



## Verifone Integration Package

---

Document Type	Technical Interface Specification
Version	3.7.15
Date	30.01.2018
Confidentiality	None
By	HRV Development

---

This Document is the property of Verifone Denmark A/S. It is transferred under the conditions of Active Non-Disclosure.

Use of this document is subject to the terms of this NDA (Non Disclosure Agreement).  
This document will be returned under request to Verifone Denmark A/S.

---



# Content

<b>1 Flexdriver</b>	<b>18</b>
<b>1.1 Preface</b>	<b>18</b>
<b>1.2 Hardware Overview</b>	<b>22</b>
<b>1.3 Software Overview</b>	<b>23</b>
<b>1.4 Functionality</b>	<b>24</b>
1.4.1 Initialisation	24
1.4.1.1 Connect	24
1.4.1.2 Disconnect	24
1.4.1.3 Open	24
1.4.1.4 Close	24
1.4.2 Transaction functionality	24
1.4.2.1 PIN purchase transactions	25
1.4.2.2 Signature purchase transactions	25
1.4.2.3 Signature refund transactions	25
1.4.3 Administrative functionality	25
1.4.3.1 End of day functionality	25
1.4.3.2 Unlock receipt functionality	25
1.4.3.3 Terminal report	26
1.4.3.4 Clock synchronization	26
1.4.3.5 Get DC Properties	26
1.4.4 The Connect and Open procedure	26
1.4.5 Verifone Error codes	28
1.4.6 Timer event callback	28
1.4.7 File operation	28
1.4.8 Language support	28
<b>1.5 The Flexdriver Interface</b>	<b>29</b>
1.5.1 Callback principle	29
1.5.1.1 Flexdriver function	30
1.5.2 Initialisation	30
1.5.2.1 flxInitCallback function	30
1.5.2.2 flxConnect function	32
1.5.2.3 flxOpen function	32
1.5.2.4 flxClose function	32
1.5.2.5 flxDisconnect function	33
1.5.2.6 flxGetTerminalState function	33
1.5.2.7 flxGetFile, flxGetFiles, flxPutFile, flxDeleteFile functions	33
1.5.2.8 flxSetConfiguration function	33
1.5.2.9 flxSetTrace functionality	37
1.5.2.10 flxGetTrace functionality	37
1.5.3 Transaction	37
1.5.3.1 flxCardTransaction function	37
1.5.3.2 flxCompleteExtCardTransaction function	39

1.5.3.3 flxGetID function . . . . .	40
1.5.3.4 flxGetSetExtendedEcrFunctions function . . . . .	40
1.5.3.5 flxTerminalProperties function . . . . .	40
1.5.3.6 flxGetDCProperties functionality . . . . .	43
1.5.4 Administration . . . . .	43
1.5.4.1 flxAdministration functions . . . . .	43
1.5.4.2 Endofday (balancing/clearing) routines . . . . .	44
1.5.4.3 Endofdaylog routine . . . . .	45
1.5.4.4 Terminal report routine . . . . .	45
1.5.4.5 Transaction totals report . . . . .	45
1.5.4.6 Transaction log report . . . . .	45
1.5.4.7 Old log routine . . . . .	45
1.5.4.8 Receipt functions . . . . .	46
1.5.4.9 Clock synchronization routines . . . . .	46
1.5.4.10 Download routines . . . . .	46
1.5.4.11 Contrast routines . . . . .	46
1.5.4.12 Clear data store routine . . . . .	47
1.5.4.13 Restore TLCMDB routine . . . . .	47
1.5.4.14 Restart terminal routine . . . . .	47
1.5.4.15 Send log routine . . . . .	47
1.5.4.16 Eject card routine . . . . .	47
1.5.4.17 Print msc routine . . . . .	48
1.5.4.18 Backlight on routine . . . . .	48
1.5.4.19 Backlight off routine . . . . .	48
1.5.4.20 Network Report . . . . .	48
1.5.4.21 Rates Report . . . . .	48
1.5.4.22 Exclude datastore record with specific stan routine . . . . .	48
1.5.4.23 Advice Forwarding routine . . . . .	49
1.5.4.24 Report file5 Status routine . . . . .	49
1.5.4.25 Report Advice Reconciliation Report routine . . . . .	49
1.5.4.26 Set Batch Number routine . . . . .	49
1.5.4.27 TCS report routine . . . . .	50
1.5.4.28 get Salt . . . . .	50
1.5.4.29 Print the Event report . . . . .	50
1.5.4.30 Send the Event report . . . . .	50
1.5.4.31 Delete the Event report . . . . .	50
1.5.4.32 The TPROPS CVS report routine . . . . .	51
1.5.4.33 Get IP Settings . . . . .	51
1.5.4.34 Set IP Settings . . . . .	51
1.5.4.35 Download Images . . . . .	51
1.5.4.36 Check Card . . . . .	51
1.5.4.37 Get TeleDone . . . . .	52
1.5.4.38 Set TeleDone . . . . .	52
1.5.4.39 Short Contactless Terminal report routine . . . . .	53

1.5.4.40	User input Terminal report routine . . . . .	53
1.5.4.41	Obsolete Terminal report routine . . . . .	53
1.5.4.42	Long Contactless Terminal report routine . . . . .	54
1.5.5	Idle/IP . . . . .	54
1.5.5.1	flxIdleIPforwarding (flxIdle) function . . . . .	54
1.5.6	Token . . . . .	55
1.5.6.1	flxReadToken() function . . . . .	55
1.5.7	Callback . . . . .	55
1.5.7.1	verSigConfirmation callback routine . . . . .	55
1.5.7.2	setCardData callback routine . . . . .	55
1.5.7.3	printReceipt callback routine . . . . .	56
1.5.7.4	printStatus callback routine . . . . .	57
1.5.7.5	abortTransaction callback routine . . . . .	57
1.5.7.6	adviceFlag callback routine . . . . .	57
1.5.7.7	setAmount callback routine . . . . .	58
1.5.7.8	getAmountFee callback routine . . . . .	58
1.5.7.9	setAmountGratuity callback function . . . . .	58
1.5.7.10	Menu callback function . . . . .	59
1.5.7.11	SetMenuResult callback function . . . . .	59
1.5.7.12	AdviceLog callback function . . . . .	59
1.5.7.13	GetReceipt callback function . . . . .	60
1.5.7.14	EarlyStanPan callback function . . . . .	60
1.5.7.15	Timer event callback function . . . . .	60
1.5.7.16	GetToken callback function . . . . .	60
1.5.7.17	PutToken callback function . . . . .	61
1.5.7.18	CheckStopList callback function . . . . .	61
1.5.7.19	BreakIP callback function . . . . .	61
1.5.7.20	HostAdvice callback routine . . . . .	62
1.5.7.21	preResult callback function . . . . .	62
1.5.7.22	getExtendedAmount callback function . . . . .	62
1.5.7.23	setExtendedAmount callback function . . . . .	63
1.5.7.24	EIEdataReceived callback function . . . . .	63
1.5.7.25	EIEdata2host callback function . . . . .	64
1.5.7.26	CardSwipe callback function . . . . .	64
1.5.8	Extra App Protocols . . . . .	65
1.5.8.1	flxExtraReply . . . . .	65
1.5.9	Callback pcbExtraAppdataReceived . . . . .	65
1.5.10	Enabling the pcbExtraAppdataReceived callback . . . . .	66
1.5.11	Flexdriver Flow . . . . .	66
1.5.12	Local cards with Flexdriver – amount displayed . . . . .	67
1.5.13	Local cards with Flexdriver – amount not displayed . . . . .	69
<b>1.6</b>	<b>General Flowcharts . . . . .</b>	<b>70</b>
1.6.1	The important flow of the receipt and the transaction result . . . . .	70
<b>Appendix 1.A</b>	<b>Extra receipt information . . . . .</b>	<b>71</b>

1.A.1 CSV - Comma Separated String - getReceipt . . . . .	73
1.A.2 Example: binary receipt of completed MobilePay transaction . . . . .	75
1.A.3 Example: binary receipt of failed Swipp prepaid scanned bar code purchase . . . . .	77
<b>Appendix 1.B Language Support . . . . .</b>	<b>79</b>
<b>Appendix 1.C Terminal Properties . . . . .</b>	<b>80</b>
<b>Appendix 1.D DCC . . . . .</b>	<b>84</b>
<b>Appendix 1.E Drop IP listing . . . . .</b>	<b>85</b>
<b>Appendix 1.F Extra App Protocols . . . . .</b>	<b>88</b>
1.F.1 flxExtraReply . . . . .	88
1.F.2 Callback pcbExtraAppdataReceived . . . . .	88
1.F.3 Enabling the pcbExtraAppdataReceived callback . . . . .	88
<b>Appendix 1.G Custom Images . . . . .</b>	<b>90</b>
1.G.1 Yomani . . . . .	90
1.G.2 Verifone . . . . .	91
1.G.2.1 Vx 820 . . . . .	91
1.G.2.2 Vx 680 . . . . .	91
1.G.2.3 Vx 520c . . . . .	91
<b>Appendix 1.H How to interpret a Network Report . . . . .</b>	<b>92</b>
<b>2 Local Payment Protocol . . . . .</b>	<b>95</b>
<b>2.1 Preface . . . . .</b>	<b>95</b>
<b>2.2 Hardware Overview . . . . .</b>	<b>96</b>
2.2.1 The Terminal . . . . .	96
<b>2.3 Software Overview . . . . .</b>	<b>97</b>
<b>2.4 Terminal Functionality . . . . .</b>	<b>98</b>
2.4.1 Initialization . . . . .	98
2.4.1.1 Connect . . . . .	98
2.4.1.2 Disconnect . . . . .	98
2.4.1.3 Open . . . . .	98
2.4.1.4 Close . . . . .	98
2.4.2 Transaction functions . . . . .	98
2.4.2.1 PIN purchase transactions . . . . .	98
2.4.2.2 Signature purchase transactions . . . . .	98
2.4.2.3 Signature refund transactions . . . . .	99
2.4.3 Administrative functionality . . . . .	99
2.4.3.1 The End of Day functionality . . . . .	99
2.4.3.2 The unlock receipt functionality . . . . .	99
2.4.3.3 The terminal report . . . . .	99
2.4.3.4 The clock synchronization . . . . .	99
2.4.3.5 Debit/Credit Properties . . . . .	99
2.4.4 The Connect and Open procedure . . . . .	100
<b>2.5 The KISS protocol . . . . .</b>	<b>102</b>
2.5.1 Approach – the KISS & BER–TLV interface . . . . .	102
2.5.2 Frame layout . . . . .	102

2.5.3 Control characters . . . . .	103
2.5.4 Time out controls . . . . .	106
2.5.5 Retry counter . . . . .	107
2.5.6 Byte stuffing . . . . .	107
2.5.7 Transmission Control Sequences . . . . .	107
2.5.8 Error recovery . . . . .	110
2.5.9 Line supervision . . . . .	111
2.5.10 Line characteristics . . . . .	112
2.5.11 Function in C to calculate CRC . . . . .	112
2.5.12 Functions in C to build KISS frame . . . . .	112
2.5.13 Functions in C to verify KISS frame . . . . .	114
<b>2.6 The BER-TLV principle . . . . .</b>	<b>116</b>
2.6.1 Definition . . . . .	116
2.6.2 Properties . . . . .	116
2.6.3 Motivation . . . . .	117
2.6.4 Quick illustrated Description . . . . .	117
<b>2.7 Data objects – Container tags . . . . .</b>	<b>119</b>
2.7.1 The INIT Container . . . . .	119
2.7.2 The DATA Container . . . . .	119
2.7.3 The TRANSACTION Container . . . . .	120
2.7.4 The ADMIN Container . . . . .	121
2.7.5 The INFO Container . . . . .	122
2.7.6 The RECEIPT Container . . . . .	123
2.7.7 The ERROR Container . . . . .	124
2.7.8 The HANDLERSTR Container . . . . .	125
2.7.9 The HOSTDATA Container . . . . .	125
2.7.10 The MENU Container . . . . .	125
2.7.11 The ENAI Container . . . . .	126
2.7.11.1 ENAI Connect reply . . . . .	127
2.7.11.2 ENAI CardSwipe reply . . . . .	128
2.7.11.3 ENAI Card Removed reply . . . . .	128
2.7.11.4 ENAI Send Data reply . . . . .	128
2.7.11.5 ENAI Syncframe . . . . .	128
2.7.11.6 ENAI sync frame – start of program download example . . . . .	128
2.7.12 The STOPLIST tag . . . . .	132
<b>2.8 Flows of TLV data objects . . . . .</b>	<b>133</b>
2.8.1 The flow of a Connect command . . . . .	133
2.8.2 The flow of a Disconnect command . . . . .	133
2.8.3 The flow of an Open command . . . . .	133
2.8.4 The flow of a Close command . . . . .	134
2.8.5 The flow of a PIN Purchase . . . . .	134
2.8.6 The flow of a Signature Purchase . . . . .	137
2.8.7 The application menu functionality . . . . .	140
2.8.8 The Advice log functionality . . . . .	140

2.8.9 Fee functionality . . . . .	140
2.8.10 Get terminal files functionality . . . . .	141
2.8.11 Abort from ECR . . . . .	141
2.8.12 The flow of a Local Card transaction (initiated by ECR) . . . . .	142
2.8.13 The flow of a Local Card transaction (initiated by the user) . . . . .	144
2.8.14 The flow of an endofday . . . . .	146
2.8.15 The flow of a Terminal report . . . . .	147
2.8.16 The flow of a Last transaction receipt request . . . . .	148
2.8.17 Advice transfer flag . . . . .	149
2.8.18 The flow of the Get DC properties . . . . .	149
2.8.19 The flow of the ENAI . . . . .	150
<b>2.9 Extra Application Protocol . . . . .</b>	<b>152</b>
2.9.1 Extra application data from terminal to ECR . . . . .	152
2.9.2 Example Extra application data from terminal to ECR . . . . .	152
2.9.3 Extra application data from ECR to terminal . . . . .	153
2.9.4 Enable ExtraReply in the terminal . . . . .	153
2.9.5 Tell terminal ECR listen for extra app data . . . . .	155
2.9.6 Example LPP Extra application data from ECR to terminal . . . . .	156
<b>2.10 Examples . . . . .</b>	<b>157</b>
2.10.1 Connect . . . . .	157
2.10.2 Open . . . . .	158
2.10.3 Close . . . . .	160
2.10.4 Disconnect . . . . .	161
2.10.5 Disconnect . . . . .	162
2.10.6 ClockSyncNETS . . . . .	162
2.10.7 Extended Issuer Envelope (EIE) . . . . .	169
2.10.7.1 SWE additional data IE part of data may contain: . . . . .	170
2.10.7.2 DCC Transaction Information . . . . .	170
2.10.7.3 Example: . . . . .	170
2.10.7.4 On terminal boot we get from the PSAM with Identifier 0x0011 . . . . .	173
2.10.7.5 The total envelope data is send to the ECR as part of OPEN pa- rameters . . . . .	173
2.10.7.6 Example DLL kasse demo - no longer supported: . . . . .	175
2.10.7.7 EIEdata2host . . . . .	177
2.10.7.8 EIEdataReceived . . . . .	177
2.10.7.9 EIEdata2host . . . . .	177
2.10.7.10 Example LPP – from DLL kasse demo - no longer supported . . . . .	180
<b>2.11 General Flowcharts . . . . .</b>	<b>200</b>
2.11.1 The important flow of the receipt and the transaction result . . . . .	200
<b>Appendix 2.A Appendix A . . . . .</b>	<b>201</b>
2.A.1 Tag definitions . . . . .	201
2.A.2 Value definitions . . . . .	206
2.A.3 Transaction data . . . . .	206
2.A.4 Amount data . . . . .	207

2.A.5 Command data . . . . .	208
2.A.6 Result data . . . . .	232
2.A.6.1 Result data with Transaction . . . . .	232
2.A.6.2 Result data with Admin . . . . .	232
2.A.6.3 Result data with Receipt . . . . .	232
2.A.7 Binary data . . . . .	233
2.A.7.1 Binary data with Info–Text . . . . .	233
2.A.7.2 Binary data with Transaction–Result . . . . .	233
2.A.7.3 Binary data with Admin–Result . . . . .	233
2.A.8 Binary data with Receipt–Text . . . . .	234
2.A.9 Acqmsg tags . . . . .	234
2.A.9.1 Acqmsg data with Transaction . . . . .	234
2.A.10 Error tags . . . . .	234
2.A.10.1 Receiving Error data . . . . .	234
2.A.11 Abort data . . . . .	235
2.A.11.1 Abort data with Info . . . . .	235
2.A.12 CAC9 data . . . . .	235
2.A.12.1 CAC9 data with Transaction . . . . .	235
2.A.13 ExtendedECR data . . . . .	235
2.A.13.1 ExtendedECR functions . . . . .	235
2.A.14 Terminal states and number . . . . .	237
2.A.15 Description of locked states . . . . .	239
2.A.16 The Merchant events and number . . . . .	240
2.A.17 The Merchant state–event diagram . . . . .	243
2.A.18 The Merchant state–events . . . . .	243
<b>Appendix 2.B Drop IP . . . . .</b>	<b>248</b>
<b>Appendix 2.C How to interpret a Network Report . . . . .</b>	<b>252</b>
<b>3 Spin Connect . . . . .</b>	<b>255</b>
<b>3.1 Preface . . . . .</b>	<b>255</b>
<b>3.2 Hardware Overview . . . . .</b>	<b>256</b>
3.2.1 The Terminal . . . . .	256
<b>3.3 Software Overview . . . . .</b>	<b>258</b>
<b>3.4 Functionality . . . . .</b>	<b>259</b>
3.4.1 Initialization . . . . .	259
3.4.2 Transaction Functionality . . . . .	259
3.4.2.1 PIN Purchase Transactions . . . . .	259
3.4.2.2 Signature Purchase Transactions . . . . .	259
3.4.2.3 Signature Refund Transactions . . . . .	259
3.4.2.4 Offline Refund Transactions . . . . .	260
3.4.2.5 MSC Loyalty Card Transactions . . . . .	260
3.4.2.6 Cancellation Transactions . . . . .	260
3.4.2.7 Cash Transactions . . . . .	260
3.4.2.8 Cash Refund Transactions . . . . .	261



3.4.2.9 Error Handling . . . . .	261
3.4.3 Administrative Functionality . . . . .	261
3.4.3.1 The End of Day Functionality . . . . .	261
3.4.3.2 The Outstanding Result Functionality . . . . .	261
3.4.3.3 The Terminal Report . . . . .	262
3.4.3.4 The Clock Synchronization . . . . .	262
3.4.4 Language Support . . . . .	262
<b>3.5 The Interface of Spin Connect . . . . .</b>	<b>263</b>
3.5.1 Spin Connect Protocol Version . . . . .	263
3.5.2 Printing Receipts on the Terminal . . . . .	263
3.5.3 Encrypting the Communication . . . . .	263
3.5.3.1 Using RS232 . . . . .	264
3.5.3.2 Calculating CRC . . . . .	266
<b>3.6 Configuration Options . . . . .</b>	<b>268</b>
3.6.1 Tested configuration . . . . .	268
3.6.2 Specific Settings for Spin Connect . . . . .	268
<b>3.7 Transaction Scenarios . . . . .</b>	<b>270</b>
3.7.1 Scenario A (Fetch by Table) . . . . .	270
3.7.2 Scenario B (Fetch by List) . . . . .	271
3.7.3 Scenario C (Fetch by Check ID) . . . . .	273
3.7.4 Scenario D (one-to-one) . . . . .	275
<b>3.8 The Spin Connect Functions . . . . .</b>	<b>277</b>
3.8.1 Get Check (001) . . . . .	277
3.8.2 Get Check Response (101) . . . . .	277
3.8.3 Check List (102) . . . . .	277
3.8.4 Admin Response (103) . . . . .	278
3.8.5 Check Result (002) . . . . .	278
3.8.6 Check Result Response (201) . . . . .	278
3.8.7 MSC Loyalty Card (003) . . . . .	278
3.8.8 Version Information (004) . . . . .	279
3.8.9 Version Information Response (401) . . . . .	279
<b>Appendix 3.A Parameters . . . . .</b>	<b>280</b>
<b>4 Gavekort Scan Bar Code . . . . .</b>	<b>283</b>
<b>4.1 Flex driver (DLL) . . . . .</b>	<b>283</b>
4.1.1 Callback pcbPutScanBarCode . . . . .	283
4.1.2 Saldo and expire data in callback pcbGetReceipt/pcbGetReceiptVB . . . . .	283
4.1.3 Example: binary receipt of prepaid scanned bar code purchase . . . . .	284
4.1.4 Example: binary receipt of completed MobilePay transaction . . . . .	285
4.1.5 Example: binary receipt of failed Swipp prepaid scanned bar code purchase . . . . .	287
<b>4.2 Local Payment Protocol (LPP) . . . . .</b>	<b>288</b>
4.2.1 LPP start scanned bar code transaction . . . . .	288
4.2.2 LPP prepaid scanned bar code to ECR . . . . .	288
4.2.3 LPP PTAG_RECEIPT now contains PTAG_SALDO and PTAG_EXP_DATE . . . . .	289

4.2.4 Example: LPP Start of scanned bar code purchase from terminal to ECR . . .	289
4.2.5 Example: LPP Start of Swipp scanned bar code purchase from terminal to ECR	290
<b>4.3 Configuration . . . . .</b>	<b>292</b>
<b>5 Extra App Protocols . . . . .</b>	<b>293</b>
<b>5.1 Preface . . . . .</b>	<b>293</b>
5.1.1 Format of the header . . . . .	293
5.1.2 Application number . . . . .	293
5.1.3 Version . . . . .	293
5.1.4 Error . . . . .	293
5.1.5 Last block . . . . .	294
<b>5.2 The Data Block . . . . .</b>	<b>294</b>
5.2.1 The Data Block . . . . .	294
5.2.2 Displayline . . . . .	294
5.2.3 Example plaintext data block . . . . .	295
5.2.4 Example XML data block . . . . .	295
<b>5.3 XML format types . . . . .</b>	<b>296</b>
5.3.1 String . . . . .	296
5.3.2 Hexstring . . . . .	296
5.3.3 Displayline . . . . .	296
5.3.4 Select pictures . . . . .	297
5.3.5 Select picture reply . . . . .	298
5.3.6 Print receipt . . . . .	298
<b>5.4 Generic LPP . . . . .</b>	<b>298</b>
5.4.1 Generic LPP start extra application - Not implemented - Do not use . . . . .	298
5.4.2 Generic LPP extra application transaction completed - Not implemented - Do not use . . . . .	299
5.4.3 Generic LPP card data - Not implemented - Do not use . . . . .	299
5.4.4 Generic LPP card data reply - Not implemented - Do not use . . . . .	299
5.4.5 Generic LPP info text . . . . .	300
5.4.6 Generic LPP info TSI (Transaction State Info) . . . . .	300
5.4.7 Generic LPP menu . . . . .	300
5.4.8 Generic LPP menu reply . . . . .	300
5.4.9 Generic LPP admin . . . . .	301
5.4.10 Generic LPP admin reply . . . . .	301
5.4.11 Generic LPP receipt . . . . .	301
5.4.12 Extra app special . . . . .	302
<b>5.5 Flexdriver (DLL) . . . . .</b>	<b>302</b>
5.5.1 flxExtraReply . . . . .	302
5.5.2 Callback pcbExtraAppdataReceived . . . . .	303
5.5.3 Enabling the pcbExtraAppdataReceived callback . . . . .	303
<b>5.6 Local Payment Protocol (LPP) . . . . .</b>	<b>304</b>
5.6.1 LPP Extra application data from terminal to ECR . . . . .	304
5.6.2 Example LPP Extra application data from terminal to ECR . . . . .	304

