP is configured for mix-mode (Attended and Unattended services) in current operation mode. This information can, for instance, be used to give the customer a choice to where he wants payment to be done: at CRIND or in the store.

The different service modes are described in DPP Chapter B.

FpStatus3 message holds such information. It shows a list of SmId's available in current FpOperationMode.

5.2.2 EPT ServiceTypes

The EptServiceTypes are equal to ServiceModes for Fps and controlled in the same way through the operation modes.

The actual EptServiceTypes are available via the EptStatus1 message.

5.3 EPT Timers

These timers are not relevant for Intelligent Payment Terminals. Configuration of the terminal application is not controlled via PSS.

5.4 EPT function setup parameters

These parameters are not relevant for Intelligent Payment Terminals. Configuration of the terminal application is not controlled via PSS.

6 Data Flow from PSS to EPT

Guidance to the data flow from PSS to EPT is described in this chapter.

Guidance to the data flow from EPT to PSS can be found in chapter 7.

6.1 Static information

6.1.1 Devices Installed

When the PSS is configured, a connected Terminal Application will receive a *FcInstallStatus* message unsolicited. The message holds a list of device IDs for each device type configured in the PSS.

The Terminal Application must check the list of installed EptIds and if the Terminal Application's own Id is represented in the list, the terminal has been configured in the PSS.

6.1.2 Grade ID's and Names

GradeIds can be read via *FcPriceSet_req* message (DPP Chapter A).

GradeNames can be read via *FcParameter_req* (DPP Chapter A).

6.1.3 VAT rate

The VAT rate can be changed with the *change_FcParameters* command and collected from the PSS using *FcParameterSet_req* command (DPP Chapter A).

A VAT rate sequence number is added to the *FcStatus* message, so that the clients have a way to know when the VAT rate has been updated.

6.1.4 Receipt Header and Footer

The receipt header and footer can be changed with the *change_FcParameters* command and collected from the PSS using *FcChangeParameters_req* command (see DPP Chapter A).

6.1.5 PosId and EptDeviceSelectOptions

The terminal application will get its PosId and a list of assigned pumps can be read via the new EptInstallData_req message.

6.2 Dynamic information

6.2.1 FC Date and Time

It might be required that the Terminal Application synchronizes its time with the *FcDateAndTime* set by the POS system.

The Terminal Application can read the current date and time from the Forecourt Controller using *FcDateAndTime_req*.

FcStatus2 provides additional parameters for RTC synchronization:

- RTC sequence number; incremented each time the date/time is changed.
- RTC date and time; the new date and time.

6.2.2 End of Shift / Shift No

It might be helpful to use the PSS to synchronize the shifts between the different POS and EPT clients in the system.

The ShiftNo can be changed in the PSS using the *FcChangeParameters_req* or the *FcAux_cmd*.

Connected clients will get ShiftNo unsolicited via the *FcStatus2* message.

6.2.3 Exchange Rates

The PSS has the possibility to receive exchange rates from one client and make them available to other clients (e.g. a Bank Note Acceptor)

See DPP Chapter E.

6.2.4 Grade Prices

The grade prices used on the pumps are controlled from an application other than the Terminal Application. But, the Terminal Application might need to know the prices for its user dialog.

Whenever there is a change in the PriceSet, all connected clients will be informed via the unsolicited message *FcPriceSet_status* (DPP Chapter A).

Clients who need to know the general grades and prices can read the PriceSet using *PriceSet_req* (DPP Chapter A).

As the current FP prices can be different from the general prices, the client has the option to request these by using *FpInfo1_req*.

It’s also possible for the client to lock FP prices (see section 7.3.1) to ensure that they do not suddenly change due to e.g. a general price update.

6.2.5 Pump Status from PSS to Payment Terminal

The requirement to the dialog on the EPT has a direct impact on the requirements to the status information for each Fuelling Point.

Some of the FP status information available in the latest FpStatus message is:

- The current FP state (Closed, Idle, Call, Error, etc.)
- Various sub-status flags informing about transaction buffer availability, if FPs are locked, if FP prices are locked, and more…
- Which grade has been selected (if any)
- The grades currently available
- The services currently available (service modes)

FpStatus3 message (DPP Chapter B) describes the information currently available, but the interface can be extended if needed.

7 Data Flow from EPT to PSS

Guidance to the data flow from EPT to PSS is described in this chapter.

Guidance to the data flow from PSS to EPT can be found in chapter 6.

7.1 Static information

If the Terminal Application can provide information about its software and hardware, the PSS can make this information available to other applications.

write_EptDeviceData (**new**) can be used by a Terminal Application to send such information to the PSS.

The PSS e.g. can forward the information to a host application via Doms Host Protocol.

7.1.1 Hardware Configuration

Examples of EPT hardware configuration data are:

- Maker
- Model
- Serial No

7.1.2 Software Configuration

Examples of EPT software configuration data are:

- Protocol Version(s)
- Software Version(s)

7.2 Dynamic information

A Terminal Application can have different types of status information, which can be directly needed or of interest for other applications.

These are sent to the PSS in the *write_EptDeviceStatus* (**new**) message. This message is based on parameter ID lists, so it is expandable if the need should arise.

The parameter list consists of:

- Service Status
- Warning Status
- Error Status
- Info Events (future implementation)

Normally, the Terminal Application sends only the parameters in which there are changes. When a parameter is included in the list, all active elements of that status parameter must be included. If, for instance, only a “Paper Low” warning was active and the “Temperature too High” warning suddenly arises,

both “Paper Low” and “Temperature too High” must be included in the parameter list holding all active warnings.

A parameter included, but with an empty list of elements, means that there are no active elements of that type (services, warnings or errors).

Sometimes a full status update is needed; this means that all the parameters of the *write_EptDeviceStatus* message must be included. This happens when:

- Terminal Application goes from offline to online
- PSS sends the *write_EptDeviceStatus_ack* unsolicited (used by the PSS to trigger a full status update)
- EPT is (re-)configured in the PSS

7.2.1 Service Status

Each supported service provides information about its current availability. If included in the list, is available.

- Printing Service available
- Card Payment Service available
- BNA Service available

7.2.2 Warning Status

The Terminal Application must report all the current active warnings to the PSS.

- Paper low
- Housing Open
- EMV housing open (idea for future)
- Temperature too high (idea for future)
- Temperature too low (idea for future)

Warnings can be added if required in specific projects.

7.2.3 Error Status

The PSS supports the error messages from the Terminal Application as defined in DPP Chapter E.

The terminal must report all the current active errors to the PSS.

- Printer error
- Display error
- Card Server Offline
- PIN Pad offline
- Card Reader offline
- Printer offline

Error status can be added if required in specific projects.

7.2.4 Info Events

This is an idea for **future** implementation and the content of the information events has not been decided yet.

7.2.5 Asynchronous Back Office Records

7.2.5.1 Storing Back Office Records

The PSS provides a message that can be used from clients (such as a terminal application to store asynchronous BORs):

NAME: store_BackOfficeRecord
EXTC: 0701H
SUBC: 00H
DATA: BorClientType +
BorClientId +
BorDataType
BorDataLen +
BorDataLen
{
BorData
}

The BorData is transparent data, and the PSS does not interpret the data.

The size of the transparent data must be determined for the single PSS application.

7.2.5.2 Reading Back Office Records

7.2.5.2.1 Method for new developments

Back office records can either be read via new_bor.xml (Doms Host Protocol, Basic Version), which will contain all records not previously read

or

one record at the time using Doms POS Protocol, Chapter A BackOfficeRecord_req (79 02):

NAME: BackOfficeRecord
CODE: F9
SUBC: 02
DATA: BorSeqNo +
BorFormatId +
BorDataLen +
BorDatalen
{
BorData
}

Here the BorFormatId will be 5 and the BorData is specified in new_bor.xsd.

7.2.5.2.2 Method for backward compatibility

In some existing solutions, Back Office Records are collected via 79 01 and here the client BorData is read in the following way:

Reply to 79 01

NAME: BackOfficeRecord
CODE: F9
SUBC: 01
DATA: BorSeqNo +
BorLength +
BorFormatId +
NoBorFields +
NoBorFields
{
BorFieldLength +
BorFieldType +
BorParId +
BorPar
}

The mapping between storing and reading will be as follows:

BorFormatId = 51 * from DPP Client via store_BackOfficeRecord"

NoBorFields = 4

BorFieldType	BorParId	BorPar
01	01	ClientType
01	02	ClientId
01	03	BorDataType
01	04	BorData

7.3 Commands

7.3.1 Reserve FP

To reserve a FP for the next transaction the Terminal Application must send the *extended_prepare_trans* message with the message parameter PosId different from ZERO.