5.6.3 LPP Extra application data from ECR to terminal . . . . .	305
5.6.4 Enable ExtraReply in the terminal . . . . .	305
5.6.5 Tell terminal ECR to listen for extra app data . . . . .	307
5.6.6 Example LPP Extra application data from ECR to terminal . . . . .	308
<b>5.7 Extra application example code . . . . .</b>	<b>308</b>
5.7.1 Usage: Extra application example . . . . .	309
<b>6 PointTerminal (DLL) . . . . .</b>	<b>310</b>
6.1 Preface . . . . .	310
<b>6.2 State machines for PointTerminalDLL . . . . .</b>	<b>311</b>
6.2.1 Abstract state machine . . . . .	312
6.2.2 LoadTerminal and UnloadTerminal functionality . . . . .	313
6.2.3 ConnectTerminal, OpenTerminal, CloseTerminal and DisconnectTerminal func- tionality . . . . .	314
<b>6.3 Interface . . . . .</b>	<b>315</b>
6.3.1 Enums . . . . .	315
6.3.2 Methods . . . . .	322
6.3.2.1 SetTerminalConfiguration(7) . . . . .	322
6.3.2.2 SetTerminalConfiguration(6) . . . . .	323
6.3.2.3 SetTerminalConfiguration(5) . . . . .	323
6.3.2.4 SetWindowsPrintConfiguration . . . . .	324
6.3.2.5 SetRawPrintConfiguration . . . . .	325
6.3.2.6 SetDisplayWindowPosition . . . . .	325
6.3.2.7 LoadTerminal . . . . .	325
6.3.2.8 UnloadTerminal . . . . .	325
6.3.2.9 ConnectTerminal . . . . .	326
6.3.2.10 OpenTerminal . . . . .	326
6.3.2.11 CloseTerminal . . . . .	326
6.3.2.12 DisconnectTerminal . . . . .	326
6.3.2.13 Transaction(13) . . . . .	327
6.3.2.14 Transaction(11) . . . . .	328
6.3.2.15 Transaction(8) . . . . .	328
6.3.2.16 Transaction(5) . . . . .	329
6.3.2.17 Transaction(3) . . . . .	330
6.3.2.18 Transaction(1) . . . . .	331
6.3.2.19 Refund . . . . .	331
6.3.2.20 Cancellation . . . . .	331
6.3.2.21 Administration . . . . .	332
6.3.2.22 AdministrationIPSettingsDHCP . . . . .	332
6.3.2.23 AdministrationIPSettingsSTATIC . . . . .	332
6.3.2.24 AdministrationGetAdviceReconciliation . . . . .	333
6.3.2.25 AdministrationSetBatchNumber . . . . .	333
6.3.2.26 AdministrationExcludeDatastoreWithSTAN . . . . .	334
6.3.2.27 AdministrationSetTeleDoneSettings . . . . .	334

6.3.2.28	PTInitCallback	334
6.3.2.29	SetAcceptCardResult	335
6.3.2.30	SetCardDataResult	335
6.3.2.31	PrintReceiptResult	335
6.3.2.32	GetTokenResult	336
6.3.2.33	PutTokenResult	336
6.3.2.34	PcbStopListResult	336
6.3.2.35	SetExtendedAmountResult	336
6.3.2.36	EIEDataReceivedResult	337
6.3.2.37	EIEData2HostResult	337
6.3.2.38	PutScanBarCodeResult	337
6.3.2.39	PcbBreakIPResult	338
6.3.2.40	CTLSTransactions	338
6.3.2.41	GetFlexDriverVersion	338
6.3.2.42	GetPointTerminalVersion	338
6.3.2.43	GetTokenInfo	339
6.3.2.44	GetTransactionResult	339
6.3.2.45	SetInterfaceReceipt	339
6.3.2.46	SetPrintHeader	340
6.3.2.47	SetPrintFooter	340
6.3.2.48	ForcePrint	340
6.3.2.49	GetCardSwipe	340
6.3.2.50	ShowDisplayWindow	341
6.3.2.51	GetLicense	341
6.3.3	Events	341
6.3.3.1	Terminal connection	341
6.3.3.1.1	LoadFinished	341
6.3.3.1.2	UnloadFinished	341
6.3.3.1.3	ConnectFinished	342
6.3.3.1.4	OpenFinished	342
6.3.3.1.5	CloseFinished	342
6.3.3.1.6	DisconnectFinished	342
6.3.3.1.7	TerminalDataFinished	342
6.3.3.2	Transactions	343
6.3.3.2.1	TransactionFinished	343
6.3.3.2.2	AcceptCard	345
6.3.3.2.3	SetCardData	345
6.3.3.2.4	PrintReceipt	346
6.3.3.2.5	EarlyStanPan	346
6.3.3.2.6	GetToken	346
6.3.3.2.7	PutToken	346
6.3.3.2.8	PcbStopList	347
6.3.3.2.9	GetExtendedAmount	347
6.3.3.2.10	SetExtendedAmount	347

6.3.3.2.11 EIEDataReceived . . . . .	347
6.3.3.2.12 EIEData2Host . . . . .	347
6.3.3.2.13 PutScanBarCode . . . . .	348
6.3.3.3 Administration . . . . .	<b>348</b>
6.3.3.3.1 AdministrationFinished . . . . .	348
6.3.3.4 Other . . . . .	<b>348</b>
6.3.3.4.1 CardSwiped . . . . .	348
6.3.3.4.2 GetReceipt . . . . .	348
6.3.3.4.3 AdviceFlag . . . . .	348
6.3.3.4.4 PcbBreakIP . . . . .	349
<b>7 PointWare Expedient File Interface</b>	<b>350</b>
7.1 Interface . . . . .	<b>350</b>
7.2 Commands . . . . .	<b>351</b>
7.2.1 Connection . . . . .	<b>351</b>
7.2.1.1 LoadTerminal . . . . .	<b>351</b>
7.2.1.2 UnloadTerminal . . . . .	<b>353</b>
7.2.1.3 Ready . . . . .	<b>354</b>
7.2.1.4 Welcome . . . . .	<b>355</b>
7.2.1.5 Close . . . . .	<b>355</b>
7.2.1.6 Exit . . . . .	<b>355</b>
7.2.2 Transactions . . . . .	<b>356</b>
7.2.3 Administration . . . . .	<b>359</b>
7.2.4 Print . . . . .	<b>362</b>
7.2.5 Show or minimize PWE . . . . .	<b>363</b>
<b>8 PointContactless</b>	<b>364</b>
8.1 Document History . . . . .	<b>364</b>
8.2 Introduction . . . . .	<b>364</b>
8.3 Other Documentation . . . . .	<b>364</b>
8.4 Format for Sending . . . . .	<b>364</b>
8.5 Message Structure . . . . .	<b>364</b>
8.6 Notes on the Security Trailer . . . . .	<b>365</b>
8.7 AcceptorAuthorisationRequest . . . . .	<b>365</b>
8.7.1 Example . . . . .	<b>368</b>
8.8 AcceptorAuthorisationResponse . . . . .	<b>370</b>
8.9 AcceptorCompletionAdvice . . . . .	<b>372</b>
8.10 AcceptorCompletionAdviceResponse . . . . .	<b>374</b>
8.11 AcceptorCancellationRequest . . . . .	<b>376</b>
8.12 AcceptorCancellationResponse . . . . .	<b>378</b>
8.13 AcceptorCancellationAdvice . . . . .	<b>380</b>
8.14 AcceptorCancellationAdviceResponse . . . . .	<b>382</b>
8.15 Communication Security . . . . .	<b>383</b>
8.15.1 Security Trailer . . . . .	<b>383</b>

8.15.1.1 Key Management . . . . .	383
8.15.1.2 MAC Computation . . . . .	384
8.15.1.3 Example . . . . .	385
8.15.2 Encrypting the Communication . . . . .	386
<b>8.16 Flow of Transactions (Sequence Diagrams) . . . . .</b>	<b>387</b>
8.16.1 Successful Authentication . . . . .	387
8.16.2 Failed Authentication . . . . .	387
8.16.3 No Authorisation Response . . . . .	388
8.16.4 No Completion Response . . . . .	389
<b>8.17 Suggestions for Additional Message Items . . . . .</b>	<b>389</b>
8.17.1 Example with no PAN and UUID . . . . .	390
<b>9 Point Payment Interface . . . . .</b>	<b>393</b>
9.1 Preface . . . . .	393
9.2 System Concept . . . . .	397
9.3 CardRequest and CardResponse . . . . .	399
9.3.1 CardRequest (Payment) . . . . .	399
9.3.2 CardResponse (Payment) . . . . .	401
9.3.3 CardRequest (Refund) . . . . .	405
9.3.4 CardRequest (BalanceQuery) . . . . .	405
9.3.5 CardRequest (Abort) . . . . .	405
9.3.6 CardRequest (Cancellation) . . . . .	405
9.3.7 CardRequest (Authorisation) . . . . .	405
9.3.8 CardRequest (SuppAuthorisation) . . . . .	405
9.3.9 CardRequest (CaptureAuthorisation) . . . . .	405
9.3.10 CardRequest (CancelAuthorisation) . . . . .	405
9.3.11 CardRequest (PrepaidScanPurchase) . . . . .	406
9.3.12 CardRequest (PrepaidScanRefund) . . . . .	406
9.3.13 CardRequest (PrepaidScanAuthorisation) . . . . .	406
9.3.14 CardRequest (PostPurchase) . . . . .	406
9.3.15 CardRequest (PostRefund) . . . . .	406
9.3.16 SubCode table . . . . .	406
9.4 DeviceRequest and DeviceResponse . . . . .	408
9.4.1 DeviceRequest (GetCardConfirmation) . . . . .	408
9.4.2 DeviceResponse (GetCardConfirmation) . . . . .	409
9.4.3 DeviceRequest (Print) . . . . .	411
9.4.4 DeviceResponse (Printer) . . . . .	412
9.4.5 DeviceRequest (Display) . . . . .	412
9.4.6 DeviceResponse (Display) . . . . .	413
9.4.7 DeviceRequest (GetKeyInput) . . . . .	413
9.4.8 DeviceResponse (GetKeyInput) . . . . .	413
9.4.9 DeviceRequest (GetConfirmation) . . . . .	413
9.4.10 DeviceResponse (GetConfirmation) . . . . .	414
9.4.11 DeviceRequest (GetMenu) . . . . .	414

9.4.12 DeviceResponse (GetMenu) . . . . .	415
9.4.13 DeviceRequest (AdviceFlag) . . . . .	415
9.4.14 DeviceResponse (AdviceFlag) . . . . .	415
<b>9.5 ServiceRequest and ServiceResponse . . . . .</b>	<b>416</b>
9.5.1 ServiceRequest (Reconciliation) . . . . .	416
9.5.2 ServiceResponse (Reconciliation) . . . . .	416
9.5.3 ServiceRequest (Diagnosis) . . . . .	416
9.5.4 ServiceResponse (Diagnosis) . . . . .	416
9.5.5 ServiceRequest (Login) . . . . .	417
9.5.6 ServiceResponse (Login) . . . . .	417
9.5.7 ServiceRequest (Logoff) . . . . .	418
9.5.8 ServiceResponse (Logoff) . . . . .	418
9.5.9 ServiceRequest (AdminFunction) . . . . .	418
9.5.10 ServiceResponse (AdminFunction) . . . . .	418
9.5.11 ServiceRequest (SignatureOnOff) . . . . .	419
9.5.12 ServiceResponse (SignatureOnOff) . . . . .	419
9.5.13 ServiceRequest (OfflineOnOff) . . . . .	419
9.5.14 ServiceResponse (OfflineOnOff) . . . . .	420
9.5.15 ServiceRequest (AdminMenu) . . . . .	420
9.5.16 ServiceResponse (AdminMenu) . . . . .	424
9.5.17 ServiceRequest (GetPreviousResult) . . . . .	424
9.5.18 ServiceResponse (GetPreviousResult) . . . . .	425
<b>9.6 Configuration and deployment . . . . .</b>	<b>426</b>
9.6.1 Single instance mode . . . . .	426
9.6.2 Server mode . . . . .	426
9.6.3 Reload config file . . . . .	427
<b>9.7 Running PPI as a Windows Service . . . . .</b>	<b>428</b>
9.7.1 Before installation of PPIService . . . . .	428
9.7.2 Install PPIService . . . . .	428
9.7.3 Uninstall PPIService . . . . .	428
<b>9.8 Default error messages . . . . .</b>	<b>429</b>
<b>9.9 Configuration (config.xml) . . . . .</b>	<b>429</b>
<b>9.10 Register DLL (FOS5) . . . . .</b>	<b>434</b>
<b>9.11 Update Logic . . . . .</b>	<b>434</b>
<b>9.12 Log files . . . . .</b>	<b>436</b>
<b>10 NFC . . . . .</b>	<b>438</b>
10.1 MIFARE Ultralight . . . . .	438
<b>11 Connection Service Provider . . . . .</b>	<b>440</b>
11.1 Internetbetingelser . . . . .	440
11.1.1 Verifone Denmark's internetbetingelser . . . . .	440
11.2 Verifone Denmark's Internet Requirements . . . . .	442

<b>12 Implementation Guide for Integrators</b>	<b>444</b>
12.1 Objective	444
12.2 Completing the Integration – Step by Step	444
12.3 Other Documents	447
<b>13 Development Agreement</b>	<b>448</b>
13.1 Scope of Agreement	448
13.2 Subject Matter	448
13.3 Verifone's Obligations	448
13.4 The Client's Obligations	448
13.5 Service	449
13.6 Independent Parties	449
13.7 Rights of Third Parties	449
13.8 Liability	449
13.9 Commencement of Agreement	450
13.10 Termination of Agreement	450
13.11 Transference of Agreement	450
13.12 Written Agreements	450
13.13 Confidential Information	450
13.14 Force Majeure	451
13.15 Venue/Choice of Law	451
13.16 Comments	451
13.17 Signature	452
13.18 Appendix A – Contact	453
13.19 Appendix B – Development Agreement	454
13.20 Appendix C – Timetable	457
<b>14 Udviklingsaftale</b>	<b>458</b>
14.1 Aftalens omfang	458
14.2 Aftalens genstand	458
14.3 Verifones forpligtelser	458
14.4 Kundens forpligtelser	458
14.5 Service	459
14.6 Uafhængige Parter	459
14.7 Tredje mands rettigheder	459
14.8 Ansvar	459
14.9 Aftalens ikrafttræden	460
14.10 Ophævelse af aftalen	460
14.11 Overdragelse af aftalen	460
14.12 Skriftlige aftaler	460
14.13 Fortrolighed	460
14.14 Force majeure	460
14.15 Værneting/Lovvalg	461
14.16 Bemærkninger	461



<b>14.17 Underskrift</b>	<b>462</b>
<b>14.18 Bilag A - Kontrakt</b>	<b>463</b>
<b>14.19 Bilag B - Udviklingsaftale</b>	<b>464</b>
<b>14.20 Bilag C - Tidsplan</b>	<b>467</b>
<b>15 FAQ</b>	<b>468</b>
<b>15.1 Basic Information</b>	<b>468</b>
<b>15.2 How is Certification Handled?</b>	<b>469</b>
15.2.1 Important before a Certification	469
<b>15.3 Description of Best Practice</b>	<b>470</b>
<b>15.4 Configuration of the Test Terminal</b>	<b>471</b>
15.4.1 Special Function Configuration	471
<b>15.5 Connect a Terminal to ECR</b>	<b>472</b>
<b>15.6 How to Connect the Terminal</b>	<b>473</b>
15.6.1 Ethernet Demands	473
15.6.2 Ethernet – How the Terminal can obtain an IP Address	473
15.6.3 Ethernet – Network Required Addresses/Names	474
15.6.4 IP and Ports used during Certification	474
15.6.5 No connection to terminal - Disable Cisco Skinny Call Control Protocol	475
15.6.6 Guide to Ethernet trace.	475
15.6.6.1 What to Capture	475
15.6.6.2 What to look for	476
<b>15.7 What is needed to make a Test Transaction</b>	<b>477</b>
<b>15.8 ‘Ingen kvittering’ – ‘No Receipt’</b>	<b>478</b>
15.8.1 Using ADMIN and ‘Pasord’	478
<b>15.9 Important Info about ECR Application</b>	<b>479</b>
<b>15.10 Error Codes – Nice to Know</b>	<b>480</b>
<b>15.11 Interpretation of other Errors</b>	<b>481</b>
<b>15.12 ‘Systemfejl’ – Errors not listed in OTRS</b>	<b>482</b>
<b>15.13 Handling of Different Transaction Types</b>	<b>483</b>
15.13.1 PIN Purchase Transactions	483
15.13.2 Signature Purchase Transactions	483
15.13.3 Offline Purchase Transactions	483
15.13.4 Refund Transactions	483
<b>15.14 Terminal Messages – in Danish and English</b>	<b>484</b>
<b>15.15 Common ECR Problems in Certification</b>	<b>485</b>
15.15.1 Handling of Multipart Receipts	485
15.15.2 Handling of Receipt Reprint	485
15.15.3 Reprint of Receipts in a given Period	485
15.15.4 Password Protection of some Admin Functions	485
15.15.5 Large Transactions	485
<b>15.16 Traces – in Case of Problems</b>	<b>486</b>

# 1 | Flexdriver

## 1.1 Preface

This document describes one of three different approaches for interfacing to the Flex Terminal. The approach described here is the Flexdriver.

Before using the Flexdriver in your ECR solution you must sign a development agreement with Verifone Denmark A/S.

The certifying authority, Nets, must certify an ECR implementation using the Flexdriver. Information of the certifying process can be found at Nets website.

Please read the FAQ.pdf section "Important before a Certification".

The Flexdriver (DLL/LIB) supports simple function calls that initiate transactions and administrative functions and the results are delivered in callback functions. The Flexdriver is a single thread system, which means that all callback functions must be processed as fast as possible. This means that stopping the thread in a callback function can cause an 'out of sync' situation. Using breakpoints during software development must therefore be done with care.

From version 2.1.00 of the flxdrv.dll, the procedures and functions of the Flexdriver are callable from Microsoft Visual Basic 6 and Microsoft .NET.

### Revision

#### 3.7.15

Fixed stoplist callback so we zero terminate and don't look for code before OK response  
Early stan pan longer than buffer  
Disable connection id send in every packet

#### 3.7.14

MobilePay support  
Ip-routing Timeout changed  
Swipp no longer supported as of 28-02-2017

#### 3.7.12

Xenta and Xentissimo are no longer supported.  
Scan bar code purchase now support Swipp transactions.  
New Post Transaction Types.  
Extended receipt have to new fields:  
•VAS - Json receipt info  
•TCC - Transaction Condition Code.

#### 3.7.11

## Revision

Long ip connect timeout reduced.  
Test of network removed if connection is lost.

3.7.10

Test of network (netstat) if connection is lost.  
Advices send to ecr if configured in psam by terminal.

3.7.00

Contactless transactions now supported and a callback for (early) amount is issued. A new report added for contactless.  
The flexdriver will now exit current method if a terminal sync is received during transaction or administrative functions.  
New flxTerminalProperties command for user input.  
Relaxed windows error check for serial port connection.

3.6.04

Better handling of lost USB connection, after the transaction/admin result 0x10002 - VENT LIDT, PRØV IGEN, it's now possible to perform an administration function by trying once more.

3.6.00

In the BinaryReceipt callback, sporadic application stop has been seen, this is now fixed.  
Better handling of lost USB connection, after the transaction/admin result 0x10002 - VENT LIDT, PRØV IGEN, it's now possible to perform the transaction by trying once more.  
flxAdministration(ADMIN\_DOWNLOADPROGRAM retur SUCCESS if error set to TEDO 0x5465446f.

3.5.03

ADMIN\_DOWNLOADPROGRAM with IP routing now has the possibility to change the behavior by setting some values using flxSetConfiguration.  
Six new parameters can be used with flxSetConfiguration:  
CONF\_DOWNLOAD\_TIMEOUT.  
CONF\_DOWNLOAD\_BLOCK\_SIZE.  
CONF\_DOWNLOAD\_TIMEOUT\_NEXT.  
CONF\_DOWNLOAD\_BLOCK\_SIZE\_NEXT.  
CONF\_DOWNLOAD\_METODE.  
CONF\_DOWNLOAD\_IP\_ROUTING\_RECV\_BUFFER\_SIZE.  
flxCARDTransaction/flxAdministration/flxIdleIPforwarding now return FAILURE if internal DLL error (>=0x10000) encountered.  
IP routing problems with param and program download fixed.

## Revision

Better handling of tracefiles in case of communication failure, so we have data to look at.

Two new admin functions to get/set ip, port of external TeleDone server.

Problem with building Linux from source fixed.

Now we don't hang on a program download if money still on terminal.

Time format of PtFxDTr changed, so it matches flxComtrace.

Missing enums.

Better handling of TeleDone status.

### 3.5.02

Handling of Secure Hash to Electronic Receipts has been added.

Three new parameters can be used with flxSetConfiguration:

CONF\_GUARDTIME.

CONF\_RECEIPT\_WIDTH.

CONF\_PIP\_TIMEOUT.

ECR connected via IP can now drop the IP connection or reboot terminal on another port.

storebox.dk has been added as an electronic receipt.

Full terminal name is now displayed on terminal report.

Disconnect USB/serial: Try to get a new connection on the device in case of this error instead of just stopping.

### 3.5.01

Added Check Card.

Admin function that will initiate a check of the icc cardreader and will return OK if a card is detected in the reader.

To support prepaid scanned bar code in the terminal flexdriver (flxdrv.dll) now have a new callback to put the scanned bar code to the terminal.

See "Gavekort Scan Bar Code" document for further details.

If terminal is in system error FFF6 or 5000 the flexdriver will issue an Admin Update PSAM command – see 1.5.2.3 flxOpen function.

### 3.5.00

PSAM now controls all masking of carddata.