With this message it is possible to:

- Lock the FP to a specific PosId
- Restrict the allowed grades, e.g. to one grade.
- Lock the prices on the FP

7.3.2 Authorize a Bank Note Transaction

To authorize an FP for a bank note payment the Terminal Application sends the *extended_authorize_Fp* message with SmId = 61.

Alternatively, the EPT can send the SmId=61 and other transaction setup parameters using *extended_prepare_trans*, and authorize the FP using *authorize_FP* message.

It might be useful to inform the PSS to automatically lock the transaction when it finalizes by using the AutoLock functionality (AuthParId 08).

If the PSS controls the systems W&M Payment Log (depends on PSS LAM Parameters), e.g. using PSS Memory Modules:

1. the authorization for BNA transactions MUST include LogData (AuthParId 07). For BNA transactions this is PrepaidMoney.
2. EptSeqNo must be printed on the receipt (e.g. part of receipt-no) as a reference to PSS Payment Log.

EptSeqNo is available as PaymentControlParameterId 12 for unsupervised transactions. Please notice that PaymentControlParameters (Chapter B) = EptSeqDataItems (Chapter E).

7.3.3 Authorize a Card Transaction

To authorize an FP for a card payment the Terminal Application must send the *extended_authorize_Fp* message with SmId = 51.

Alternatively, the Terminal Application can send the SmId=51 and other transaction setup parameters using *extended_prepare_trans*, and authorize the FP using *authorize_FP* message.

It might be useful to inform the PSS to automatically lock the transaction when it finalizes by using the AutoLock functionality (AuthParId 08).

If the PSS controls the systems W&M Payment Log (depends on PSS LAM Parameters), e.g. using PSS Memory Modules:

1. the authorization for card transactions must include LogData (AuthParId 07). For Card Transaction it must be CardTrackTwo and optional CardAuthCode. Track2 is included to provide a search facility, but where PCI DSS must be considered, full track data should not be sent. Send all 0's or hide some of the nipples with 0's.
2. EptSeqNo must be printed on the receipt (e.g. part of receipt-no) as a reference to PSS Payment Log.

EptSeqNo is available as PaymentControlParameterId 12 for unsupervised transactions. Please notice that PaymentControlParameters (Chapter B) = EptSeqDataItems (Chapter E).

7.3.4 Cancel Reservation

A FP reservation is cancelled by sending the *cancel_Fp_auth_and_setup* message (DPP Chapter B).

7.3.5 Cancel Authorization

An FP authorization is cancelled by sending the *cancel_Fp_auth_and_setup* message (DPP Chapter B).

7.3.6 Read Transaction Data

All connected clients will be notified[1] when a transaction has been created and it might be locked to a specific client (e.g. an EPT).

The relevant Terminal Application can read the transaction and do the required processing, e.g. deliver it to a Payment Server and print a receipt to the customer.

7.3.7 Clear Transaction Data

When the Terminal Application has processed the transaction it must clear the transaction from the PSS.

[1] Depending on which unsolicited data the client has subscribed to. See section 4.2.1.

With *clear_FpUnSupTrans* the Terminal Application can provide Payment Data to be included in the Back Office Record generated by the PSS.

The data is transparent binary data, and the PSS does not interpret the data.

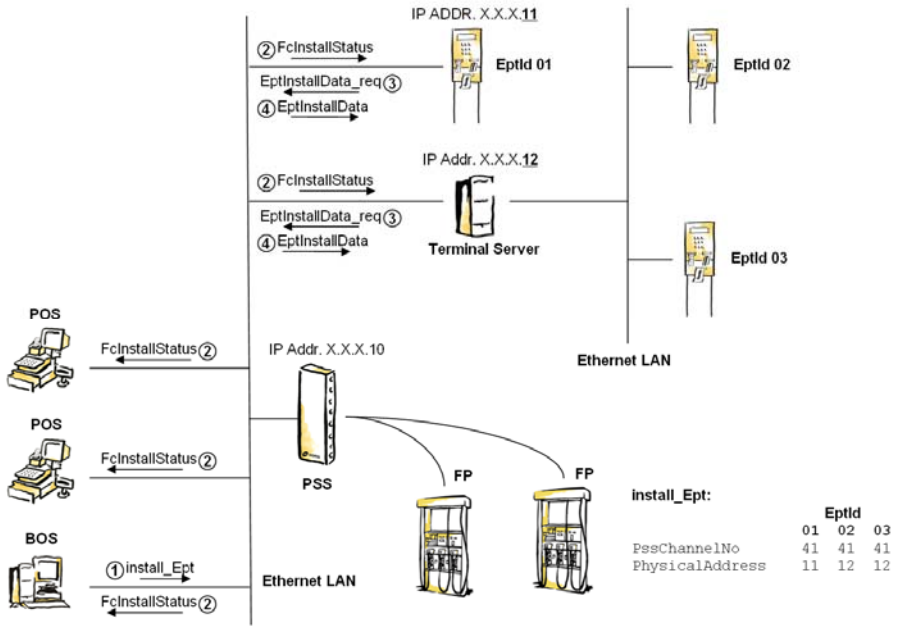
The size of the transparent data must be determined for the single PSS application.