Callback to support Cashback amount so cardholder can receive cash – see "The setExtendedAmount call-back function".

Admin functions to get and set the terminals IP settings.

~~If terminal is configured with "Handling of Secure hash to Electronic Receipts" token can be received already in SetCardData.~~

### 3.4.02

Clarifying that no "no receipt" / "ingen kvittering" no longer will appear from terminal sw version 3.4.00.

### 3.4.00

Lock receipt discontinued, so no more "ingen kvittering" / "no receipt".

KeyEntry of transaction type.

## Revision

	Support for EIE.
3.3.00	Typos corrected, administrative functions added, setCardData mandatory.
3.1.00	Support for PSAM handled MSC track2 local cards.
3.0.00	Language support.
2.5.00	Version now follow the OCX version, new transaction types, new DCC menu command. CheckStopList now include status. Local cards not applicable.
2.3.00	New callback cardData(...) to enable transaction types key purchase, key refund, (key signature) and key authorization. EcrExtendedFunctions documentation.
2.2.10	New callback hostAdvice(...) to enable ECR applications to handle timeout of host advises.
2.2.01	flxConnect and flxOpen bug in return of value OPEN_NO_RECEIPT (0x13) GetReceipt for the _stdcall interface is added Asw1Asw2.
2.2.00	Added transaction types with tokens, Authorization (Original or Supplementary), Capture and Reversal (Authorization). Gratuity. Limit in first parameter in the pcbPrintReceipt callback function. Check Stop List function. flxIdleIPforwarding function. flxGetID function.

### Colorcode:

Additions since last release

Changes since last release

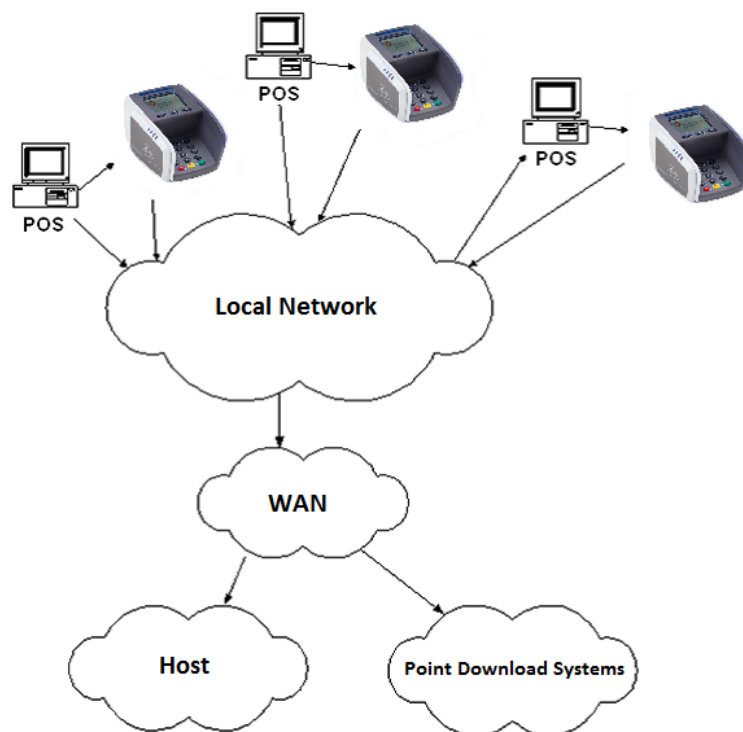
Removals since last release



## 1.2 Hardware Overview

The terminal and ECR can communicate using an RS232 serial cable, connected between the terminal's ECR port and a COM/serial port on the ECR.

For communication with acquire and Verifone Denmark the terminal can either have Ethernet cable connected or it can route IP communication via the RS232 cable through the ECR. The last option requires the terminal to be configured for "IP routing via RS232". The terminal is also capable of communicating to the ECR by TCP/IP. When communicating by TCP/IP, no COM cable is used, hence the ECR and terminal communicate by the local network.

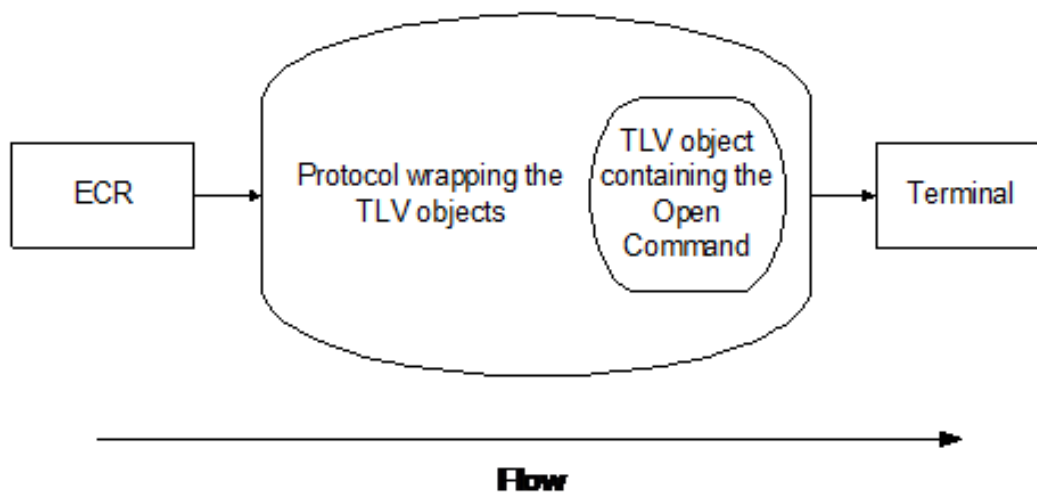


The figure shows an example of a normal setup. Only three YOMANI terminals are shown, but there is no limit to the number of terminals in the network.

### 1.3 Software Overview

The software interface is based on the ASN.1 TLV principle. This document describes a number of tags and defining the information the tag holds. All transferred data objects are encoded in BER TLV – and the document will give a description of the structure of the TLV data objects. Finally, the tags are wrapped in the KISS Protocol and for Ethernet communication in the TCP/IP Protocol.

The KISS Protocol is a serial byte oriented protocol, full duplex, asynchronous, and with no notion of master or slave. It is described elsewhere. Software delivered by Verifone Denmark A/S has been tested on Windows 7 (64 bit).



## 1.4 Functionality

The terminals functionality is fulfilling the CT-TRG-OTRS specifications which can be found at Nets website. It is recommended for the ECR developer to read the merchant section in the OTRS specification.

Terminal functions and options are configurable through parameter download from Verifone Denmark. Observe that functions and options are configured to support the wanted ECR functions, e.g. if the ECR uses token transaction types the terminal must be configured to handle these transaction types.

Please read the FAQ.pdf section “Important before a Certification”.

### 1.4.1 Initialisation

The initialization functions are used to make the terminal available to the users ECR. The functions must be initiated from the ECR. Make sure the terminal has finished it's boot sequence before issuing these commands, it cannot communicate to a connected or open command during the boot sequence.

#### 1.4.1.1 Connect

The connect command, can only be used in 'DISCONNECTED', 'CONNECTED', 'OPEN', and 'COM\_ERROR' state.

The ECR must send a 2-byte compatibility number to verify its software compatibility. The terminal replies with a 2-byte number. The first byte defines the software compatibility and the second byte defines software changes, which does not affect compatibility.

#### 1.4.1.2 Disconnect

The disconnect command, can be used in both 'CONNECTED' state and 'OPEN' state, e.g. just after the close command. The terminal display will change to 'Lukket'.

#### 1.4.1.3 Open

The open command changes the text in the terminal display to 'Terminalen er klar'. It can be used if the terminal is in 'CONNECTED', 'OPEN' and 'COM\_ERROR' state.

#### 1.4.1.4 Close

The close command changes the text in the terminal display from 'Terminalen er klar' to 'Velkommen'. It can only be used if the terminal is in 'OPEN' mode.

Observe that the order of these functions is Connect – Open ... application ... Close – Disconnect.

### 1.4.2 Transaction functionality

These functions are used to make transactions. If the transactions is keyentry added, the user must enter carddata using the terminal keys.



#### **1.4.2.1 PIN purchase transactions**

PIN based transaction can be initiated from either the terminal, by swiping a card, or from the ECR, by sending purchase information to the terminal. If initiated from the terminal, the user must enter the PIN code and wait for the amount from the ECR. If initiated from the ECR, the user is requested to swipe/insert a card, enter a PIN code and press 'Godkend'.

#### **1.4.2.2 Signature purchase transactions**

Signature based transactions must be initiated from the ECR. After initiating the transaction the user is requested to swipe/insert a card and approve the amount. This transaction type generates two receipts, unless host rejects the transaction or a communication error occurs. In this case only 1 receipt is generated. In case of successful host communication/verification, the first receipt is generated by the terminal and sent immediately to the ECR. The receipt must be printed, the user must sign and the operator must verify the signature. The second receipt is then generated based on the operator's decision (signature accepted/rejected). Signature verification is controlled by the PSAM.

#### **1.4.2.3 Signature refund transactions**

Refund transactions must also be initiated from the ECR. After initiating the transaction the user is requested to swipe/insert a card and (depending on PSAM decision) approve the amount. This transaction type generates two receipts, unless host rejects the transaction or a communication error occurs. In this case only 1 receipt is generated. In case of successful host communication/verification, the two receipts are generated by the terminal and immediately send to the ECR. Both receipts must be printed, and the operator must sign the customer receipt.

### **1.4.3 Administrative functionality**

The administrative functions are used to retrieve information from the terminal or set-up the terminal. All administrative functions can only be initiated from the ECR. Here are some of them.

#### **1.4.3.1 End of day functionality**

The end of day routine must be called at least one time every 24 hours. It is used to empty and balance the terminals Data store against the transaction inquirer host system. It is also used for PSAM updates.

#### **1.4.3.2 Unlock receipt functionality**

Unlock receipt request is used to fetch a copy of the last receipt from the terminal Data store if the terminal is locked in the NO\_RECEIPT state. Booting the terminal will not unlock the terminal. It is also not possible to make transactions in this state.

This command is not needed anymore since the terminal state "Locked in No Receipt" no longer exists.

#### **1.4.3.3 Terminal report**

The terminal report gives vital information on the terminal configuration, e.g. software and hardware version, communication module etc.

#### **1.4.3.4 Clock synchronization**

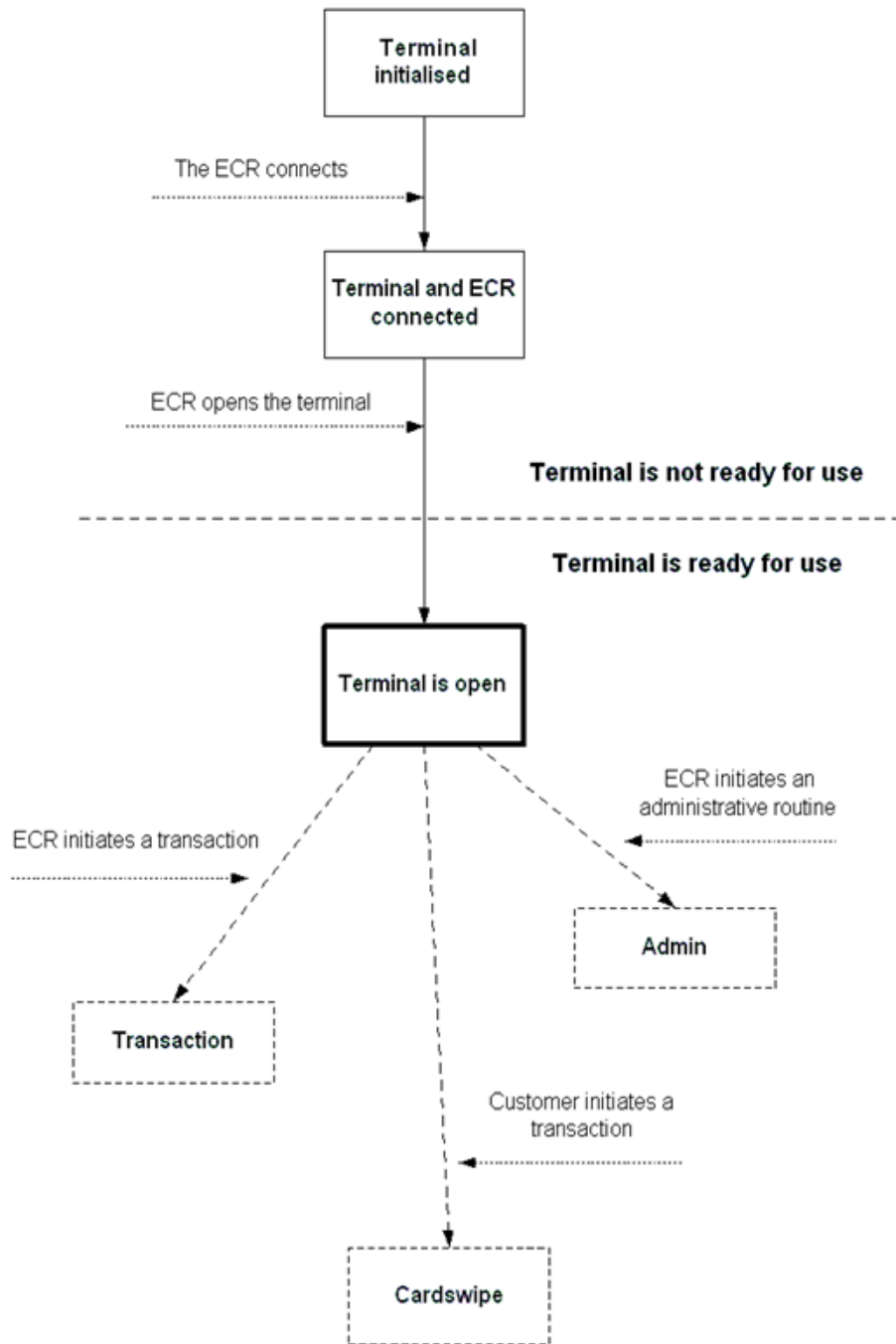
The clock synchronization functionality is very useful when testing the terminal communication capabilities, testing the connection to Nets and Verifone Denmark, or testing the connection in case of communication errors during transactions.

#### **1.4.3.5 Get DC Properties**

Get DC properties are used to get the status of a previous transaction. The PSAM (version  $\geq 50$ ) stores data of the last 8 approved transactions. This function will retrieve the data of a transaction if the search key (STAN) matches.

### **1.4.4 The Connect and Open procedure**

The terminal operates in many internal states, and it is only 'usable' for the customer and the ECR in the 'OPEN' state. Notice that sequence of connect and open. When ending terminal operation the application can issue the flxClose and flxDisconnect commands.



### 1.4.5 Verifone Error codes

In addition to NETS errorcodes (asw1asw2) the Flexdriver defines internal Verifone error codes starting at 0x10000. The current codes can be found in flex.h.

```
// ERROR CODES
typedef enum ERR_E
{
    ERR_UNKNOWN = 0x00010000,
    ERR_TERM_NOT_READY,
    ERR_NO_CONNECTION,
    ERR_NO_RECEIPT_NOT_USED,
    ERR_SW_NOTCOMPATIBLE,
    ERR_NO_LICENSE,
    ERR_TOKEN_TRANS,
    ERR_SYSTEM_ERROR,
    ERR_SPARE,
    ERR_STATE,
    ERR_TOKEN,
    ERR_IP_ROUTING,
    ERR_TRANSMISSION,
    ERR_STOP_TIMEOUT,
    ERR_TLV_NOT_ALLOWED,
} ERRORCODE_TYPE;
```

### 1.4.6 Timer event callback

A new feature in version 2.1.01 is the timer callback method. The DLL's transaction and communication 'layer' implements blocking timers, e.g. the KISS communication protocol on the serial port retransmits every four seconds if the terminal for some reason do not acknowledge a received frame, and worst case the DLL may block for 16 seconds. Enabling the timer event callback will allow the application to avoid total blocking.

### 1.4.7 File operation

Transferring files to and from the terminal is possible with the flxGetFile, flxGetFiles, flxPutFile and flxDeleteFile functions. Contact Verifone Denmark for more information of these functions. No longer supported.

### 1.4.8 Language support

The Flexdriver will use an ascii textfile – flxText.txt – for support of different languages, if the file is present in the install directory.

For more information, see 1.B Appendix.

## 1.5 The Flexdriver Interface

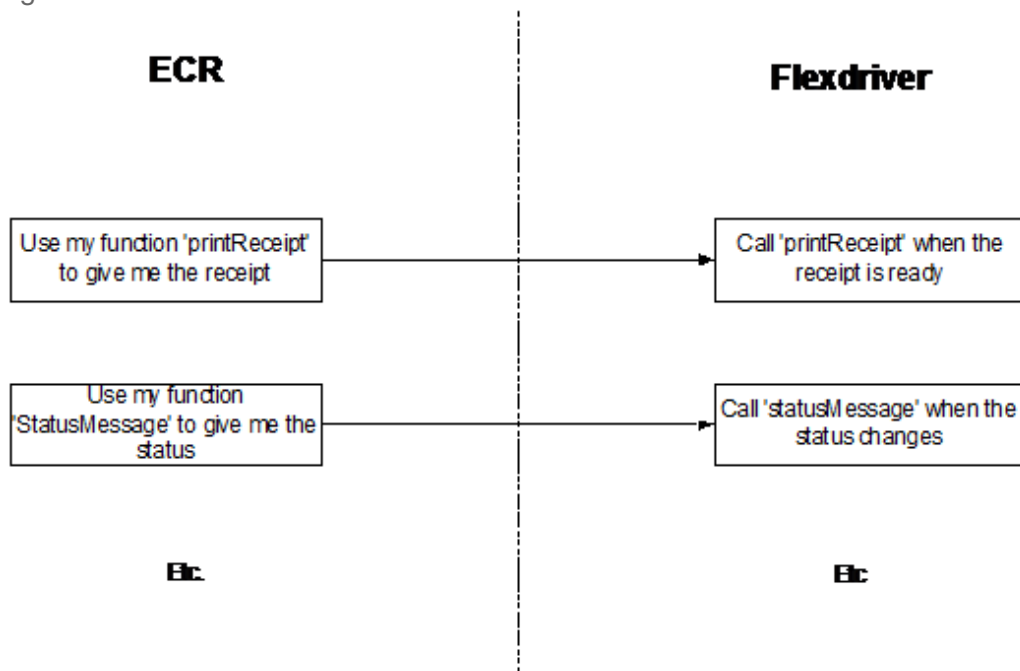
This section describes how to interface to the terminal using the Verifone YOMANI/XENTA Flexdriver. The Flexdriver is a software component developed and supplied by Verifone Denmaek A/S to simplify the integration for the ECR developer. The Flexdriver is programmed in C, and the following is available for the Flexdriver software package:

- MS Win 32 Bit DLL
- flex.h header file
- Documentation

### 1.5.1 Callback principle

The Flexdriver contains methods, which ensures that the synchronization and format used in the terminal is respected. The overall principle is based on 'callback' methods, which must be implemented in the software that interfaces with the Flexdriver.

So to use the Flexdriver, it is important that callback functionality is supported in the software interfacing to the Flexdriver.



As seen above, the callback function is defined with respect to the Flexdriver. This means that the Flexdriver is initialized with the callback functions that the interfacing software is using to process the respectively functionality.

### 1.5.1.1 Flexdriver function

This section describes the function used to setup the Flexdriver and initiate routines on the Terminal. For up to date information consult the latest flex.h file.

Using functions with Visual Basic 6, the WINAPI is using the **\_stdcall** naming convention. The function name preceded by an underscore ( \_ ) and followed by an at sign ( @ ) and the size of the functions arguments in bytes. No case translation is performed.

If not otherwise noted the flex functions will return SUCCESS = 0x00 if run successful, and FAILURE = 0x01 if an error is encountered during processing. Additionally an error code may be supplied.

## 1.5.2 Initialisation

### 1.5.2.1 flxInitCallback function

This function is used to setup the Flexdriver with the functions that the ECR uses to print receipt etc.

Synopsis

```
void flxInitCallback( FLX_CALLBACK methodId, void* methodPtr );
```

Parameter description:

methodId informs the Flexdriver on which connection to use the method methodPtr.

The possible values for method\_id are:

FLX_CALLBACK_SET_CARDDATA	Function which uses the PCI truncated card number
FLX_CALLBACK_PRINT_RECEIPT	Function to print the receipt
FLX_CALLBACK_DISPLAY_STATUS	Function to display status messages
FLX_CALLBACK_VERSIGCONFIRM	Function which verifies the signature
FLX_CALLBACK_ABORT	Function which aborts the transaction
FLX_CALLBACK_ADVICEFLAG	Function which uses advice flags
FLX_CALLBACK_ADVICELOG	Function to get a copy of terminal data store operations
FLX_CALLBACK_MENU	Function to display menus on the ECR display
FLX_CALLBACK_MENURESULT	Function to reply to selected menu
FLX_CALLBACK_SET_AMOUNT	Function which delivers the amount
FLX_CALLBACK_SET_AMOUNT_FEE	ECR calculates the fee in this function
FLX_CALLBACK_GET_AMOUNT_FEE	Terminal calculates fee and delivers it in this function
FLX_CALLBACK_SET_AMOUNT_GRATUITY	Function which delivers the gratuity amount
FLX_CALLBACK_GET_BINARY_RECEIPT	Terminal delivers a binary receipt
FLX_CALLBACK_EARLY_STAN_PAN	Terminal delivers transaction stan and PCI truncated pan
FLX_CALLBACK_TIMER	Function to avoid timer blocking
FLX_CALLBACK_PUT_TOKEN	Send a transaction Token to the terminal
FLX_CALLBACK_GET_TOKEN	Terminal delivers a transaction Token
FLX_CALLBACK_CHECK_STOPLIST	ECR checks Stop List and delivers a authorization code
FLX_CALLBACK_BREAK_IP	Break flxIdleIPforwarding
FLX_CALLBACK_CARD_DATA	Send carddata to terminal

Notice that callback methods must be initiated with flxInitCallback before the rest of the drivers methods can be used. None of the callback methods used as argument to flxInitCallback must be blocking (the functions must return promptly).

Example:

```
flxInitCallback( FLX_CALLBACK_SET_CARDDATA, setCardData );
flxInitCallback( FLX_CALLBACK_PRINT_RECEIPT, printReceipt );
flxInitCallback( FLX_CALLBACK_DISPLAY_STATUS, printStatus );
flxInitCallback( FLX_CALLBACK_ABORT, abortTransaction );
flxInitCallback( FLX_CALLBACK_VERSIGCONFIRM, verSigConfirmation );
flxInitCallback( FLX_CALLBACK_ADVICEFLAG, adviceFlag );
flxInitCallback( FLX_CALLBACK_MENU, showMenu );
flxInitCallback( FLX_CALLBACK_MENURESULT, resultMenu );
```

### 1.5.2.2 flxConnect function

#### Synopsis

```
int flxConnect( int EcrExtendedFunctions );
```

#### Parameter description:

EcrExtendedFunctions    bit 1 (0x02) if set preresult used  
                             bit 2 (0x04) if set extended trace information

#### Return values

Always returning SUCCESS = 0x00

### 1.5.2.3 flxOpen function

#### Synopsis

```
int flxOpen( void );
```

If terminal is in system error FFF6 or 5000 the Flexdriver will issue an Admin Update PSAM command.