8 Sequence Diagrams

Assumption

The sequence diagrams shown in the following sections are all based on the assumption that the client has subscribed to the unsolicited messages shown or mentioned.

8.1 Configuration Sequence

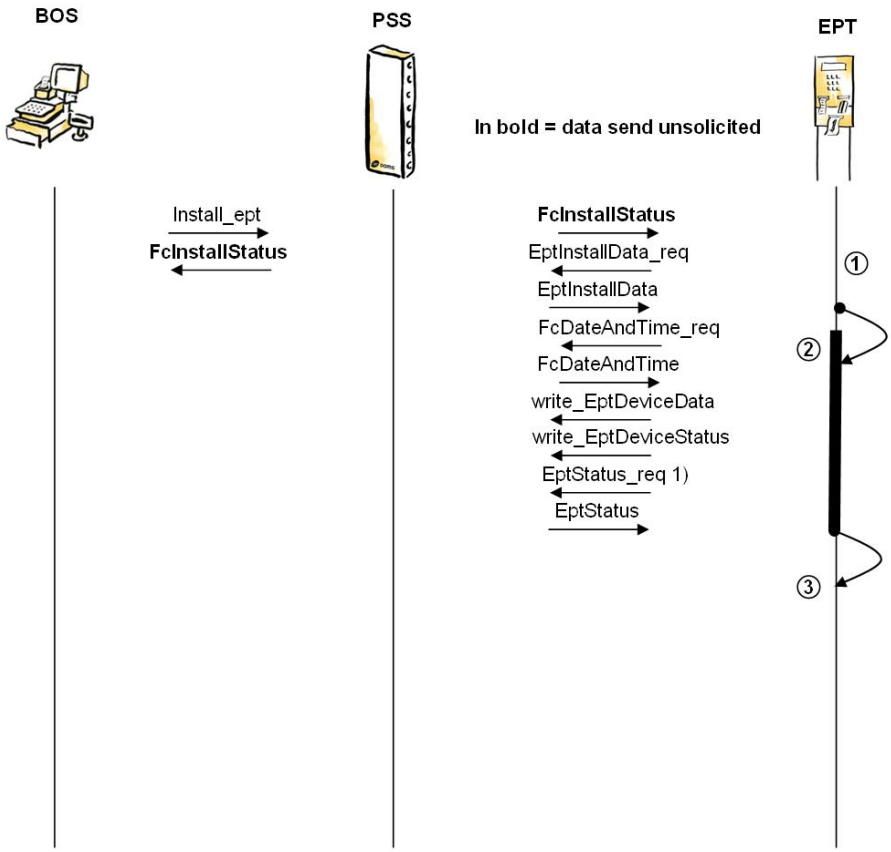


1. PSS receives installation status.
2. PSS sends FcInstallStatus message unsolicited to all connected devices.
3. EPT checks if its own EptId is represented in the EPT FcInstallStatus list, and if so, it sends an EptsInstallData_req message to – among others things - get a list of FPs assigned to that EptId.
4. EPT receives install data and saves the relevant information:
    a. PosId[2] to be used in commands towards PSS

FcInstallStatus is also used by the Terminal Applications to check when the terminal is no longer installed in the PSS (when EptId is no longer in the install list), and do appropriate handling.

[2] PSS issues PosId from 89 and one down for each terminal. POS must used other Id's, typically from 1 and up.

8.2 Initialization sequence



1) EptStatus is used by EPT to check if it terminal is opened for sale.

1. When Terminal Application receives FcInstallStatus it checks if its EptId is in the list. If it is, the terminal is installed in the PSS.

   If terminal has been installed before, it requests EptInstallData and checks if data has changed. If data has changed, the terminal might have been re-installed and it must handle appropriately.

   Terminal is not (or no longer) installed in PSS:

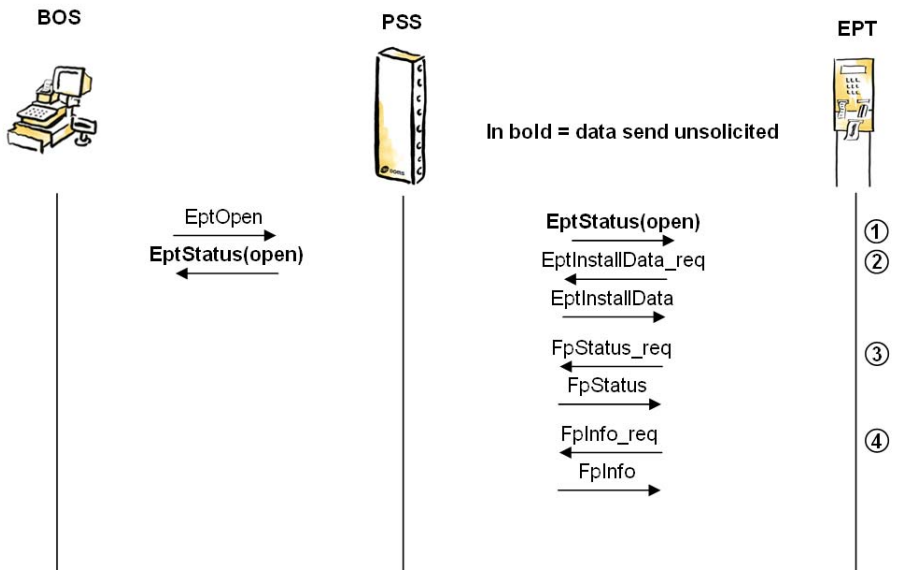
2. The Terminal Application shall make no further requests to the PSS.

   If it had previously been installed, the Terminal Application must clear its installation data received from PSS. The terminal display probably needs to inform customer that the terminal is closed.

3. If the terminal has just been installed / reinstalled, it synchronizes its time with PSS and sends relevant device specific information and status to the PSS.

4. Continue to Open Sequence.

8.3 Open Sequence



The Terminal Application needs to monitor FpStatus for the FPs assigned to it.

1. When the terminal is opened the Terminal Application starts to acquire data needed to operate. In EptStatus the Terminal Application can read the available services.

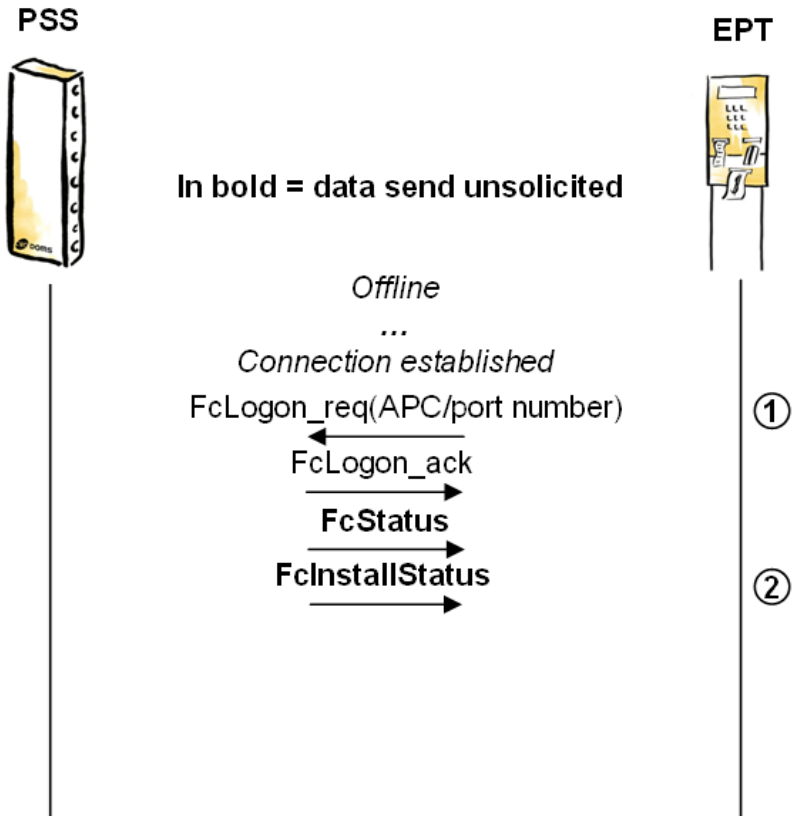
   Which data is needed highly depends on the specific Terminal Application.