#### Parameter description:

No parameters.

#### Return values

CONNECT_OK	0x00
OPEN_NO_RECEIPT	0x13
CONNECT_NO_INIFILE	0x01
CONNECT_INIFILE_READ_ERROR	0x02
CONNECT_COMPORTINITFAILURE	0x03
CONNECT_SW_NOTCOMPATIBLE	0x04
CONNECT_TERM_NOT_RESP	0x05
CONNECT_COMPORTOPEN	0x06
General flx error values	
DATALINKERROR	0x07
FUNCTION_NOT_POSSIBLE	0x08
TIMEOUT_IN_COMMUNICATION	0x09
CONNECT_NO_LICENCE	0x0A
CONNECT_INTERNAL_ERROR	0x0B
CONNECT_SYSTEM_ERROR	0x0C
CONNECT_KISS_ERROR	0x0D
CONNECT_TLV_NOT_ALLOWED	0x0E

### 1.5.2.4 flxClose function

#### Synopsis

```
int flxClose( void );
```

#### Parameter description:

No parameters



Return values

See 1.5.2.3 flxOpen function.

#### 1.5.2.5 flxDisconnect function

Synopsis

```
int flxDisconnect ( void );
```

Parameter description:

No parameters

Return values

See 1.5.2.3 flxOpen function.

#### 1.5.2.6 flxGetTerminalState function

Synopsis

```
int flxGetTerminalState ( void );
```

Cannot be used to detect close or disconnected states. The terminal has to be open to tell this.

Parameter description:

No parameters

Return value

Terminal state.

If state is 7 terminal is open and ready for new transaction or admin functions.

#### 1.5.2.7 flxGetFile, flxGetFiles, flxPutFile, flxDeleteFile functions

Description

It is possible for the ECR to fetch the terminal data store and configuration files. The function can only be used when the terminal is in IDLE mode, hence the ECR may not be 'connected'.

No supported anymore.

#### 1.5.2.8 flxSetConfiguration function

Synopsis

```
int flxSetConfiguration( int Func, ... );
```

Description:

```
enum CONF_E
```

```
{
```

```
    CONF_PARAM = 1,      // Placement of inifile
```

```
    CONF_TRACE,          // Enable trace
```

```
    CONF_EXTTRACE,        // Enable extensive tracing
```

```
    CONF_EXTTRACE_PLUS,   // Enable extensive tracing with append
```

```
    CONF_FILEPATH,        // Define directory path for flxdrv.ini and trace files
```

```

CONF_KISSPARAM,
CONF_NO_TRACE_AT_ALL=16,
CONF_RESETPORT,
CONF_GUARDTIME,
CONF_RECEIPT_WIDTH,
CONF_PIP_TIMEOUT,
CONF_DOWNLOAD_TIMEOUT,
CONF_DOWNLOAD_BLOCK_SIZE,
CONF_DOWNLOAD_TIMEOUT_NEXT,
CONF_DOWNLOAD_BLOCK_SIZE_NEXT,
CONF_DOWNLOAD_METODE,
CONF_IP_ROUTING_RECV_BUFFER_SIZE,
CONF_USE_GETTIMEOFDAY,
};

```

This function is used to setup the communication method in the Flexdriver.

If using CONF\_FILEPATH, this call to flxSetConfiguration must be called before any other flx functions.

Func: describes the additional parameters:

CONF\_PARAM – the following parameters defines comtype (RS232 or IP), port or IP address and baud or IP port.

CONF\_USERINPUT – the following parameters defines how the terminal is configured for user-input of accounttype, DCC or gratuity (ask Verifone Denmark).

Examples:

Example of how to set the Flexdriver to run RS232 with 19200 baud on COM port 1:

```
rc = flxSetConfiguration( CONF_PARAM, RS232_COMM, "COM1",19200 );
```

For Linux e.g. use:

```
rc = flxSetConfiguration( CONF_PARAM, RS232_COMM, "/dev/tty1",19200 );
```

Example of how to set the Flexdriver to run TCP/IP to a terminal with IP address 192.168.0.37 on port 2000:

```
rc = flxSetConfiguration( CONF_PARAM, ETHERNET_COMM, "192.168.0.37", 2000 );
```

An optional receiver timeout can be defined by adding a parameter (default is 500 ms).

```
rc = flxSetConfiguration(CONF_PARAM, ETHERNET_COMM, "192.168.0.37", 2000, 3000);
```

Example of how to define the file path to C:\(note that the path ends with '\\').

```
rc = flxSetConfiguration(CONF_FILEPATH, ETHERNET_COMM, "C:\\", 0);
```

Example of how to define user input:

```
rc = flxSetConfiguration(CONF_USERINPUT, 1, "1", 0);
```

Return values:

Return 0x00 for SUCCESS and 0x01 for FAILURE.

Three new parameters to be used with flxSetConfiguration.

CONF\_GUARDTIME is used to set the time before the DLL will initiate a new transaction/admin on the terminal. The OTRS states 500 ms and this is the default. Values lower than 500 are not accepted and will be set to 500.

Example of setting guard time to 750 ms:

```
flxSetConfiguration(CONF_GUARDTIME, 750, ComPortORIPAddress, 1);  
// CONF_GUARDTIME = 18
```

CONF\_RECEIPT\_WIDTH is used to set the width of the receipts printed. Values lower than 24 is not accepted, it will be set to 24.

Observe some admin functions use 48 chars per line, this will still be the case even if a value of 24 has been set.

Some admin functions center text, but the algorithm used doesn't handle this well, so it is advised to set the value to 24 before using the admin functions involved.

Example of setting receipt width to 48 chars:

```
flxSetConfiguration(CONF_RECEIPT_WIDTH, 48, ComPortORIPAddress, 1);  
// CONF_RECEIPT_WIDTH = 19
```

CONF\_PIP\_TIMEOUT used by Verifone Denmark to tweak an internal timeout, don't try to set this without consulting Verifone Denmark first. Default is set to 70 ms.

Currently OCX doesn't have support for this extra configuration.

ECR connected via IP can now drop the IP connection or reboot terminal on another port.

Handling of communication errors between ECR and terminal are improved, to minimize the need for reboot of ECR or terminal to get the next transaction/administrative function started.

At boot the terminal listen for a new connection from the ECR, RS232/USB or IP. If an IP connection is made, terminal will begin listen on port 2001 for a command, to either drop the IP connection or to reboot.

The command can be sent via telnet but it have to be completed within 10 s, to be successful, and only if the right data has been sent.

Where XXXXXX is the terminal identification

To drop the IP connection send

DropIp:00XXXXXX

To reboot the terminal send

Reboot:00XXXXXX

If the command is accepted by the terminal, it will respond with:

Dropping IP port 2000 on terminal 00XXXXXX

or

Restarts terminal 00XXXXXX – Wait 1 – 2 min.

The flxdrv.dll has a new function to send the above commands.

Function:

```
int flxDropIp(char *ip_addr, char *terminal_ident, int to_do);
```

Tell the terminal to drop IP connection, making it ready for a new connect:

```
Input:  ip_addr  Terminals IP address  
        ex. 10.0.0.156  
        terminal_ident  Terminal identifier  
        ex. 00990768 Must be 8 digits
```

```
to_do Action terminal has to perform
0: DropIp Dropping terminals IP connection
1: Reset Restarts terminal
```

```
Output: 1 - SUCCESS - OK
        0 - FAILURE - not OK
```

See Appendix 1.E Drop IP listing

To optimize ADMIN\_DOWNLOADPROGRAM with IP routing it is now possible to change the behavior by setting some values using flxSetConfiguration, the default values now work on all the platforms we tried on.

Consult Verifone Denmark ([development.hrv@verifone.com](mailto:development.hrv@verifone.com)) if problems with ADMIN\_DOWNLOADPROGRAM with IP routing should occur.

#### CONF\_DOWNLOAD\_TIMEOUT

Timeout on first packet send to terminal

Default 125 ms

#### CONF\_DOWNLOAD\_BLOCK\_SIZE

Block size on the first packet send to terminal

Default 1000 bytes

#### CONF\_DOWNLOAD\_TIMEOUT\_NEXT

Timeout on all remaining packets send to terminal

Default 25 ms

#### CONF\_DOWNLOAD\_BLOCK\_SIZE\_NEXT

Block size on all remaining packets send to terminal

Default 1000 bytes

#### CONF\_DOWNLOAD\_METODE

Method used for ADMIN\_DOWNLOADPROGRAM

0. use timeout on first block only

1. use timeout on every block

2. No timeout

3. Auto DLL adjust block size and timeout

Default is set to 3.

#### CONF\_IP\_ROUTING\_RECV\_BUFFER\_SIZE

Size used with ADMIN\_DOWNLOADPARAM

Default 8191 bytes

#### CONF\_USE\_GETTIMEOFDAY

Our Linux LIB previous used time() for some internal timers giving a resolution of 1 second, now we default use gettimeofday() for a millisecond resolution, like the Windows DLL.

If you want to go back to the old 1 second resolution timer, you can use the flxSetConfiguration(CONF\_USE\_GETTIMEOFDAY) to disable.

Example:

```
flxSetConfiguration(CONF_USE_GETTIMEOFDAY, 0, ComPortORIPAdress , 1);
```

A problem seen on Linux LIB with the 1 second timer: You are waiting for CardSwipe and the BreakIP is only called once every second, giving a delay before you can start the transaction after a card has been swiped/inserted, with the new timer, the BreakIP is called every 50ms

### 1.5.2.9 flxSetTrace functionality

#### Synopsis

```
void flxSetTrace( int traceLevel );
```

#### Description

There are four levels of tracing (defined in flex.h). All four levels generate text files located in the same directory as the Flexdriver.

Make sure ECR has backup algorithm for the trace files, so they don't grow to a size slowing the transactions, you are unable to send trace files for analysis, depended on the number of transactions copy/compress daily/weekly.

The low level communication on RS232/USB/Ethernet is written in: flxComTrace.txt.

The next level DLL commands, results etc. are written in: PtFxDrTr@VVVV.txt, where VVVV is the current DLL version number.

Files are usually found in the same folder as the flxdrv.dll.

CONF_PARAM = 1	Placement of inifile Not supported anymore
CONF_TRACE	Enable trace
CONF_EXTTRACE	Enable extensive tracing
CONF_EXTTRACE_PLUS	Enable extensive tracing with append
CONF_NO_TRACE_AT_ALL	= 16

### 1.5.2.10 flxGetTrace functionality

#### Synopsis

```
int flxGetTrace( void );
```

## 1.5.3 Transaction

### 1.5.3.1 flxCardTransaction function

#### Synopsis

```
int flxCardTransaction( FLX_TRANSTYPE transactionType,  
    FLX_AMOUNT amount,  
    int currencyCode,  
    unsigned char merchantInitiative,  
    const unsigned int *refno,  
    unsigned int *error );
```

#### Description:

flxCardTransaction is the method to use when initiating a 'Purchase' on the Flex Terminal.

transactionType: Describes the transaction type, possibilities are:

FLX_TRANS_PURCHASE	Debit transaction
FLX_TRANS_REFUND	Credit transactoin
FLX_TRANS_ORIGINAL_AUTH	Original authorization, outputs a token
FLX_TRANS_SUPL_AUTH	Token based transaction
FLX_TRANS_CAPTURE	Token based transaction
FLX_TRANS_REVERSAL_AUTH	Token based transaction
FLX_TRANS_PREPAID_PURCHASE	Debit prepaid transaction
FLX_TRANS_PREPAID_REFUND	Credit prepaid transaction
FLX_TRANS_PREPAID_ORIGINAL_AUTH	Orig auth prepaid transaction, no token output
FLX_TRANS_PREPAID_SCAN_PURCHASE	Debit prepaid scan transaction
FLX_TRANS_PREPAID_SCAN_REFUND	Credit prepaid scan transaction
FLX_TRANS_PREPAID_SCAN_ORIGINAL_AUTH	Orig auth prepaid scan transaction, no token output
FLX_TRANS_CANCELLATION	
FLX_TRANS_CARDCHECK	Card check
FLX_TRANS_POST_PURCHASE	Post Purchase Request
FLX_TRANS_POST_REFUND	Post Refund Request,

### **Amount**

Describes the size of the amount in the smallest unit (øre for DKK). It must always be a positive value, also at refund transactions. The amount can also be delivered to the Flexdriver via the setAmount callback routine. If the setAmount routine is used, the amount in the flxCardTransaction function is ignored by the Flexdriver. It is also possible to use the getExtendedAmount callback routine.

### **CurrencyCode**

Describes the currency code according to "ISO 4217", e.g. DKK = 208 and EUR = 978.

### **MerchantInitiative**

Is compatible with "Merchant Initiative" in OTRS document, it is important to understand that it is only a request to perform a transaction in a specific way, the request may be denied by card or PSAM.

It could e.g. be relevant to request "SIGNATURE" if cardholder has forgotten the PIN code and wants to use signature instead or to request "OFFLINE" if the network is down.

MI\_PIN = 0x00 (is actually "DEFAULT" which in most cases is PIN)  
MI\_ONLINE = 0x50  
MI\_OFFLINE = 0x60  
MI\_FORCED\_CVM = 0x80  
MI\_FORCED\_PIN = 0x81  
MI\_FORCED\_SIGNATURE = 0x82

Values can be combined by "Bitwise OR" so "FORCED\_OFFLINE\_SIGNATURE" would be:  
01100000|10000010 = 11100010 (( 0x60|0x82 = 0xE2 ))

If the type KEY\_ENTRY is added to the transactionType, the terminal will ask for carddata - cardnumber, expiration date and card verification number.

Using the FLX\_TRANS\_PREPAID\_SCAN\_PURCHASE transaction type, it's now possible to start a Swipp transaction to a specific phone number given as a cardnumber like:

**sw:45PPPPPPPP** where sw tells it's a Swipp transaction, the 45 is the country code without the normal plus, and the P's are the phone number digits going to be used for the transaction. Other country codes - shorter/longer phone numbers are allowed as long the maximum length of the total string passed is 19 char or below. White space in the end is not allowed.

### Refno parameter

Can be used for tracking transactions on the terminal. The terminal does not change the refno, so it's up to the ECR to keep track of transactions.

### Error

A pointer to an error code, if any. The value can be all the values defined in the OTRS specification, e.g. 1017 for wrong PIN code. Some of the error codes starting at 0x10000 are defined by Verifone Denmark. The current definitions can be found in the flex.h file.

Return values for Nets transactions:

0x00	Transaction approved
0x01	Transaction is not approved (rejected, communication error or aborted)

Return values for Local Card transactions (use flxCompleteExtCardTransaction to complete)

0x10	Transaction approved
0x11	Transaction is not approved (rejected, communication error or aborted)

### 1.5.3.2 flxCompleteExtCardTransaction function

#### Synopsis

```
int flxCompleteExtCardTransaction( int result )
```

Description:

This function is used to complete a Local Card Transaction.

The result parameter must be 0x00 to approve the transaction and 0x01 to reject the transaction.

Return values

Return 0x00 for SUCCESS and 0x01 for FAILURE

Description

This function returns the current trace level

### 1.5.3.3 flxGetID function

Synopsis

```
int flxGetID(void);
```

Description

This function returns the terminal ID as an integer, e.g. 990125. Return value: Terminal ID as integer.

### 1.5.3.4 flxGetSetExtendedEcrFunctions function

Synopsis

```
int flxGetSetEcrExtendedFunctions(char command, int * value);
```

Description

This function sets or returns the ECR Extended Function value. For more information of ECR Extended Functions consult Verifone Denmark.

Command:           Set 0x84, set to value  
                    Get 0x85, get value

Return value:       0 if success  
                    Non 0 if failure

value:

Bit

- |    |        |   |
|----|--------|---|
| 0. | 0x0001 | No PTAG_INFO during transaction                     |
| 1. | 0x0002 | PRERESULT as TLV                                    |
| 2. | 0x0004 | DebugInfo SEQNO, EVENT and FROMSTATE                |
| 3. | 0x0008 | Pre-connect to host                                 |
| 4. | 0x0010 | Print only PIN transactions in transaction-log      |
| 5. | 0x0020 | Print only approved transactions in transaction-log |
| 6. | 0x0040 | Opel/Seb solution - No longer valid                 |
| 7. | 0x0080 | Reply spaces in authcode on CheckStopList           |
| 8. | 0x0100 | Card inserted signal to ECR                         |
| 9. | 0x0200 | EMV 4.3a PDOL early amount needed                   |

### 1.5.3.5 flxTerminalProperties function

Synopsis

```
char * flxTerminalProperties (int command, char *str, size_t length, uint *error);
```

Description



Perform terminal properties (get or set commands). More info from Verifone Denmark.

command: ADMIN\_PROPS\_GET - 1: str is output of terminal settings  
 ADMIN\_PROPS\_PREPECEIPT - 2: str is input with amount in ascii for Prereceipt  
 ADMIN\_PROPS\_DATASTORE - 3: str is input for datastore cleaning  
 ADMIN\_PROPS\_SET - 4: str is input param for flxAdministration function  
 ADMIN\_PROPS\_ADVICERECON - 5: str is input for advice reconciliation report  
 ADMIN\_PROPS\_BATCH - 6: str is input for setting batch number  
 ADMIN\_PROPS\_GET\_LABELS - 7: str is empty  
 ADMIN\_PROPS\_EVENTLOG\_SEND - 8: str is syslogserverip for send eventlog  
 ADMIN\_PROPS\_EVENTLOG\_DELETE - 9: str is syslogserverip for send and remove eventlog  
 ADMIN\_PROPS\_SETIP\_SETTINGS - 10: str is input for setting terminals IP  
 ADMIN\_PROPS\_GET\_EXT\_LABELS - 11: str is empty  
 ADMIN\_PROPS\_SETTELEDONE\_SETTINGS - 12: str is input for setting IP/Port to server going to be called after teleload has been done on terminal  
 ADMIN\_PROPS\_USER\_INPUT - 13: str is input for  
 <min>,<max>,<textid1>,<textid2>,<timeout> e.g. "1,12,550,551,60"  
 user can enter minimum 1 and max 12 digits  
 and display textid 550 and 551 (contact Verifone for valid txtid)  
 within time of 60 seconds.  
 Return entered digits error is 0 or null and error is 0 or ERR\_STOP\_TIMEOUT

Ex. If you make a marketing survey, where you want to ask the customer for the postal code, always 4 digits long (DK), to avoid delaying the transaction for a long periode the timeout is set to 30 seconds.

You send:

4,4,textid1,textid2,30

The screen could look something like:

Your Company Name  
 Enter postal code:

\_\_\_\_\_

str – pointer to csv ascii string

1: -

2: <AMOUNT>,<CURRENCYCODE>

3: <PSAMID>,<RECORDID>,<FILENUMBER>,<STAN>

4: <PARAM1>,<PARAM2>,...,<PARAMX>

5: <ADVICE\_FILE\_INDEX>

6: <BATCHNR1>,<CURRENCY1>,...,<BATCHNRX>,<CURRENCYX>

7:

8: <SYSLOGIP>

9: <SYSLOGIP>

10:<IP >,<SUBNET >,<GATEWAY>,<DNS1>,<DNS2>,<DOMAIN >

11: 42

12:<IP >,<Port >

13:<MIN >,<MAX >,<TEXTID1 >,<TEXTID1 >,<TIMEOUT >

length            length of string str

Return:           null if command not valid or str too short or command fail.

1: pointer to ascii csv str including properties

<TERMINALID>,<RECEIPTTYPE>,<GRATUITYMODEL>,<DCC>,<TOKEN>,<PREPAID>,<KEYENTRY>,<OFFLINE>,<IP ROUTING>,<MERCHANTNUMBER>,<MERCHANTNAME>,<MERCHANTCITY>,<MERCHANTADDRESS>,<MERCHANTZIP>,<MERCHANTPHONE>,<MERCHANTBRN>,<FLXTRACELEVEL>,<LANGUAGE>,<VAT>,<USERINPUT>,<COUNTRYCODE>,<DEFAULTCURRENCY>,<CONTACTLESSFLAGS>,<TERMINALTYPE>,<DCCMARKUP>,<TCS>,<SOFTWARE\_VERSION>,<CDP>,<PSAM\_CDP>,<HARDWARENAME>,<EIE>,<PSAM\_EIE>,<ISSUER\_ENVELOPE\_NON\_EMV\_SIZE>,<ISSUER\_ENVELOPE\_EMV\_SIZE>,<TOTAL\_ISSUER\_ENVELOPE\_NON\_EMV\_SIZE>,<TOTAL\_ISSUER\_ENVELOPE\_EMV\_SIZE>,<ERECEIPT\_SCHEMES>,<PSAM\_VERSION>

See Appendix 1.C Terminal Properties

2 – 6: str or text NULL

8 – 9:

### 1.5.3.6 flxGetDCProperties functionality

#### Synopsis

```
int flxGetDCProperties( int stan, STAN *pStan, DC_RESULT *dcRes );
```

#### Description

This function returns the status and data (DC\_RESULT) of previous approved transactions stored in the PSAM. This function is implemented as separate Administration functions.

## 1.5.4 Administration

### 1.5.4.1 flxAdministration functions

```
int flxAdministration( FLX_ADMIN_FUNCTION func, unsigned int *error );
```

This function is used to initiate administrative functions on the terminal. The flex.h file encloses an enumerated list of the functions. Each adminFunction definition (see flex.h) is described in the following sections.

Error is a pointer to an error code, if any. The value can be all the values defined in the OTRS specification, e.g. 0xFFF3 for system error. Some of the error codes are defined by Verifone Denmark. The definitions can be found in the flex.h file:

#### Error codes

ERR\_TRANSMISSION added.

```
// ERROR CODES
typedef enum ERR_E
{
    ERR_UNKNOWN = 0x00010000,
    ERR_TERM_NOT_READY,
    ERR_NO_CONNECTION,
    ERR_NO_RECEIPT_NOT_USED,
    ERR_SW_NOTCOMPATIBLE,
    ERR_NO_LICENSE,
    ERR_TOKEN_TRANS,
    ERR_SYSTEM_ERROR,
    ERR_SPARE,
    ERR_STATE,
    ERR_TOKEN,
    ERR_IP_ROUTING,
    ERR_TRANSMISSION,
    ERR_STOP_TIMEOUT,
    ERR_TLV_NOT_ALLOWED,
} ERRORCODE_TYPE;
```

Return values are SUCCESS or FAILURE, with error set.

If terminal is has IP routing enabled, the flxAdministration(ADMIN\_DOWNLOADPROGRAM) will now call the flxIdleIPforwarding (1 to enable IP routing) and the flxAdministration wait for the teleload to be perform before it return, with the status of the download. The result of the flxAdministration will indicate the teleload was done, and error will be set to TeDo = 0x5465446f.

A display message during the teleload will tell the status of the download, the same as the message send to the server below.

If BreakIP callback has been set, the sync will have the value TeDo = 0x5465446f and you should set the result to 1 to break the flxIdleIPforwarding. If not this is handled by the DLL internally.

If the terminal is configured to run with the terminals build in Ethernet, it's possible to have a TeleDone entry in the terminal with information about an external server / port to be called by the terminal after a program download has been done.

The flxAdministration(ADMIN\_DOWNLOADPROGRAM) will return and connection to the terminal will be lost, so one have to either wait a long time or get a signal from the external server telling the teleload has been done.

The external server will receive a message and the connection closed, no reply expected.

The message contains a html line:

<PointSay>text2 = "TELELOAD FÆRDIG" text3 = " SUCCESS - 01"</PointSay>

or on error ex.

<PointSay>text2 = "TELELOAD FEJL" text3 = " ERROR - 5a"<PointSay>

#### 1.5.4.2 Endofday (balancing/clearing) routines

Value

ADMIN\_ENDOFDAY

Description

This value will initiate a balancing routine on the terminal and prints a transaction report on the Merchant Unit Printer. Endofdaylog will also deliver the log, which is a short summary of each transaction. The endofday/endofdaylog must be called (either by the person operating the Merchant Unit or automatically) at least once every 24 hours. This routine includes emptying of the terminals transactions database against the host (Nets), plus an update of the PSAM. Notice that the status can be approved even though the terminal does not connect to Nets. NB: If the ECR does not log every transaction electronically, the ADMIN\_ENDOFDAYLOG must be used.

Note that in some error situations, the line length of the transaction record is twice the length of the regular reports, (48 chars. vs. 24).

#### **1.5.4.3 Endofdaylog routine**

Value

ADMIN\_ENDOFDAYLOG

Description

If the ECR does not log every transaction electronically, the ADMIN\_ENDOFDAYLOG must be used.

This value will initiate the same routine as the ADMIN\_ENDOFDAY. In addition it will attach a record for all balanced transactions. Note that the line length of the transaction record is twice the length of the regular reports, (48 chars vs. 24).

#### **1.5.4.4 Terminal report routine**

Value

ADMIN\_REPORT\_TERMINALREPORT

Description

This function fetches the terminal report. The terminal report includes vital information regarding the terminal SW version, communication card etc.

#### **1.5.4.5 Transaction totals report**

Value

ADMIN\_REPORT\_TOTALS

Description

This function fetches the transaction totals in the terminal.

#### **1.5.4.6 Transaction log report**

Value

ADMIN\_REPORT\_LOG

Description

This function fetches the complete transaction log in the terminal.

#### **1.5.4.7 Old log routine**

Value

## ADMIN\_REPORT\_OLDLOG

### Description

The ADMIN\_REPORT\_OLDLOG fetches the summary of transaction and the log from the previous endofday.

### 1.5.4.8 Receipt functions

#### Values

ADMIN\_LASTRECEIPT and ADMIN\_UNLOCK\_RECEIPT

#### Description

ADMIN\_UNLOCK\_RECEIPT is used to unlock terminal if it is locked in "NO RECEIPT" state.

NOTE: This state no longer appears from version 3.4.00 and the function will not be needed.

ADMIN\_LAST\_RECEIPT will fetch the stored receipt (if any) from the terminal. During normal operation, the receipts are cleared from the terminal memory when a new transaction is initiated (a card is inserted/swiped or flxCardTransaction is called).

### 1.5.4.9 Clock synchronization routines

#### Value

ADMIN\_CLOCKSYNCPnets or ADMIN\_CLOCKSYNCPPOINT

#### Description

These routines synchronize the clock in the terminal. The routines can be used to 'test' the connection to Nets and Verifone Denmark and are very useful during terminal setup. The clock synchronization to Verifone Denmark will set the time, date and year. The clock synchronization to Nets will only set the time and date.

### 1.5.4.10 Download routines

#### Values

ADMIN\_DOWNLOADPARAM, ADMIN\_DOWNLOADTLCMDB,  
ADMIN\_DOWNLOADPROGRAM, ADMIN\_UPDATEFEETABLE,  
ADMIN\_UPDATEPSAM, ADMIN\_UPDATEDCCRATES, ADMIN\_GETSALT

#### Description

These functions are used to download files to the terminal.

ADMIN\_DOWNLOADPROGRAM is used to fetch the terminal software. This software is common to all YOMANI/XENTA terminals.

ADMIN\_DOWNLOADPARAM is used to fetch terminal specific parameter files.

ADMIN\_DOWNLOADTLCMDB is used to get the communication file.

ADMIN\_UPDATEPSAM is used (with IP routing) to update the PSAM.

ADMIN\_UPDATE\_DCCRATES updates the DCC rates.

ADMIN\_GETSALT is used with "Handling of Secure hash to Electronic Receipts".

### 1.5.4.11 Contrast routines

#### Values

ADMIN\_CONTRASTUP or ADMIN\_CONTRASTDOWN

Description

These function are used to increase or decrease the contrast level in the terminal display.

#### **1.5.4.12 Clear data store routine**

Value

ADMIN\_CLEARDATASTORE

Description

This function clears the terminal data store, which is:

- Any stored receipt(s)
- Logs and reports (current and earlier)
- Any pending PSAM update replies
- All unbalanced transaction data

This function must be password protected to avoid clearing the terminal data store unintentionally.

#### **1.5.4.13 Restore TLCMDB routine**

Value

ADMIN\_ADMIN\_RESTORETLCMDB

Description

This function can be used to restore the communication file (TLCMDB) to default settings.

#### **1.5.4.14 Restart terminal routine**

Value

ADMIN\_RESTARTTERMINAL

Description

This function restarts the terminal. The terminal only accepts this function in OPEN mode. Communication with the terminal ends, and must be reinitialized.

#### **1.5.4.15 Send log routine**

Value

ADMIN\_SENDLOG

Description

This function sends the terminal data store and logfiles to Verifone Denmark Server System.

#### **1.5.4.16 Eject card routine**

Value

ADMIN\_EJECTCARD

Description

This function can be used to force an eject of a card in the motorized card reader attached to the SPIN terminal. The function is not used in conjunction with YOMANI/XENTA terminals.

#### **1.5.4.17 Print msc routine**

Value

ADMIN\_MSC

Description

This function is used to fetch the card range table in the terminal. This function is for future use.

#### **1.5.4.18 Backlight on routine**

Value

ADMIN\_BACKLIGHT\_ON

Description

This function is used to turn on the terminal light.

#### **1.5.4.19 Backlight off routine**

Value

ADMIN\_BACKLIGHT\_OFF

Description

This function is used to turn off the terminal light.

#### **1.5.4.20 Network Report**

Value

ADMIN\_NETWORK\_REPORT

Description

This function is used to check the network and print a report.

#### **1.5.4.21 Rates Report**

Value

ADMIN\_RATES\_REPORT

Description

This function is used to print a DCC currency rates report.

#### **1.5.4.22 Exclude datastore record with specific stan routine**

Value

ADMIN\_EXCLUDE\_DATASTORE\_RECORD\_WITH\_STAN

Description

Use `flxTerminalProperties( ADMIN_PROPS_SET )` to set the STAN and use `flxTerminalProperties(ADMIN_EXCLUDE_DATASTORE_RECORD_WITH_STAN )` to remove a faulty advice from file5.

Alternatively you could use `flxTerminalProperties( ADMIN_PROPS_DATASTORE )` to set the STAN and call the `flxAdministration( ADMIN_EXCLUDE_DATASTORE_RECORD_WITH_STAN )` in one call.



#### **1.5.4.23 Advice Forwarding routine**

Value

ADMIN\_ADVICEFORWARDING

Description

This value will initiate an emptying of the Terminals transactions database against the host (Nets).  
No data will be printed.

#### **1.5.4.24 Report file5 Status routine**

Value

ADMIN\_REPORT\_FILE5STATUS

Description

This value will initiate printing of File5 Status report on the Merchant Unit Printer.

#### **1.5.4.25 Report Advice Reconciliation Report routine**

Value

ADMIN\_GETADVICERECON

Description – need parameter:

Use flxTerminalProperties( ADMIN\_PROPS\_SET to set the parameter ) and call flxAdministration ( ADMIN ADMIN\_GETADVICERECON )

or

Use flxTerminalProperties( ADMIN\_PROPS\_ADVICERECON to set the parameter and automatic call flxAdministration ( ADMIN ADMIN\_GETADVICERECON )

Parameter is ASCII string with number 0 to 12.

0 – To get advice reconciliation report on current advice file.

1 – To get advice reconciliation report on advice file after one end of day (EOD)

2 – To get advice reconciliation report on advice file after 2 EOD's.

12 – To get advice reconciliation report on advice file after 12 EOD's

#### **1.5.4.26 Set Batch Number routine**

Value

ADMIN\_SET\_BATCH\_NUMBER

Description

Need parameter:

Use flxTerminalProperties( ADMIN\_PROPS\_BATCH, "" ) to set the parameter

To set a batch number use this format:

CURR, NAME

for parameter.

Where CURR is one of the texts

DKK, ISK, JPY, NOK, SEK, CHF, GBP, USD, EUR

And NAME is a text up to 12 characters long.

To go back to default – terminal selects batch numbers – set the first 4 characters of parameter to AUTO.

#### **1.5.4.27 TCS report routine**

Value

ADMIN\_REPORT\_TCS

Description

Contact Verifone Denmark.

#### **1.5.4.28 get Salt**

Value

ADMIN\_UPDATESALT

Description

Update salt value for Handling of Secure hash to Electronic Receipts – contact Verifone Denmark.

#### **1.5.4.29 Print the Event report**

Value

ADMIN\_EVENTLOG\_PRINT

Description

Print the PCI eventlog report.

#### **1.5.4.30 Send the Event report**

Value

ADMIN\_EVENTLOG\_SEND

Description

Send the PCI eventlog to a syslog server UDP/514, with IP address that you provide.

Terminal must not use IP routing, to get the eventlog in case of IP routing, ask Verifone Denmark to change the terminals setup to NO IP routing, make a param download, connect the terminal to the network with an Ethernet cable, make the ADMIN EVENTLOG SEND, ask Verifone Denmark to change back the terminals setup to IP routing, make a param download, remove the Ethernet cable from the terminal.

#### **1.5.4.31 Delete the Event report**

Value

ADMIN\_EVENTLOG\_DELETE

#### Description

First we try to send the PCI eventlog to a syslog server UDP/514, with IP-address that you provide, next we delete the PCI eventlog. Could be used if the terminal is short of space and you do not want to keep the eventlog.

Terminal must not use IP routing, to get the eventlog in case of IP routing, ask Verifone Denmark to change the terminals setup to NO IP routing, make a param download, connect the terminal to the network with an Ethernet cable, make the ADMIN EVENTLOG SEND, ask Verifone Denmark to change back the terminals setup to IP routing, make a param download, remove the Ethernet cable from the terminal.

#### **1.5.4.32 The TPROPS CVS report routine**

##### Value

ADMIN\_REPORT\_TPROPS\_CVS

##### Description

Get and save the Terminal Properties used in the flexDriver to control Extended Issuer Envelope (EIE). Format is the same as in flxTerminalProperties.

#### **1.5.4.33 Get IP Settings**

##### Value

ADMIN\_GETIP\_SETTINGS

Get the terminals IP settings.

Format: IP;SUBNET;GATEWAY;DNS1;DNS2;DOMAIN;DHCP

#### **1.5.4.34 Set IP Settings**

##### Value

ADMIN\_SETIP\_SETTINGS

Sets the terminals IP settings.

Format: IP;SUBNET;GATEWAY;DNS1;DNS2;DOMAIN;DHCP

#### **1.5.4.35 Download Images**

##### Value

ADMIN\_DOWNLOAD\_IMAGES

##### Description

Download images from Verifone Denmark's host to terminal. For more info see Appendix 1.G.

#### **1.5.4.36 Check Card**

##### Value

ADMIN\_CHECK\_CARD

##### Description

After a successful transaction this admin can be called repeatedly until the customer has removed the card from the ICC reader in the terminal. A minimum of 500 ms should pass before calling ADMIN\_CHECK\_CARD.

This value will initiate a check of the ICC card reader and will return OK if a card is detected in the reader.

Use flxIdleIPforwarding(3) to wait for card swipe.

Return

If card in reader, SUCCESS (0) is returned, is card removed FAILURE (1) is returned.

#### 1.5.4.37 Get TeleDone

flxAdministration(ADMIN\_GETTELEDONE\_SETTINGS)

Will produce a receipt like this

```
*****
```

```
TELEDONE Settings Rapport
```

```
*****
```

```
IP;10.0.0.123
```

```
PORT;2001
```

```
*****
```

or if nothing set

```
*****
```

```
TELEDONE Settings Rapport
```

```
*****
```

```
IP;
```

```
PORT;
```

```
*****
```

#### 1.5.4.38 Set TeleDone

flxAdministration(ADMIN\_SETTELEDONE\_SETTINGS)

If TCP/IP communication used and you run the administrative function

ADMIN\_DOWNLOADPROGRAM, the terminal will try to send a message to a server after the terminal has been updated (Teleload has been done)

ADMIN\_SETTELEDONE\_SETTINGS will set the IP address, Port of the server to be called, first you must set the IP address, Port using flxTerminalProperties( ADMIN\_PROPS\_SET )

you can use the

flxTerminalProperties( ADMIN\_PROPS\_SETTELEDONE\_SETTINGS) to set the IP,Port and a call to flxAdministration(ADMIN\_SETTELEDONE\_SETTINGS) is done before flxTerminalProperties exits. (Only one call needed)

The IP,Port has to be in this format:

IP,Port

Ex. Server has IP: 10.0.0.123 and listen on Port: 2001

10.0.0.123,2001

A receipt is printed like this

\*\*\*\*\*

Set TELEDONE Settings Rapport

\*\*\*\*\*

Success - TeleDone selected

IP;10.0.0.123

PORT;2001

\*\*\*\*\*

To remove the entry use only the comma (both IP and Port empty)

,

and the receipt will look like this

\*\*\*\*\*

Set TELEDONE Settings Rapport

\*\*\*\*\*

Success - TeleDone TLCMDB entry removed

\*\*\*\*\*

The message send to the server from the terminal will look like this

<PointSay> text2="TELELOAD END" text3=" SUCCESS - 00"</PointSay>

in case of no download or error

<PointSay> text2="TELELOAD ERROR" text3=" ERROR - 01"</PointSay>

#### **1.5.4.39 Short Contactless Terminal report routine**

Value

ADMIN\_REPORT\_CTLREPORT\_SHORT

Description

This function makes report with contactless limits of last transaction. Limits are used to configure contactless verification tool.

#### **1.5.4.40 User input Terminal report routine**

Value

ADMIN\_USER\_INPUT

Description

Contact Verifone Denmark.

#### **1.5.4.41 Obsolete Terminal report routine**

Value

ADMIN\_SWIPP\_ID

Description

Obsolete - Don't use

#### 1.5.4.42 Long Contactless Terminal report routine

Value

ADMIN\_REPORT\_CTLREPORT\_LONG

Description

This function makes report with contactless limits and all data of last transaction. Very long about 1500 lines.

### 1.5.5 Idle/IP

#### 1.5.5.1 flxIdleIPforwarding (flxIdle) function

Synopsis

int flxIdleIPforwarding(int synchronize, int \*error ); or int flxIdle(int synchronize, int \*error );

Description

This function is intended for use if the terminal used ip forwarding, sending ip traffic through the ecr or if the ecr want to pickup a card inserted event.

If e.g. a new PSAM is installed in the terminal, the ecr must forward ip traffic during PSAM installation after terminal reset. This function will repeatedly call the callback function pcbBreakIP.

Synchronize: If 1 the function will look for a synchronization frame from the terminal, and when found it will continue forwarding ip traffic until the callback function pcbBreakIP returns 1. If 0 the function will not look for the synchronization frame, but continuously forward ip traffic. If 2 the function will return CARD\_SWIPED, when card swiped or inserted.

If 3 the function will call callback FLX\_CALL\_CARD\_SWIPE, when card swiped or inserted, and run until pcbBreakIP return 1.

flxSetEcrExtendedFunction(0x100), must be called once before start of flxIdleIPforwarding(3).

\*error: Pointer to error returned (ERR\_IP\_ROUTING).

```
enum ERR_E
{
    ERR_UNKNOWN = 0x00010000,
    ERR_TERM_NOT_READY,
    ERR_NO_CONNECTION,
    ERR_NO_RECEIPT, (Not relevant from terminal SW version 3.4)
    ERR_SW_NOTCOMPATIBLE,
    ERR_NO_LICENCE,
    ERR_TOKEN_TRANS,
    ERR_SYSTEM_ERROR,
    ERR_KEY_TRANS,
    ERR_STATE,
    ERR_TOKEN,
    ERR_IP_ROUTING,
    ERR_TRANSMISSION,
    ERR_STOP_TIMEOUT,
    ERR_TLV_NOT_ALLOWED,
}
```

Return value: The function returns SUCCESS or FAILURE when pbcBreakIP not defined, or CARD\_SWIPED.

## 1.5.6 Token

### 1.5.6.1 flxReadToken() function

#### Synopsis

```
int flxReadToken (int what, char *tokenfile, uint *error);
```

#### Description

This function reads the unencrypted part of a transaction token.

what:           0: token data \*\*  
                  Data: <type >,<ver >,<aid>,<amount>,<curr>,<exp>,  
                      <VPKca>,<VK>,<ALGH>, <Merchant>,  
                  1: additional token data  
                      Data <feepercent>,<cvm>,<OriginalRefNo>,  
                  2: dcc token data  
                      <dccamount>,<dccamountfee>,<dccamountextra>,<dcccurre>,<exp>,  
                      <dccrate>,<clerkid>,  
tokenfile:       Tokenfilename  
error:           error code non zero (ERR\_TOKEN) if token not found.  
Return value:    string  
                  Example:  
                  token data: 'EMV,1,A00000000031010,457,USD,2,02,01,01,0107601100,'

\*\* see OTRS for more information.

## 1.5.7 Callback

### 1.5.7.1 verSigConfirmation callback routine

#### Synopsis

```
int verSigConfirmation( void );
```

#### Description

verSigConfirmation is used to approve/reject signature transactions. As an example, it could be implemented so the operator e.g. presses 'F1' for approval and 'F2' for rejected. The PSAM decides if this functionality is used or not. If it is not called and the transaction is approved, both receipts will be sent from the terminal straight away. If a fake signature is given, the operator must run a refund.

Return values:

SIGNATURE_OK	0x01
SIGNATURE_NOTOK	0x00
SIGNATURE_NOTKNOWNYET	0x02 // return 2 if the operator hasn't pressed ok/notok

### 1.5.7.2 setCardData callback routine

#### Synopsis

```
int setCarddata( const char *cPan, int cardType );
```

#### Description

Mandatory function.

setCarddata is used by the Merchant Unit SW to reject certain types of credit cards. E.g. if a customer wishes to get some cash together with the purchased item, then the Merchant Unit should ensure that the swiped card isn't a credit card. Otherwise the store will give the customer credit.

The cPan parameter delivers the first 8 digits of the track2 number (PCI padded) which is encoded in the ASCII format.

If a fee.ini file is present in the terminal, the cPan also delivers the CRC number of the card. The CRC number (e.g. 1 for Dankort) is separated from the 8 digit pan by a comma. The 'cardType' parameter is used to inform which card is swiped/inserted. Values are 0x00 for Nets related cards and 0x01 for other cards (e.g. Local cards). Also if configured the ereceipt token and hash values are delivered in cPan as an commaseparated ascii string, the format of the string is:

```
'<masked-pan>,<crc>,<storebox.dk-token>,<ekvittering.dk-token>,  
<nets-storebox.dk-hash>,<nets-ekvittering.dk-hash>,<nets-spare-hash>'
```

e.g: masked pan, crc and two Verifone Denmark ereceipt tokens and no nets psam hash values

```
'457199AAAAAA3429,1,  
bcb64439b3b8abe697933db30a43c37cada444212637ff18bf0166a7ee0648ce,  
2f40b9a58c9dd80d356f4b938dbea21d27fdc9c5f476d28f0c4e9f296652e59b,,,'
```

Return values:

CARDNUMBER_NOTOK	0x00
CARDNUMBER_OK	0x01
CARDNUMBER_OK_AMOUNTOTHER	0x02
CARDNUMBER_OK_CONFIRM	0x03 // External card confirm...
CARDNUMBER_OK_ECR_ACTION0	0x04
...	
CARDNUMBER_OK_ECR_ACTION16	0x14
...	
CARDNUMBER_OK_SILENTABORT	0x80

### 1.5.7.3 printReceipt callback routine

Synopsis

```
int printReceipt( unsigned int receipt_status, const char* cText );
```

Description

printReceipt prints a formatted receipt or report on the Merchant Unit Printer. Text is coded as ISO 8859-15 and is 0 terminated. The cText parameter has a maximum length of 16385 characters incl. '\0'. If a larger text must be printed it is divided in smaller segments.

Normal the length of cText is below 2000, but if receipt width is set via SetConfiguration more characters will be used.



The printReceipt must return the result of the receipt printing. If 0x00 is returned signaling an error condition the terminal will enter a no print state, which must be unlocked with the administrative function unlock receipt, before further transactions can be performed.

Since terminal sw version 3.4 the “no print state” has been discontinued. The receipt\_status parameter can be used to control when to cut the receipt. It is 0x01 if there are missing segments (e.g. during ADMIN\_REPORT\_LOG) and 0x0 if it is the last/whole segment (all transaction receipts). If it's 0x02 the receipt needs (signature) verification – must be printed (the dll may also call the menu callback\_function). You can use bit 1 (0x00/0x01) as cutting information. This function must be implemented.

Return value

Return RECEIPT\_OK = 0x01 for receipt received and printed OK, and RECEIPT\_NOTOK = 0x00 for error condition.

#### **1.5.7.4 printStatus callback routine**

Synopsis

```
void printStatus( const char cStat[21], unsigned int line );
```

Description

printStatus is used to show the terminals statuses on the ECR Display. The Flexdriver automatically calls printStatus when the status changes.

cStat is coded as ISO 8859-15 and is 0 terminated with a maximum length of 21 incl. '\0'.

The 'line' parameter is the terminal suggestion to which status line the text should be placed. Four lines are supported in the terminal. Values are 0x01 for line1, 0x02 for line2 etc. This function is mandatory to implement.