2. The Terminal Application collects EPT installation data, and saves relevant install data.

3. If needed, the FP status of the FPs linked to that EPT can be fetched to make it possible for the EPT to verify if the pump is open and ready for card or note sale, whatever relevant.

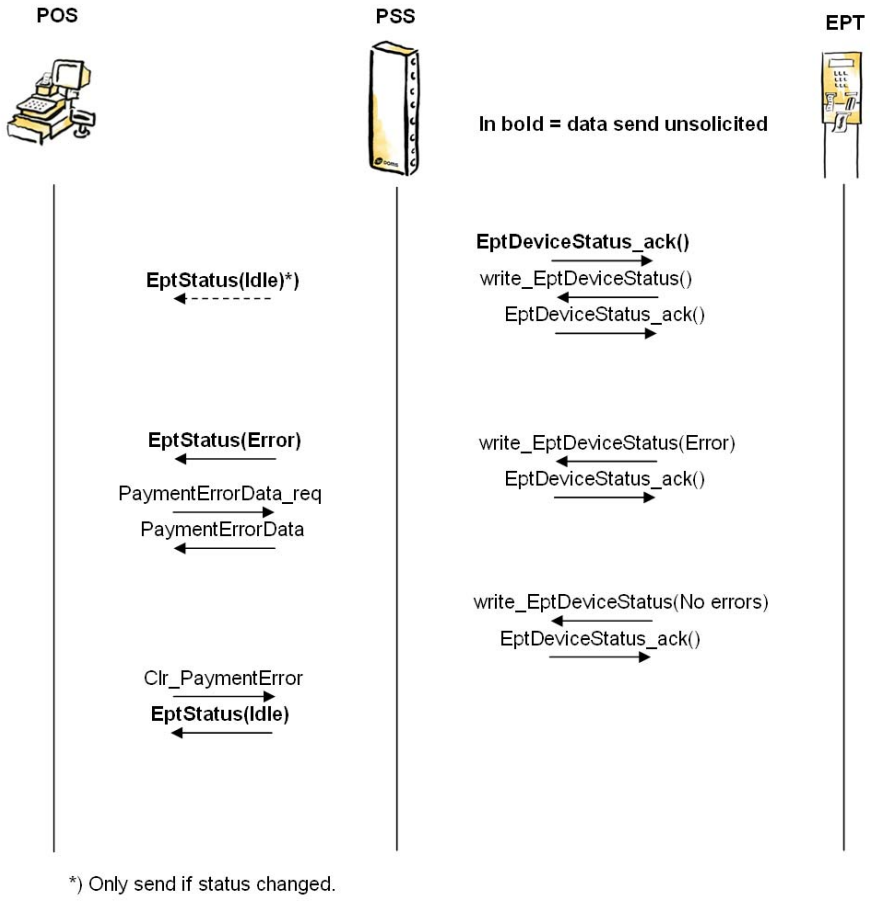
4. To get current grades and prices on the FPs these can be collected using FpInfo_req.

8.4 Online Sequence



1. When EPT has successfully connected to the PSS, EPT sends the logon message to each APC (IP port number) and on success PSS sends FcStatus and FcInstallStatus messages unsolicited.
2. Continue to Initialization Sequence.

8.5 Status handling



The **Terminal Application** (EPT) must send a write_EptDeviceStatus message:

- When the PSS unsolicited (on APC 2) sends a write_EptDeviceStatus_ack (used by the PSS to trigger Terminal Application to send a write_EptDeviceStatus update). A **full update** must be send.
- When it is configured/re-configured in the PSS. A **full update** must be send.
- When already configured and goes from offline to online. A **full update** must be send.
- Each time it detects a change that will result in a change in the message compared to last write_EptDeviceStatus message successfully sent to the PSS.

Full update:

A full update must consist of all parameters (EptDeviceStatusParId) known by the Terminal Application, not just the changes since last time the message was sent.

Note!

EptStatus is also sent to the Terminal Application unsolicited but to keep sequence diagram simpler, it is not shown here.