Return value

None

#### **1.5.7.5 abortTransaction callback routine**

Synopsis

```
int ecrAbortTransaction( void );
```

Description

abortTransaction can be used by the Merchant Unit software to abort Transactions. During transactions the Flexdriver repeatedly calls the abortTransaction function. Notice that even though the Merchant Unit SW tries to abort a Transaction, it might not be aborted. It is the terminal, which decides if the Transaction is aborted, or not. This function is mandatory to implement.

Return values

Return OPERATORWISHTOABORT = 0x01 if the operator wish to abort and

OPERATORDONOTWISHTOABORT = 0x02 if the operator do not wish to abort.

#### **1.5.7.6 adviceFlag callback routine**

Synopsis

```
void adviceFlag( void );
```

#### Description

This callback routine is called if the host indicates that a balancing/endofday routine must be called.

This means that the ECR must not be able to run further transactions before the endofday routine is run and approved. This function is mandatory to implement. (See new supplemental callback function 1.5.7.20 HostAdvice callback routine.

Return value

None

### 1.5.7.7 setAmount callback routine

#### Synopsis

```
FLX\_AMOUNT setAmount( void );
```

#### Description

This callback routine fetches the amount from the ECR. If this function isn't initialized in the flx-InitCallback function, then the amount from the flxCARDTransaction is used.

Return value

The transaction amount. The returned value must be the smallest available unit for given currency (øre for DKK).

### 1.5.7.8 getAmountFee callback routine

#### Synopsis

```
void getAmountFee( FLX\_AMOUNT fee );
```

#### Description

If the ECR isn't able to calculate a fee, the terminal can make the calculations and send it to the ECR. To do this, the ECR must initialize and implement this function (initialized with FLX\_CALLBACK\_GET\_AMOUNT\_FEE). The callback function is called when the fee amount is calculated by the terminal. The fee parameter is in øre.

Return value

None

### 1.5.7.9 setAmountGratuity callback function

#### Synopsis

```
FLX\_AMOUNT setAmountGratuity( void );
```

#### Description

This callback routine fetches the gratuity amount from the ECR (only active gratuity model 4 for transaction type capture).

Return value

The gratuity amount. The returned amount must be the smallest available unit for given currency (øre for DKK).

### 1.5.7.10 Menu callback function

#### Synopsis

```
void Menu( unsigned char command, int timeout,  
           unsigned char lines, const char menus[][ 21 ] );
```

#### Description

This function is currently used to display information to the operator, e.g. if or if not an EMV card is placed correctly in the terminal and for DCC selection.

The selected answer is given in the setMenuResult callback function.

command	<del>1: Application list select menu. No longer used, selection is on terminal.</del> 2: EMV card placed correct selection (menu line 0 e.g. 'KORT ISAT KORREKT') 3: <del>DCC currency select (menu line 0 is &lt;dccAmount&gt;,&lt;dccCurrency&gt;,&lt;exp&gt;).</del> <del>No longer used, now selection is on terminal.</del>
timeout	timeout for menu in milliseconds ( 0 if no timeout )
lines	number of lines in the menu
menus	menu texts

Return value

None

### 1.5.7.11 SetMenuResult callback function

#### Synopsis

```
int setMenuResult( char* rc );
```

#### Description

When a menu is displayed on the ECR screen, the setMenuResult is called repetitively (like abort), to verify the choice made by the operator.

#### Return values

setMenuResult must return 0 if the answer isn't known yet (the operator hasn't pressed a key), and no timeout has occurred, otherwise 1 is returned. When returning 1, in case of application select menu, the chosen menu point must be entered in the rc variable e.g. \*rc = 1 if the first menu line is selected. In case of EMV card is placed correct, \*rc=1 for 'yes' and \*rc=-1 or 0 for 'no'. If a timeout has occurred, \*rc = -2. For DCC selection \*rc=1 is 'yes'.

### 1.5.7.12 AdviceLog callback function

#### Synopsis

```
void pcbAdviceLog( unsigned int len, const unsigned char* data );
```

#### Description

The terminal is capable of sending copies of data store operations to the ECR. This is done by initializing and implementing this function.

len: length of host message

data: host data

Return value

None

### 1.5.7.13 GetReceipt callback function

#### Synopsis

```
void pcbGetReceipt (ReceiptStruct *rs);
```

#### Description

The terminal is capable of sending a binary receipt (the receipt broken down into its items). This function returns a C struct holding the receipt items.

#### Return value

None

### 1.5.7.14 EarlyStanPan callback function

#### Synopsis

```
void pcbEarlyStanPan (STAN* stan, PAN* pan, char *DTHR);
```

#### Description

The terminal is capable of sending the PCI padded PAN and the transaction STAN.

#### Return value

None

### 1.5.7.15 Timer event callback function

#### Synopsis

```
void pcbTimer (unsigned int remain);
```

#### Description

If this callback is activated the DLL will break blocking timers, and informing of the timer remaining in milliseconds.

#### Return value

None

### 1.5.7.16 GetToken callback function

#### Synopsis

```
int pcbGetToken (int *TokenLength, unsigned char *TokenData);
```

#### Description

This callback must be activated if an original authorization is performed. The callback will deliver the transaction token in TokenData and length in TokenLength, which is used in capture and reversal transactions using the putToken callback method. Max token length is 1024 bytes (see OTRS).

Return value      RECEIPT\_OK if token received and saved OK  
                    RECEIPT\_NOTOK if errors during token receive

### 1.5.7.17 PutToken callback function

#### Synopsis

```
void pcbPutToken (int TokenLength, const unsigned char *TokenData);
```

#### Description

This callback must be activated if capture, supplemental, reversal, post purchase or post refund is performed. The callback must deliver the transaction token from the original authorization in TokenData and length in TokenLength.

Return value

None

### 1.5.7.18 CheckStopList callback function

#### Synopsis

```
int pcbStopList (char *Code);
```

#### Description

This callback must be activated if offline transactions are performed (merchant initiative = 0x60). The function is called repeatedly until it returns 1 indicating that the code is ready. The callback must deliver the 6 character authorization code issued by Nets and a 2 character stop list status separated by ',' e.g. '123456,80', authorization code = '123456', status = '80' (in hex).

A timer will require this callback function to return 1 within 600 seconds. Best practice is to require the transaction authorization code before the transaction is started, e.g. for 'Dankort' transactions this is usually done by calling Nets with merchant id, card number and other required information, and receiving the authorization code. If a business decision is done, that no code is required, the ECR may return 6 spaces (0x20).

The returned code is printed on the receipt as 'AUT KODE:'

Return value	0 – code not ready yet
	1 – code ready

Status codes from OTRS 2.5:

Return value	0 – code not ready yet
	1 – code ready
'00'	Card not found in Stop List
'01'	Card found in Stop List
'02'	Card found in Stop List (pickup requested
'03'	Stop List not found
'04'...'7F'	RFU
'80'	Voice Authorization rejected
'81'...'FF'	RFU

### 1.5.7.19 BreakIP callback function

#### Synopsis

```
int pcbBreakIP (int sync);
```

#### Description

This callback must be activated if the terminal uses IP forwarding and the ECR handles e.g. installation of a new PSAM. It is repeatedly called by the flxIdleIPforwarding function.

sync: Is 1 if the terminal sync frame was detected (only valid when flxIdleIPforwarding is called with synchronize set to 1), and 0 if the frame is not detected.

The terminal always sends a sync frame (0x55,0xa5,0x5a,0xaa) during boot if it uses IP forwarding.

Return value      0 – continue flxIdleIPforwarding  
                    1 – break flxIdleIPforwarding

### 1.5.7.20 HostAdvice callback routine

#### Synopsis

```
void hostAdvice( unsigned char tag, char *text );
```

#### Description

This callback routine is called if the host issues an advice and is supplemental to the callback routine adviceFlag. The callback passes the host tag and its supplemental text string (max 20 char) that can be empty. E.g. tag 0xCA text 'CAtestCA'. At present host tags 0xC9 and 0xCA are passed. Tag 0xff is an internal tag from the terminal indicating that service is required. See the OTRS for more information.

Return value

None

### 1.5.7.21 preResult callback function

#### Synopsis

```
int preresult( unsigned char result );
```

#### Description

This callback is used with the EcrExtendedFunctions and signals that the ECR application is alive. Consult Verifone Denmark for more information.

Result:	0x00 (0)	Success
	0x01 (1)	Success – signature/refund transaction – operator accepted signature
	0x80 (128)	Rejected by Nets/PSAM
	0x81 (129)	Rejected – signature/refund transaction – operator rejected signature
	0x82 (130)	Rejected – Other reasons

Return value      Must return 0xA1 (161 decimal) at all times to inform the terminal the user application is still alive. No matter the value of Code.

### 1.5.7.22 getExtendedAmount callback function

```
typedef enum FLX_AMOUNT_T
```

```

{
    FLX_AMOUNT_AMOUNT,
    FLX_AMOUNT_FEE,
    FLX_AMOUNT_GRATUITY,
    FLX_AMOUNT_VAT,
    FLX_AMOUNT_BACK,
} FLX_AMOUNT_TYPE;

```

### Synopsis

```

void getExtendedAmount( FLX_AMOUNT_TYPE type, FLX_AMOUNT amount,
                        currCode curr)

```

### Description

This callback is currently used to send user entered amounts (e.g. gratuity) to the ECR.

Return value

None

### 1.5.7.23 setExtendedAmount callback function

### Synopsis

```

FLX_AMOUNT setExtendedAmount( FLX_AMOUNT_TYPE type, currCode *curr );

```

### Description

This callback is used to send amounts from the ECR to the terminal (e.g. VAT).

Return value

None

### 1.5.7.24 EIEdataReceived callback function

### Synopsis

```

int EIEdataReceived (int *EIE_data_length, unsigned char *EIE_DATA_BLOCK);

```

### Description

This callback must be activated if extended issuer envelope (EIE) is going to be used. The callback will deliver EIE\_data\_length bytes of struct EIE\_DATA\_BLOCK containing data send by Issuer host to the terminal as part of the transaction flow.

EIE\_data\_length is calculated in the following way:

```

1           for resp_mode
+4          for length_ie
+4          for length_eie
+length_ie
+length_eie

```

Fill in the struct by copy EIE\_data\_length bytes of EIE\_DATA\_BLOCK

```

struct
{
byte resp_mode;
// 0x00 - Empty not used - allow memset 0x00 to be used
// 0x01 - Request - Append EIE to PSAM (PSAM start with empty EIE)
// 0x02 - Advice - Clear EIE in PSAM before update with this
// 0x03 - Advice - Append to EIE hold by PSAM

```

```

// 0x04 - Advice - Clear EIE in PSAM only
// 0x05 - Request - Clear EIE in PSAM only
// 0x06 - Response from Host to the Request EIE
int length_ie; // Length of SW IE part of data
int length_eie; // Length of EIE part of data
byte data[512]; // 0..length_ie+length_eie
PACKED EIE_DATA_BLOCK; // Extended Issuer Envelope Data Block

```

Observe: length\_ie and length\_eie is stored as little-endian.

TAGlength in data as big-endian.

Return value

None

### 1.5.7.25 EIEdata2host callback function

#### Synopsis

```
int EIEdata2host (int *EIE_data_length, unsigned char *EIE_DATA_BLOCK);
```

#### Description

This callback must be activated if extended issuer envelope (EIE) is going to be used. The callback will request the EIE\_DATA\_BLOCK and length in EIE\_data\_length, of data to be send to Issuer host as part of the transaction flow.

EIE\_data\_length calculation and EIE\_DATA\_BLOCK format

- see EIEdataReceived callback above.

Return value:

- 0 no data
- 1 contains valid data

### 1.5.7.26 CardSwipe callback function

#### Synopsis

```
int pcbCardSwipe (int status);
```

#### Description

This callback must be activated if flxIdleIPforwarding(3, is used).

The callback is called as a card is swiped/inserted with the status set to 0x20

The ECR has to fetch this status and stop the flxIdleIPforwarding on the next BreakIP callback, before starting transaction with flxCardTransaction.

If transaction success and the ECR want to detect the card is removed from the reader, it's advised to call the admin flxAdministration(ADMIN\_CHECK\_CARD) repeatedly until the customer has removed the card. If the transaction fails this admin may hang as the card already removed.

The FLX\_CALL\_CARD\_SWIPE found in the DLL pdf should be changed to FLX\_CALLBACK\_CARD\_SWIPE.



## 1.5.8 Extra App Protocols

To support the extra applications in the terminal Verifone Denmark's Flexdriver (flxdrv.dll) now have a new callback and a function to reply to the extra application in the terminal.

### 1.5.8.1 flxExtraReply

This function is used to send data to the extra application in the terminal.

The ECR must ensure connection to the terminal is running, before sending data to the extra application in the terminal.

This is done by running flxIdleIPforwarding or flxCardTransaction.

```
int flxExtraReply(struct ExtraAppReplyStruct T *ExtraAppData, int extra length)
```

or the Visual Basic interface.

```
int WINAPI flxExtraReplyVB(struct ExtraAppReplyStruct_T *ExtraAppData, int  
extra_length)
```

Returns

0: OK

1: Communication to terminal timed out (after 20 sec) and a callback is made to tell the extra app, this status.

## 1.5.9 Callback pcbExtraAppdataReceived

The callback is called as a result of the extra application performing a send data to the ECR. For this to work the ECR must be listening for the callback, if the function flxIdleIPforwarding(2, ref error); is called. We wait for a card swipe and if some extra application data is received it passed on in the pcbExtraAppdataReceived callback.

If a card has been swiped, the flxIdleIPforwarding must be stopped, and the flxCardTransaction called to start a transaction.

This is done by setting the status to abort in the FLX\_CALLBACK\_ABORT.

In the switch between flxIdleIPforwarding and flxCardTransaction the extra app will receive a response saying ECR not listening for data from the terminal, this should prevent packets being lost.

```
static int (*pcbExtraAppdataReceived)(size_t ExtraApplength,  
const unsigned char *EXTRA_APP_DATA_BLOCK_block, char AppNo) = NULL;
```

or the Visual Basic interface.

```
static void (WINAPI *pcbExtraAppdataReceivedVB)(int Extralength,  
unsigned char *EXTRA_APP_DATA_BLOCK_block, char AppNo, int *res) = NULL;
```

### 1.5.10 Enabling the pcbExtraAppdataReceived callback

To register the callback in the flxdrv.dll you need to do an

```
flxInitCallback(FLX_CALLBACK.FLX_CALLBACK_EXTRA_APP_DATA_RECEIVED,  
pcbExtraAppdataReceivedDelegate);
```

and set the EcrExtendedFunctions bit 0x0100 this may be done like this:  
First obtaining EcrExtendedFunctions

```
flxGetSetEcrExtendedFunctions(ADMIN_GET_ECR_EXTENDED_FUNCTIONS,  
EcrExtendedFunctions);
```

Second set the bit

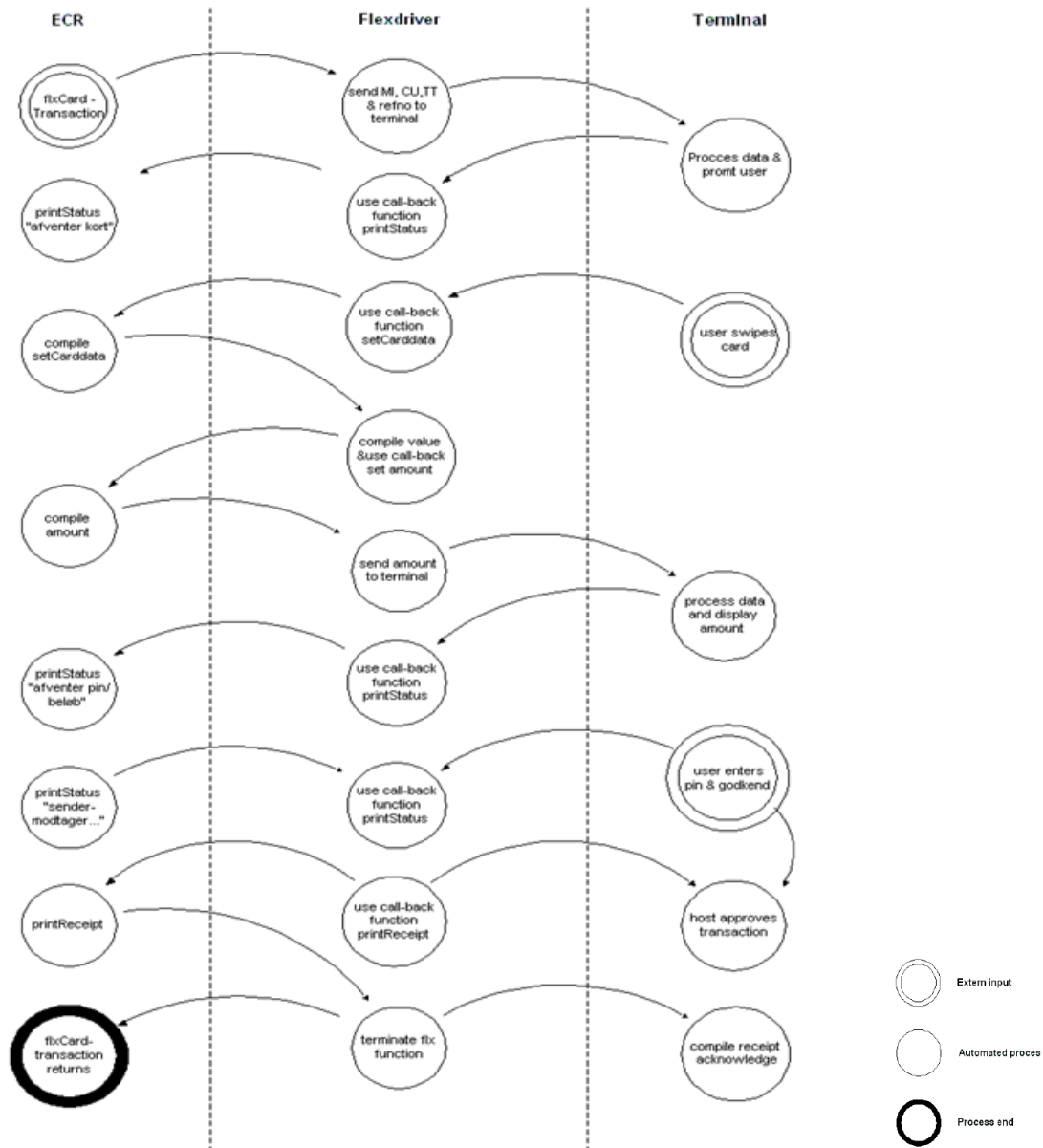
```
EcrExtendedFunctions = EcrExtendedFunctions | 0x0100;
```

Third set the EcrExtendedFunctions in terminal to the new value

```
flxGetSetEcrExtendedFunctions(ADMIN_SET_ECR_EXTENDED_FUNCTIONS,  
EcrExtendedFunctions);
```

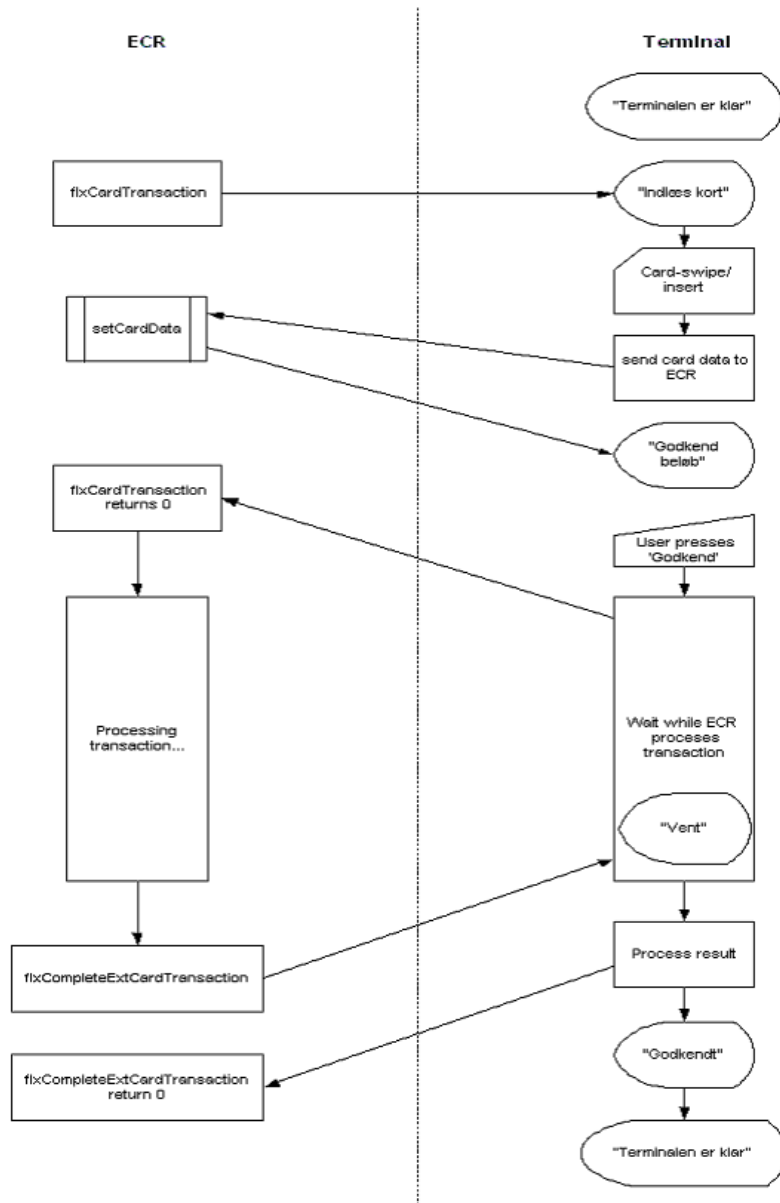
### 1.5.11 Flexdriver Flow

The Flexdriver is single threaded, which means that none of the callback functions may be blocking. The Flexdriver uses 90 percent of its runtime in the 'readEventsFromTerminal' loop. This means that the faster callback routine returns, the sooner the Flexdriver can return to this loop and thereby avoid retransmission on the link layer. The figure below illustrates a simplified Flexdriver transaction flow.



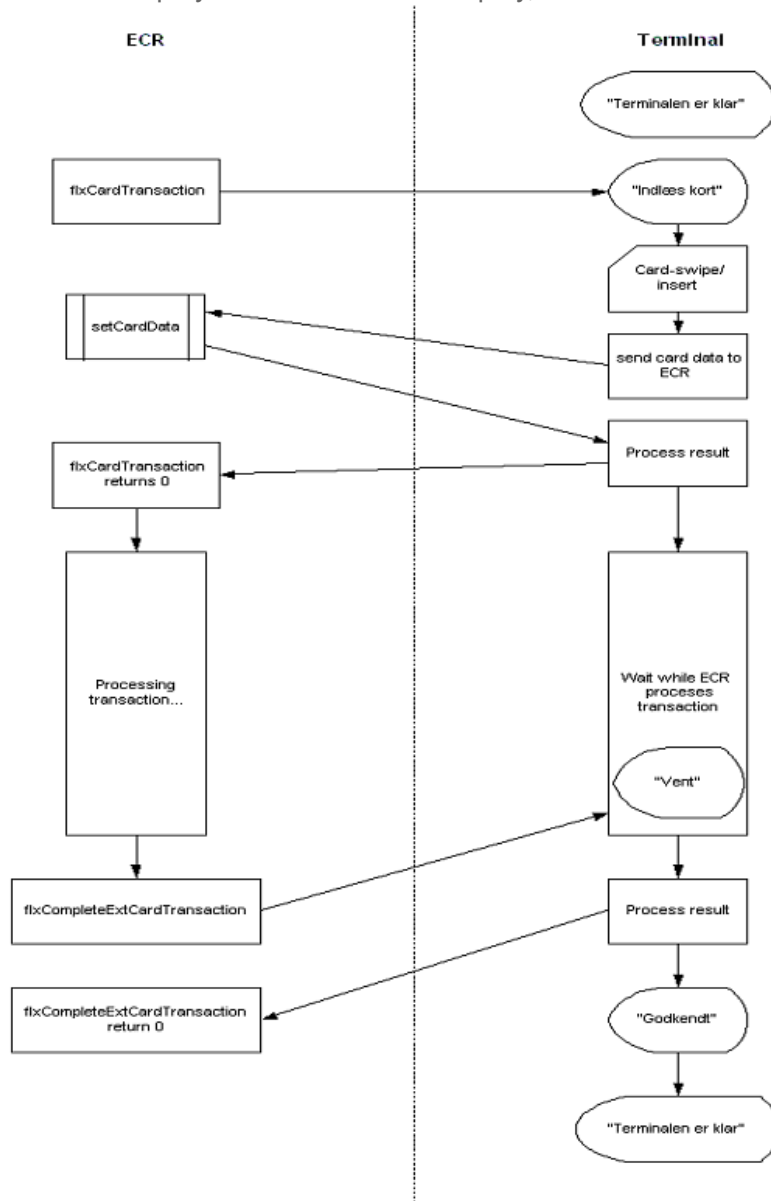
### 1.5.12 Local cards with Flexdriver – amount displayed

The Local Card functionality is comprised of two flx functions. The purchase is initiated in the same way as with normal transactions, but flxCARDTransaction returns after delivering the card data. The terminal is now listening for input while the ECR processes the transaction. Finally the ECR calls flxCompleteExtCardTransaction with the transaction result as a parameter.



### 1.5.13 Local cards with Flexdriver – amount not displayed

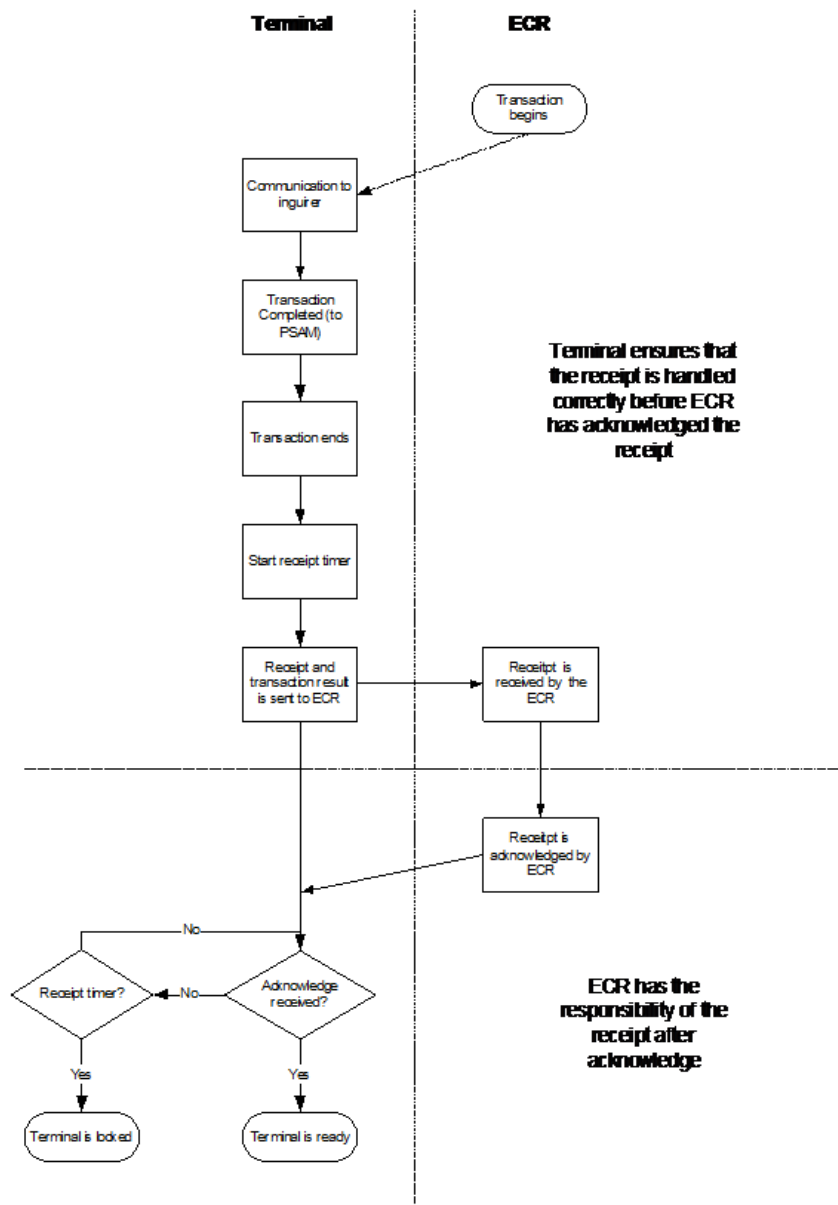
If the amount shouldn't be displayed in the terminal display, the flow is a little different.



## 1.6 General Flowcharts

### 1.6.1 The important flow of the receipt and the transaction result

As seen in the illustration below, the receipt is the responsibility of the ECR after acknowledging the receipt.



## 1.A Extra receipt information

Extra receipt information (binary receipt). The callback function `getReceipt(ReceiptStruct *)` returns data extracted from the receipt generated by the terminal. The terminal must be configured to send the information to the DLL. The struct is reset to all zeros by default, and only valid receipt information is mapped into the struct.

Two examples are given in section 1.A.2 and 1.A.3: a completed MobilePay transaction and a failed Swipp transaction.

Consult the file **flex.h** for the latest version of `ReceiptStruct_T`.

```
// see: NETS Electronic Receipt design document (JKP, august 2013)
// Defined Scheme Ids
// Value ERCo
// 01 Kvittering.dk
// 02 eKvittering Aps
// 03 1F
// 20 Nets Denmark A/S
#define KVITTERINGDK_SCHEME 0x01
#define EKVITTERINGDK_SCHEME 0x02
#define NETSDK_SCHEME 0x20

#define NETS_HASH_MAX 3
#define EHASH_LENGTH_MAX 32

// Placeholder for NETS ereceipt HASH values etc.
// See. OTRS Retrieve Hash Value command ('0019')
typedef struct NetsHashStruct_T
{
    unsigned char scheme_id;
    unsigned char algorithm_id;
    unsigned char salt_version;
    unsigned char len_hash;
    unsigned char hash[EHASH_LENGTH_MAX];
} PACKED NetsHash, *pNetsHash;

typedef struct NetsHashResStruct_T
{
    NetsHash NetsEreceipts[NETS_HASH_MAX]; // NETS E-receipt NETS_HASH_MAX = 3
    short hashRes;
} PACKED NetsHashRes;

typedef struct ReceiptStruct_T
{
    time_t      gmttime;           // Windows (Unix) time stamp (gmt) (NOTE: 64 bit)
    FLX_AMOUNT  total;             // total amount in smallest currency unit
    FLX_AMOUNT  extra;             // extra -
    FLX_AMOUNT  fee;               // fee -
    FLX_AMOUNT  gratuity           // gratuity -
    int         currency;          // Currency code (208=DKK, 978=EUR etc.)
    int         Stan;              // Nets ref number converted from BCD code
```

```

int      PSAM_Creator;          // PSAM #
int      PSAM_ID;              // PSAM ID #
int      MTI;                  // Nets apacs transaction code
int      Asw1Asw2;             // PSAM retur code (see OTRS spec.)
int      ActionCode;           // Transaction result code
int      refNr;                // Transaction ref. nr passed from user
char     netsResult;           // Nets Transaction result
char     CvmStatus;            // Cardholder verification method see EMV specs
char     CardDataSource;       // Magnetic- or chip card
char     TR;                   // Transaction request
char     nota;                 // True if signature on receipt
char     copy;                 // True if receipt copy
char     mt;                   // EMV decline code see EMV specification
char     psn;                  // EMV pan sequence number see EMV spec.
char     lenPAN;               // Length of PAN
char     lenAID;               // Length of AID
char     PAN[10];              // Primary account number BCD code
char     AID[16];              // EMV application ident.
char     ATC[2];               // EMV value see EMV specification
char     AED[3];               // EMV value see EMV specification
char     ARC[2];               // EMV value see EMV specification
char     PosEntryMode[3];      // Transaktionen information see OTRS/EMV.
int      number;               // Merchant Nets number
char     authorizationCode[6+1]; // Nets authorization code as string
char     CardName[16+1];       // Card name as string
char     TermIdent[8+1];       // Terminal ID string
char     name[18+1];           // Merchant name as string
char     city[16+1];           // Merchant city as string
char     address[24+1];        // Merchant address as string
char     zip_code[8+1];        // Merchant zip code as string
char     phone[24+1];          // Merchant phone number as string
char     cvr[12+1];            // Merchant CVR number as string
int      DCCcurrency;          // DCC Currency code
char     DCCrate[8+1];         // DCC rate in ascii (e.g. '0.138400')
int      CardCRC;
char     CancellationAllowed;
char     BatchNumber[12+1];    // Batch Number as string
FLX_AMOUNT vat;
char     Ereceipt[64+1];       // E-receipt
FLX_AMOUNT saldo;
FLX_AMOUNT dcctotal;
FLX_AMOUNT dccfee;
FLX_AMOUNT dccgratuity;
FLX_AMOUNT cashback;
char     expiredate[2];        // local card expire date as 2 byte
                                   // (first: year (bcd) 0-99, second: month (bcd) 0-11)

int      NetsEreceiptLen;
NetsHashRes NetsEreceipts;     // NETS E-receipt NETS_HASH_MAX = 3
char     Kreceipt[64+1];       // e-receipt Verifone Denmark storebox.point-ts.dk
char     vasJsonReceipt[256];
char     tcc[3];               // Transaction Condition Code
} PACKED ReceiptStruct, *pReceiptStruct;

```



### 1.A.1 CSV - Comma Separated String - getReceipt

The `_stdcall` interface of `getReceipt(BSTR *rc)` returns the information as a csv (semicolon separated string) including:

```
Time (yyyy-mm-dd hh:mm); // 0
PAN;
total;
extra;
fee;
gratuity; // 5
currency;
Stan;
PSAM_Creator;
PSAM_ID;
netsResult; // 10
Asw1Asw2;
_CvmStatus;
CardDataSource;
authorizationsCode;
CardName; // 15
TermIdent;
number;
name;
_city;
address; // 20
zip;
phone;
cvr;
refNr;
DccCur; // 25
DccRate;
CRC;
CancellationAllowed;
vat;
ereceipttoken; // 30
SALDO;
BACK;
DCCTOTAL;
DCCGEBYR;
DCCDRIKKEPENGE; // 35
CARDDATASOURCE;
AID;
ATC;
```

```
AED;  
ARC;           // 40  
EXPDATE;  
STOREBOXDK;  
EKVITTERING;  
NETSDANMARK;  
STOREBOX;     // 45  
vasJsonReceipt;  
TCC;          // Transaction Condition Code
```

## 1.A.2 Example: binary receipt of completed MobilePay transaction

BINARY RECEIPT:

```
2017-03-02 11:24;920861615641535085;35;0;0;0;208;12345;0;0;0000;0000;0;      ;
MobilePay;    ;0;      Verifone QA;      Herlev;      Knapholm 7; 2730;    ;
POSDKVF418;0;0;1      ;508;;0;0; 991187170302112423      ;0;0;35;0;0;0;;0;000000;;0000;;;
5e9e967ee5d7464993b2b9e831b081a5;
{"time":"11:24","date":"2017-03-02","currency":"DKK","amount":35,
"orderId":"991187170302112423","customerToken":"","
"customerReceiptToken":"5e9e967ee5d7464993b2b9e831b081a5",
"transactionId":"61258283","tr":0,"status":"DONE"
};VF1
```

1. TID	: 2017-03-02 11:24	
2. PAN	: 920861615641535085	
3. TOTAL	: 35	
4. EXTRA	: 0	
5. GEBYR	: 0	
6. DRIKKEPENGE	: 0	
7. VALUTAKODE	: 208	
8. STAN	: 12345	(locationId - "xxxxx") prefixed 0's are removed
9. PSAMCREATOR	: 0	
10. PSAMID	: 0	
11. STATUS	: 0000	
12. ASW1ASW2	: 0000	
13. CVMSTATUS	: 0	
14. AUTORISATIONSKODE	:	
15. KORTNAVN	: MobilePay	
16. TERMINALID	:	
17. AFTALENUMMER	: 0	
18. NAVN	: Verifone QA	(merchantName)
19. BY	: Herlev	(locationCity[5])
20. ADRESSE	: Knapholm 7	(locationStreet)
21. POSTNUMMER	: 2730	(locationCity)
22. TELEFON	:	
23. CVR	: POSDKVF418	(merchantId)
24. REF.NUMMER.	: 0	
25. DCC VALUTAKODE	: 0	
26. DCC KURS	: 1	
27. CRC	: 508	
28. BATCH	:	
29. ANNULLERINGAKTIV	: 0	
30. MOMS	: 0	
31. E-KVITTERING	: 991187170302112423	(orderId)
32. SALDO	: 0	
33. BACK	: 0	
34. DCCTOTAL	: 35	
35. DCCGEBYR	: 0	
36. DCCDRIKKEPENGE	: 0	

37. CARDDATASOURCE	: 0
38. AID	:
39. ATC	: 0
40. AED	: 000000
41. ARC	:
42. EXPDATE	: 0000
43. STOREBOXDK	:
44. EKVITTERING	:
45. NETSDANMARK	:
46. STOREBOX	: 5e9e967ee5d7464993b2b9e831b081a5
47. vasJsonReceipt	: {"time":"11:24", "date":"2017-03-02" "currency":"DKK", "amount":35, "orderId":"991187170302112423", "customerToken":""," "customerReceiptToken":"5e9e967ee5d7464993b2b9e831b081a5", "transactionId":"61258283", "tr":0,, "status":"DONE" } 
48. TCC	: VF1

For readability newlines are inserted in the vasJsonReceipt - real data are without newlines.

### 1.A.3 Example: binary receipt of failed Swipp prepaid scanned bar code purchase

As Swipp is no longer available as a feature, this example should serve as a guideline for implementing MobilePay.

BINARY RECEIPT:

```
2015-12-10 09:51;4561626605;3600;0;0;0;208;0;0;0;0000;120B;0; ; VAS TYPE 507; ;0;
; ; ; ; ; ;0;0;1 ;
507;;0;0;;0;0;0;0;0;0;0;000000;;0000;;;;;
{"date":"2015-12-10","time":"09:51","currency":"DKK","amount":"3600",
"merch_id":"Verifone","pos_id":"991027","key_id":"3","trans_id":"","
"ref_id":"8c4a53049f2311e58a3100081924a19b","status":"FAILED","fail_code":"404001"};VF1
```

1. TID	: 2015-12-10 09:51
2. PAN	: 4561626605
3. TOTAL	: 3600
4. EXTRA	: 0
5. GEBYR	: 0
6. DRIKKEPENGE	: 0
7. VALUTAKODE	: 208
8. STAN	: 0
9. PSAMCREATOR	: 0
10. PSAMID	: 0
11. STATUS	: 0000
12. ASW1ASW2	: 120B
13. CVMSTATUS	: 0
14. AUTORISATIONSKODE	:
15. KORTNAVN	: VAS TYPE 507
16. TERMINALID	:
17. AFTALENUMMER	: 0
18. NAVN	:
19. BY	:
20. ADRESSE	:
21. POSTNUMMER	:
22. TELEFON	:
23. CVR	:
24. REF.NUMMER.	: 0
25. DCC VALUTAKODE	: 0
26. DCC KURS	: 1
27. CRC	: 507
28. BATCH	:
29. ANNULLERINGAKTIV	: 0
30. MOMS	: 0
31. E-KVITTERING	:
32. SALDO	: 0
33. BACK	: 0
34. DCCTOTAL	: 0
35. DCCGEBYR	: 0
36. DCCDRIKKEPENGE	: 0
37. CARDDATASOURCE	: 0

38. AID	:
39. ATC	: 0
40. AED	: 000000
41. ARC	:
42. EXPDATE	: 0000
43. STOREBOXDK	:
44. EKVITTERING	:
45. NETSDANMARK	:
46. STOREBOX	:
47. vasJsonReceipt	: {"date":"2015-12-10", "time":"09:51", "currency":"DKK", "amount":"3600", "merch_id":"Verifone", "pos_id":"991027", "key_id":"3", "trans_id":"", "ref_id":"8c4a53049f2311e58a3100081924a19b", "status":"FAILED", "fail_code":"404001" }
48. TCC	: VF1

### Example Transaction Condition Code TCC:

Search for **Transaction Condition Code** in latest OTRS  
 We expanded it with a V for VAS in the **Card Entry Mode**

## 1.B Language Support

The text strings are initialized to Danish text strings by default. The Windows dll supports text strings from a Windows resource (\*.rc) file.

```
// Microsoft Visual C++ generated include file.
// Used by flxdll.rc

#define IDS_ERMSG_TERM_NOT_READY 100
#define IDS_ERMSG_CARD_INIT 101
#define IDS_ERMSG_DATA LINK 102
#define IDS_ERMSG_NO_RECEIPT 103
Not relevant from terminal SW version 3.4
#define IDS_ERMSG_SYSTEM_ERROR 104
#define IDS_ENAI_MSG_CONNECT 105
#define IDS_ENAI_MSG_DISCONNECT 106
#define IDS_ENAI_MSG_SEND 107
#define IDS_ENAI_MSG_RECEIVE 108
```

If the file flxText.txt file exist in the install directory its content will be used. Nine text strings are currently supported, the strings must be placed in correct order.

Example of format of the file flxText.txt with Danish text strings:

```
# If line starts with # it is skipped
#
# Format: index, max 20 characters
# ERMSG_TERM_NOT_READY 100
# ERMSG_CARD_INIT 101
# ERMSG_DATA LINK 102
# ERMSG_NO_RECEIPT 103
# ERMSG_NO_RECEIPT 103
(Not relevant from terminal SW version 3.4)
# ERMSG_SYSTEM_ERROR 104
# ENAI_MSG_CONNECT 105
# ENAI_MSG_DISCONNECT 106
# ENAI_MSG_SEND 107
#
100,TERMINAL OPTAGET
101,PORT FEJL
102,VENT LIDT, PRØV IGEN
103,INGEN KVITTERING (Not relevant from terminal SW version 3.4)
104,SYSTEM FEJL
105,OPKALD...
106,AFBRYDER...
107,SENDER...
108,MODTAGER...
```

## 1.C Terminal Properties

Terminal properties returned from flxTerminalProperties();  
separation is ',' (comma)

```
1:      <TERMINALID>
        max 8 char eg: '990197'
2:      <RECEIPTTYPE>
        'g'
3:      <GRATUITYMODEL>
        '0'
4:      <DCC>
        '0' DCC disabled
        '1' DCC enabled
5:      <TOKEN>
        '0' Tokenbased transaction disabled
        '1' Tokenbased transaction enabled
6:      <PREPAID>
        '0' Prepaid transaction disabled
        '1' Prepaid transaction enabled
7:      <KEYENTRY>
        '0' Keyentry disabled
        '1' Keyentry enabled
8:      <OFFLINE>
        '0' Offline transaction disabled
        '1' Offline transaction enabled
9:      <IP ROUTING>
        '0' IP routing disabled
        '1' IP routing enabled
```



```

10:      <MERCHANTNUMBER>
        '0' 10 char eg. '0760110001'
11:      <MERCHANTNAME>
        18 char eg: 'Nets PSAM-146 EØÅ'
12:      <MERCHANTCITY>
        16 char eg: 'BALLERUP
13:      <MERCHANTADDRESS>
        24 char eg: 'LAUTRUPBJERG 10
14:      <MERCHANTZIP>
        8 char eg: 'DK-2750          '
15:      <MERCHANTPHONE>
        24 char eg: ' (+45) 44 68 44 68          '
16:      <MERCHANTBRN>
        12 char eg: '12345678          '
17:      <FLXTRACELEVEL>
        number
        '16' no trace
18:      <LANGUAGE>
        3 char
        'DAN' Danish
        'SWE' Swedish
        'ENG'
        'NOR'
        'GER'
        'FRA'

```

19      <VAT>  
         number  
         eg. '250' vat is 25.0 %  
20      <USERINPUT>  
         '0' user input of dcc and gratuity disabled  
         '1' user input of dcc and gratuity enabled  
21      <COUNTRYCODE>  
         '208'          Denmark  
         '752'          Sweden  
         '578'          Norway  
         '826'          UK  
         '276'          Germany  
         '250'          France  
22      <DEFAULTCURRENCY>  
         '208'          DKK  
         '752'          Sweden  
         '578'          Norway  
         '826'          UK  
         '276'          Germany  
         '250'          France  
23      OLD CONTACTLESSFLAGS  
         '0' hardcoded

```

24    <TERMINALTYPE>
      '17'      - Cash
      '33'      - Quasi Cash
      '34'      - retail (hex '22')
      '35'      - UTP
      '36'      - Hotel/Restaurant
      '37'      - Fuel
25    <DCC_MARKUP>
      '3.0000'
26    <TCS>
      '1'      - 0, 1 or 2
27    <SOFTWARE_VERSION>
      '3.4.00' (example)      - terminal TAPA version
28    <CDP>      card data protection
      '12'      - 0, 2, 4, 12
29    <PSAMCDP>  - psam cdp support
      '4'      - 0, 4
30    <HARDWARENAME>
      'YOMANI'   - YOMANI, XENTA,...
31    <EIE>
      '0' Disabled
      '128' Enabled
32    <PSAM_EIE>
      '0' Disabled
      '128' Enabled
33    <ISSUER_ENVELOPE_NON_EMV_SIZE>
34    <ISSUER_ENVELOPE_EMV_SIZE>
35    <TOTAL_ISSUER_ENVELOPE_NON_EMV_SIZE>
36    <TOTAL_ISSUER_ENVELOPE_EMV_SIZE>
37    <ERECEIPT_SCHEMES>
      "00000000" schemes see OTRS "Retrieve Hash Values" command
      e.g. "80000003" scheme "01", "02" and "20"
38    <PSAM_VERSION>
      "80.07" reported PSAM version e.g. 80.07
39    <CONTACTLESSFLAGS>
      ,,

```

## 1.D DCC

Terminals configured to offer DCC will – if DCC is selected – calculate DCC amounts based on the exchange rate received for the selected DCCcurrency. The DCCcurrency is selected from the terminals card number table based on the customers card PAN prefix.