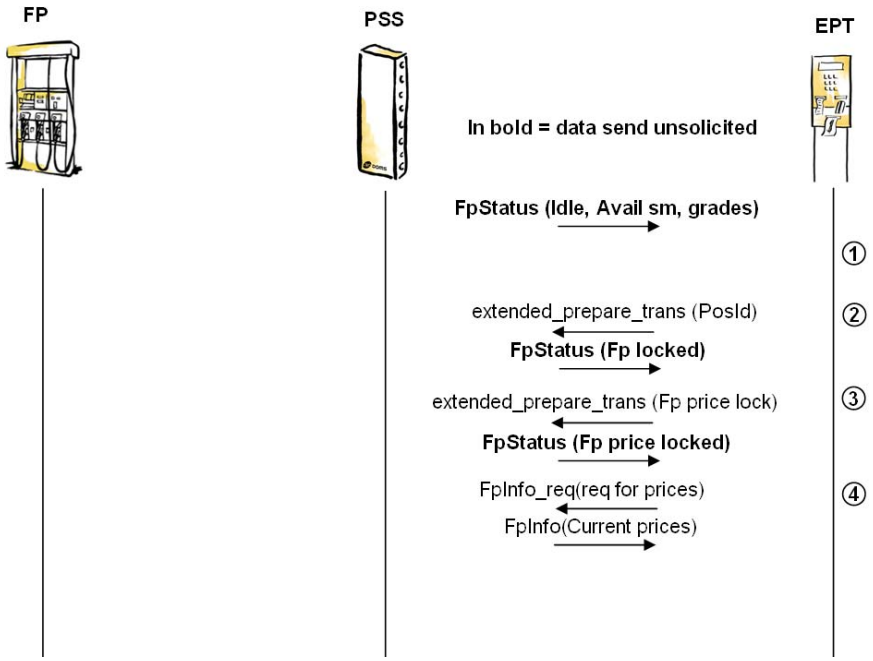
The Terminal Application must use EptStatus to keep itself informed about the available services provided by the PSS. If needed, it can send an EptStatus_req message to get the EptStatus.

Each error must be cleared by the **POS Application**. Some errors can be cleared before the problem has been fixed (printer errors), and the operation will continue with limited functionality (e.g. receipts not possible). Another error will block operation: POS can not clear the error until the problem is no longer there and it has been reported to PSS.

8.6 Transaction handling

Following sequence diagrams show how to handle transactions.

8.6.1 FP Selection (Customer Arrives)



1. Customer arrives and customer dialog starts.

   From previously received unsolicited FpStatus, the Terminal Application has the information about the status, service modes and grade options available for each of the FPs linked to the terminal.

   The available FPs are represented to the customer.

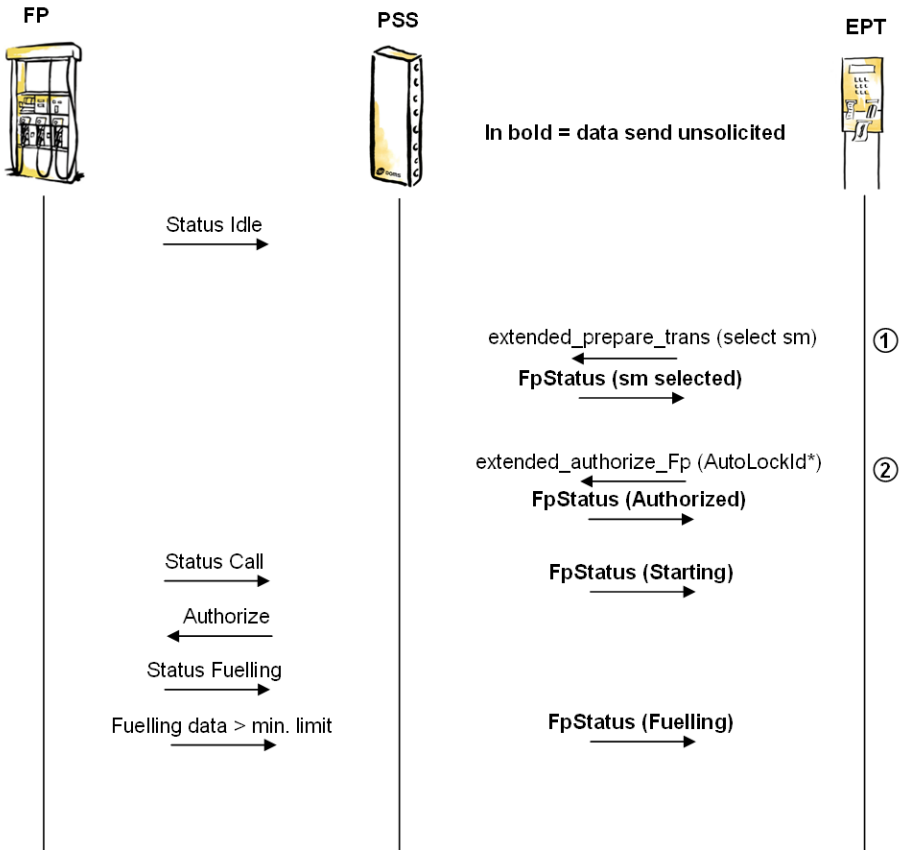
2. Terminal Application reserves (locks) the Fp by specifying PosId in the extended_prepare_trans message.

3. The Terminal Application can optionally lock prices, which may be needed if prices must be shown in the customer dialog. The lock ensures that the customer gets fuel to the prices he is represented to.

4. The current (locked) FP prices can be read, if needed.

   Continue with Authorize Sequence.

8.6.2 Authorization Sequence



1. Terminal Application specifies the service mode used for next transaction (SmId 5x or 6x).

2. Terminal Application authorizes the Fp. If the PSS controls the systems payment log, the authorization command must include the log-data (e.g. receipt number).

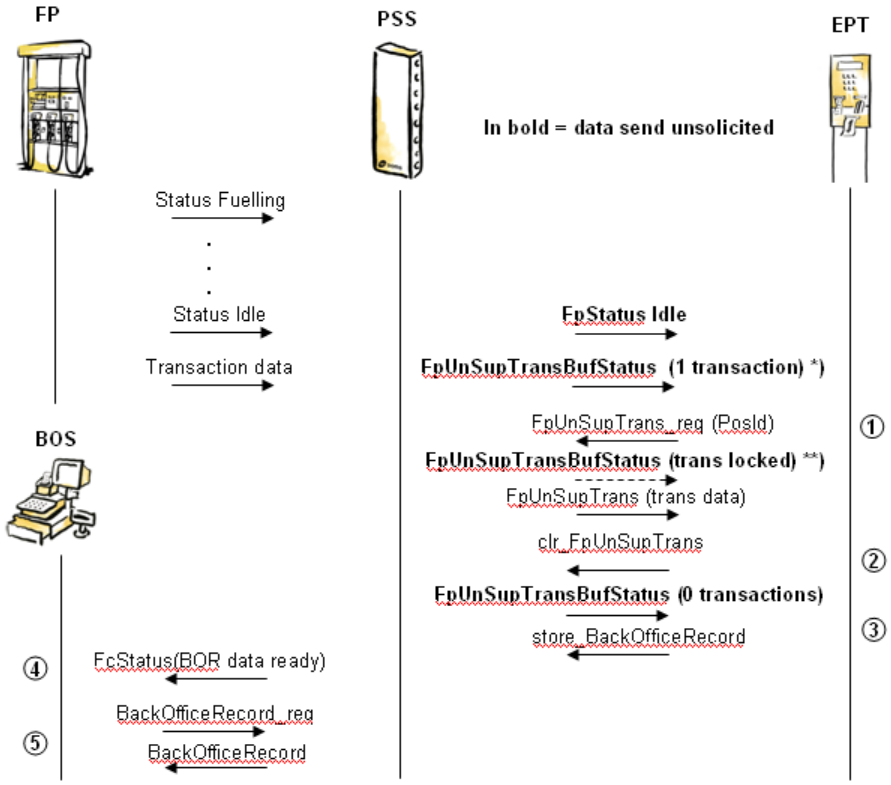
The Terminal Application can send the extended_prepare_trans message with one (as shown in this example) or more parameters.

Note:

If the FP is set in a service mode for bank note, card, or prepaid sale, it will automatically be locked to the PosId specified, unless PosId is ZERO.

*) If AutolockId is specified as AuthParId the PSS will automatically lock the transaction to that Id.

8.6.3 Clear Transaction Sequence



1. Terminal Application requests transaction data using FpUnSupTrans with non-zero PosId in order to lock transaction, or in case it is automatically locked by the PSS (sm parameter), the PosId is used as key.

2. Terminal Application clears transaction in PSS by sending clr_FpUnSupTrans. Optionally, the terminal can specify data to be included in the Back Office Record (BOR) created by the PSS.

3. Back Office Record is created and sent to the PSS.

4. An unsolicited FcStatus message is sent if the PSS BOR buffer previously was empty.

5. BOS retrieves the BOR from the PSS.

9 History

Date	Rev.	Init.	Comments
2009-12-15	00	HBH,PAN	First release