Rates are received and updated daily (can differ during weekends) and include a markup. If rates are outdated the terminal will not offer DCC.

If DCC is selected by the user (customer) DCCcurrency is different from merchant currency and DCCrate is the rate received (including markup).

The merchant receives payment from DCC transactions in his currency (merchant currency), and normally do not need the DCC information, but the terminal splits the 'received rate' into 'exchange rate' and 'exchange markup' to calculate DCC amounts, where 'received rate' = 'exchange rate' + 'exchange markup' and markup is a configuration constant (e.g. 3%) set in the terminal, and defined by the acquirer, the company supplying the rates and the merchant.

Transaction amount (DCCcurrency) = Transaction amount (merchant currency) \* 'received rate'.

Transaction fee (DCCcurrency) = Transaction fee (merchant currency) \* 'exchange rate'.

Transaction Gratuity (DCCcurrency) = Transaction Gratuity (merchant currency) \* 'exchange rate'.

Note: Merchant selection of DCC is deprecated.

## 1.E Drop IP listing

```
int flxDropIp(char *ip_addr, char *terminal_ident, int to_do)
{
    int mysocket;
    struct sockaddr_in dest;
    char wzRec[256];
    char searchResult[256];
    int nLeft = 256;
    int iPos = 0;
    int nData = 0;
    int res = 0;
#ifdef _WIN32
    uint NonBlock = 0;
#endif
    fd_set Reader ;
    int n ;
    struct timeval tv ;
    int iLoopCnt = 0;

    mysocket = socket(AF_INET, SOCK_STREAM, 0);

    memset(&dest, 0, sizeof(dest)); /* zero the struct */
    dest.sin_family = AF_INET;
    dest.sin_addr.s_addr = inet_addr(ip_addr);
        /* set destination IP number */
    dest.sin_port = htons(2001);
        /* set destination port number */

    if (connect(mysocket, (struct sockaddr *)&dest,
        sizeof(struct sockaddr)) == SOCKET_ERROR)
    {
        _trace2File("Error connect %s 2001 %d - %s",
            ip_addr, errno, strerror(errno)) ;
        return 0;
    }
    _trace2File("Connect %s 2001 %d - %s", ip_addr,
        errno, strerror(errno));

    /* Change the socket mode on the listening socket
        from blocking to non-block */
#ifdef _WIN32
    NonBlock = 1;
    if (ioctlsocket(mysocket, FIONBIO, &NonBlock) == SOCKET_ERROR)
    {
        _trace2File("ioctlsocket() failed - %d - %s", errno,
            strerror(errno));
        return 0;
    }
#else
    fcntl(mysocket, F_SETFL, fcntl(mysocket, F_GETFL, 0) | O_NONBLOCK);
#endif
    // process data
```

```

if (to_do == 0)
{
    sprintf(wzRec,"DropIp:%s\n", terminal_ident);
    sprintf(searchResult,"Dropping IP port 2000 on
        terminal %8.8s\n", terminal_ident);
}
else
{
    sprintf(wzRec,"Reboot:%s\n", terminal_ident);
    sprintf(searchResult,"Restarts terminal %8.8s
        - Wait 1-2 min.\n", terminal_ident);
}
nLeft = strlen(wzRec);
do
{
    nData = send( mysocket, &wzRec[iPos], nLeft, 0 );
    if( nData == SOCKET_ERROR ) {
        _trace2File("Error sending data %d - %s", errno,
            strerror(errno)) ;
        break;
    }
    nLeft -= nData;
    iPos += nData;
} while( nLeft > 0 );

// Set up the file descriptor set.
FD_ZERO(&Reader) ;
FD_SET(mysocket, &Reader) ;

// Set up the struct timeval for the timeout.
tv.tv_sec = 10 ;
tv.tv_usec = 0 ;

// Wait until timeout or data received.
_trace2File("Before select..");
n = select ( mysocket+1, &Reader, NULL, NULL, &tv ) ;
if ( n == 0)
{
    _trace2File("Timeout..");
    return 0;
}
else if( n == -1 )
{
    _trace2File("Error..");
    return 0;
}
_trace2File("After select..n=%d FD_ISSET(mysocket, &Reader)=%d",
    n, FD_ISSET(mysocket, &Reader));
memset( &wzRec, 0, sizeof( wzRec ) );
nLeft = strlen(searchResult);
iPos = 0;
if (FD_ISSET(mysocket, &Reader))
{

```

```

do
{
    nData = recv( mysocket, &wzRec[iPos], nLeft, 0 );
    if( nData == SOCKET_ERROR ) {
        _trace2File("Error receiving data %d - %s",
                    errno, strerror(errno)) ;

        break;
    }
    nLeft -= nData;
    iPos += nData;
} while((nLeft > 0) && (iLoopCnt++ < 10));
}
_trace2File("Data Received len=%d", iPos);
if (iPos > 0)
{
    if (strncmp(wzRec, searchResult, strlen(searchResult)) == 0)
    {
        _trace2File("Received expected result from terminal");
        if (pKiss)
        {
            pSleep(CONNECT_SLEEP);
            PKISS_Delete(&pKiss);
        }
        res = 1;
    }
}
shutdown(mysocket, 2); // SD_BOTH
closesocket(mysocket);
return res;
}

```

## 1.F Extra App Protocols

To support the extra applications in the terminal flexdriver (flxdrv.dll) now have a new callback and a function to reply to the extra application in the terminal.

### 1.F.1 flxExtraReply

This function is used to send data to the extra application in the terminal, our DLL kasse demo (no longer supported) show examples of use of the command.

The ECR must ensure connection to the terminal is running, before sending data to the extra application in the terminal.

This is done by running flxIdleIPforwarding or flxCardTransaction.

int flxExtraReply(struct ExtraAppReplyStruct\_T \*ExtraAppData, int extra\_length)  
or the Visual Basic interface

int WINAPI flxExtraReplyVB(struct ExtraAppReplyStruct\_T \*ExtraAppData,  
int extra\_length)

Returns 0 – All OK

-1 : Communication to terminal timed out (after 20 sec) and a callback is made to tell the extra app, this status.

### 1.F.2 Callback pcbExtraAppdataReceived

The callback is called as a result of the extra application performing a send data to the ECR. For this to work the ECR must be listening for the callback, if the function flxIdleIPforwarding(2, ref error); is called. We wait for a card swipe and if some extra application data is received it passed on in the pcbExtraAppdataReceived callback.

If a card has been swiped, the flxIdleIPforwarding must be stopped, and the flxCardTransaction called to start a transaction.

This is done by setting the status to abort in the FLX\_CALLBACK\_ABORT.

In the switch between flxIdleIPforwarding and flxCardTransaction the extra app will receive a response saying ECR not listening for data from the terminal, this should prevent packets being lost.

static int (\*pcbExtraAppdataReceived)(size\_t ExtraApplength,  
const unsigned char \*EXTRA\_APP\_DATA\_BLOCK\_block, char AppNo) = NULL;

or the Visual Basic interface

static void (WINAPI \*pcbExtraAppdataReceivedVB)(int Extralength,  
unsigned char \*EXTRA\_APP\_DATA\_BLOCK\_block, char AppNo, int \*res) = NULL;

### 1.F.3 Enabling the pcbExtraAppdataReceived callback

To register the callback in the flxdrv.dll you need to do an

flxInitCallback(FLX\_CALLBACK.FLX\_CALLBACK\_EXTRA\_APP\_DATA\_RECEIVED,  
pcbExtraAppdataReceivedDelegate);

and set the EcrExtendedFunctions bit 0x0100 this may be done like this:

First obtaining EcrExtendedFunctions



```
flxGetSetEcrExtendedFunctions(ADMIN_GET_ECR_EXTENDED_FUNCTIONS,  
EcrExtendedFunctions);  
Second set the bit  
EcrExtendedFunctions = EcrExtendedFunctions | 0x0100;  
Third set the EcrExtendedFunctions in terminal to the new value  
flxGetSetEcrExtendedFunctions(ADMIN_SET_ECR_EXTENDED_FUNCTIONS, EcrExtendedFunctions);
```

## 1.G Custom Images

### 1.G.1 Yomani

It is possible for customers with a Yomani terminal to show a custom image on the idle screen. Additionally, if the customer is using an ECR integration, the customer can get an image, i.e. their logo, shown in the top right corner during transactions. An example of this is shown in Figure 1.1.



Figure 1.1: Example images

Name	Size (pixels)	Format
Idle image	320 × 211	png
Transaction image	140 × 35	png

Table 1.4: Yomani image sizes

## 1.G.2 Verifone

### 1.G.2.1 Vx 820

Name	Size (pixels)	Format
Idle image	240 × 156	png
Transaction image	69 × 36	png

**Table 1.5:** Vx 820 image sizes

### 1.G.2.2 Vx 680

Name	Size (pixels)	Format
Idle image	240 × 124	png
Transaction image	69 × 36	png

**Table 1.6:** Vx 680 image sizes

### 1.G.2.3 Vx 520c

Name	Size (pixels)	Format
Idle image	320 × 124	png
Transaction image	69 × 28	png

**Table 1.7:** Vx 520c image sizes

The images **must** be in png format and **must** follow the size specification shown in Table 1.4, 1.5, 1.6 or 1.7, all other images will be rejected!

In case the pictures does not fit exactly into the required size, it is possible to use transparent pixels to fill out the remaining space.

To use this functionality, send the pictures and terminal number(s) to [kundeservice@verifone.com](mailto:kundeservice@verifone.com).

## 1.H How to interpret a Network Report

On terminals with software version 2.2.01 or newer a Network Report can be made. The report can be found in test menu – 7 – 6 or the title 'NETVÆRK'.

The network type is written in the terminal, but if the type is Ethernet a report is also written on the printer.

### Network Report

2013-01-22

14:52

### Explanation

TERM:	00990067	The terminals id
DOMAIN:		If the terminal belongs to a domain, part of point.local network, the name is written here
DNS1:	192.168.0.200	Domain Name Server 1
DNS2:	192.168.0.204	Domain Name Server 2
IP:	192.168.0.56	The IP address of the terminal
SUBNET:	255.255.255.0	The terminal under a subnet
GATEWAY:	192.168.0.1	If IP address can't be found then ask here
DHCP:	192.168.0.204	Were shall the terminal ask to get its address

### PING TEST

DNS1	
Ping 192.168.0.200 – SUCCESS	A DNS shall not reply on ping, but often does
DNS2	
Ping 192.168.0.204 – SUCCESS	
GATEWAY	
Ping 192.168.0.1 – SUCCESS	A DNS shall not reply on ping, but often does

### DNS lookup TEST

Lookup on Domain Name Server to check if it recognizes the IP addresses we need to make:  
Transactions,  
Send logfiles,  
Download parameter,  
Download TLCMDB,  
Download programs etc.

DNS1: 192.168.0.200

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : param.point-ts.dk Used by sendlog, download param, TLCMDB  
IP : 80.164.132.228

Host to find : time.point-ts.dk Verifone time server  
IP : 80.164.132.227

Host to find : rtl.point-ts.dk Used by download program  
IP : 80.164.132.227

DNS2: 192.168.0.204 Same as above on Domain Name Server 2

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : param.point-ts.dk  
IP : 80.164.132.228

Host to find : time.point-ts.dk  
IP : 80.164.132.227

Host to find : rtl.point-ts.dk  
IP : 80.164.132.227

## **TLCMDB:**

[100]

NAME: test.point-ts.dk

PORT: 22000

RECV ERROR -3

Err Timeout

Connect [100] – FAILED

[101]

NAME: test.point-ts.dk

PORT: 22000

RECV ERROR -3

Err Timeout

Connect [101] – FAILED

[120]

NAME: test.point-ts.dk

PORT: 22000

Date: 20070122 14:53:03

Connect [120] – SUCCESS

[121]

NAME: test.point-ts.dk

PORT: 22000

RECV ERROR -3

Err Timeout

Connect [121] – FAILED

[181]

NAME: param.point-ts.dk

PORT: 24000

[182]

NAME: time.point-ts.dk

PORT: 13

2007-01-22 14:52

## **Contents of TLCMDB entries**

Who is contacted when TLCMDB entry 100 is chosen

Which port is chosen

Status on the get date/time that the terminal has made for entry 100

A timeout error has occurred

Date and time is shown when connection is made

It is not possible to make a test on entry 181 and 182

Verifone time server

## 2 | Local Payment Protocol

### 2.1 Preface

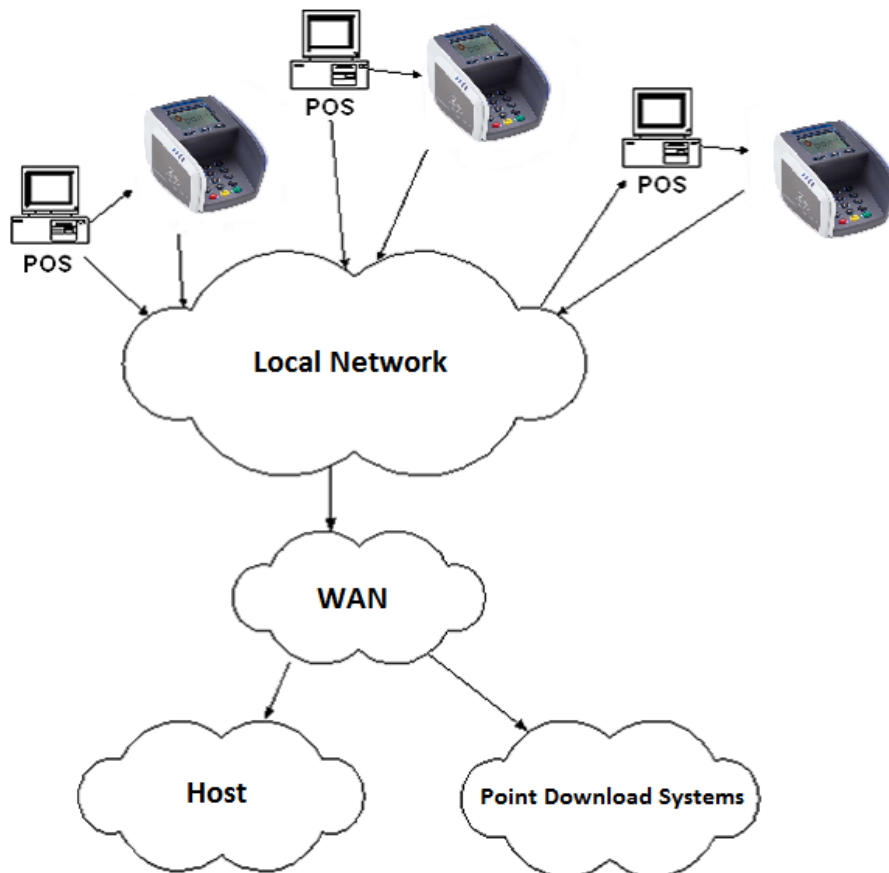
This document describes one of three different approaches for interfacing to the Flex Terminal. The approach described here is how to communicate directly with the terminal, using TLV structures embedded in the KISS communication protocol.

#### Revision

- |        |   |
|--------|---|
| 3.7.12 | Tags TCC - Transaction Condition Code and VAS - VAS Json receipt added.<br>Xenta and Xentissimo are no longer supported.  |
| 3.7.10 | Admin Contactless settings report.  |
| 3.5.03 | Admin Get/Set Teledone server ip/port.  |
| 3.5.02 | eKvittering added.<br>ECR connected via IP can now drop the ip connection or reboot terminal on another port.   |
| 3.5.00 | PSAM now controls all masking of carddata. So e.g. local cards can provide all carddata if "whitelisted" by Nets.<br>Support for Cashback amount (see Appendix 2.A.1 Tag definitions).<br>Admin functions to get and set the terminals IP settings (see Appendix 2.A.5 Command data).<br>If terminal is configured e-kvittering token is received already in CardData.<br>See section 2.7.2 The DATA Container. |

## 2.2 Hardware Overview

The terminal and ECR can communicate either by using a RS232 serial cable, connected between the terminals ECR port and a COM/serial port on the ECR, or by TCP/IP. When communicating by TCP/IP, no COM cable is used, hence the ECR and terminal communicate by the local network.



The illustration shows an example of a normal setup. Only three YOMANI terminals are shown, but there is no limit to the number of terminals in the network. The terminal can also communicate to the host and Verifone Denmark by ISDN.

### 2.2.1 The Terminal

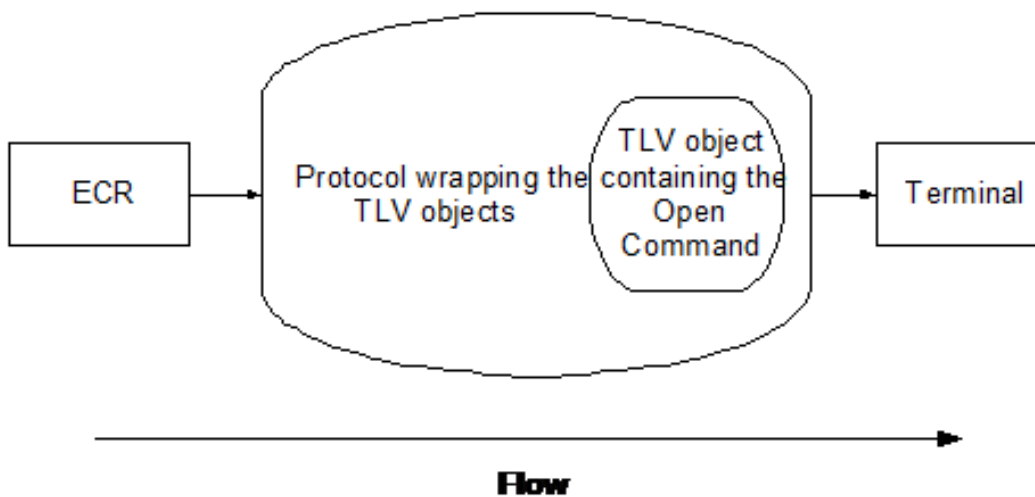
The terminal can be configured with four different communication modules. TCP/IP is only supported with the Ethernet module.

Communication module	RS232	TCP/IP
Ethernet	Supported	Supported
ISDN	Supported	Not supported
GSM	Supported	Not supported
PSTN	Supported	Not supported



## 2.3 Software Overview

The software interface is based on the ASN.1 TLV principle. This document provides a number of tags defining which information the tag withholds. All the transferred data objects are encoded in BER TLV – this document will give a description of the structure of TLV data objects. Finally, the tags are wrapped up in the KISS Protocol. The KISS Protocol is a serial byte oriented protocol, totally full duplex, asynchronous, and with no notion of master or slave. It is described elsewhere in this document. The illustration shows an example of an OPEN command:



## **2.4 Terminal Functionality**

The terminal functionality is fulfilling the Nets OTRS specification. It is recommended for the ECR developer to read the merchant section in the OTRS specification.

### **2.4.1 Initialization**

The initialization functions are used to make the terminal available to the user. The functions must be initiated from the ECR.

#### **2.4.1.1 Connect**

The connect command, is used to connect the terminal in 'DISCONNECTED' state, e.g. just after the terminal is booted. The ECR must send a 4 byte compatibility number to verify its SW compatibility (e.g. 0x02,0x00,0x00,0x00).

#### **2.4.1.2 Disconnect**

The disconnect command, is used to disconnect the terminal in the 'CONNECTED' state.

#### **2.4.1.3 Open**

The open command changes the text in the terminal display to 'TERMINALEN ER KLAR'. It is used if the terminal is in 'CONNECTED' mode.

#### **2.4.1.4 Close**

The close command changes the text in the terminal display from 'TERMINALEN ER KLAR' to 'VELKOMMEN'. It is used if the terminal is in 'OPEN' mode.

### **2.4.2 Transaction functions**

These functions are used to make transactions.

#### **2.4.2.1 PIN purchase transactions**

PIN based transaction is initiated from either the terminal, by swiping/inserting a card, or from the ECR, by sending purchase information to the terminal. If initiated from the terminal, the user must enter the PIN code and wait for the amount from the ECR. If initiated from the ECR, the user is requested to swipe/insert a card, enter a PIN code and press 'Godkend' ('OK'). This transaction always generates a receipt.

#### **2.4.2.2 Signature purchase transactions**

Signature based transactions must be initiated from the ECR. After initiating the transaction the user is requested to swipe/insert a card and approve the amount. This transaction type generates two receipts, unless the host rejects the transaction or a communication error occurs. In this case

only one receipt is generated. In case of successful host communication/verification, the first receipt is generated by the terminal, and immediately sent to the ECR. The receipt must be printed, the user must sign it, and the operator must verify the signature (depending on PSAM configuration). The second receipt is then generated based on the operator's decision if the functionality is available in the PSAM (signature accepted/rejected).

#### **2.4.2.3 Signature refund transactions**

Refund transactions must be initiated from the ECR. After initiating the transaction the user is requested to swiped/insert a card and approves the amount. This transaction type generates two receipts, unless the host rejects the transaction or a communication error occurs. In this case only one receipt is generated. In case of successful host communication/verification, the two receipts are generated by the terminal and immediately send to the ECR. Both receipts must be printed, and the operator must sign the customer receipt.

### **2.4.3 Administrative functionality**

The administrative functions are used to retrieve information from the terminal or set-up the terminal. Administrative functions can only be initiated from the ECR.

#### **2.4.3.1 The End of Day functionality**

End of Day routine must be called at least one time every 24 hours. It is used to empty and balance the terminals Datastore against the transaction inquirer host system. It is also used for PSAM updates.

#### **2.4.3.2 The unlock receipt functionality**

Unlock receipt request is used to fetch a copy of the last receipt from the terminals Datastore if the terminal is locked in the NO\_RECEIPT State. Booting the terminal will not unlock the terminal. It is not possible to initiate transactions in this state.

#### **2.4.3.3 The terminal report**

The terminal report gives vital information on the terminal configuration, e.g. SW and HW version, communication module etc.

#### **2.4.3.4 The clock synchronization**

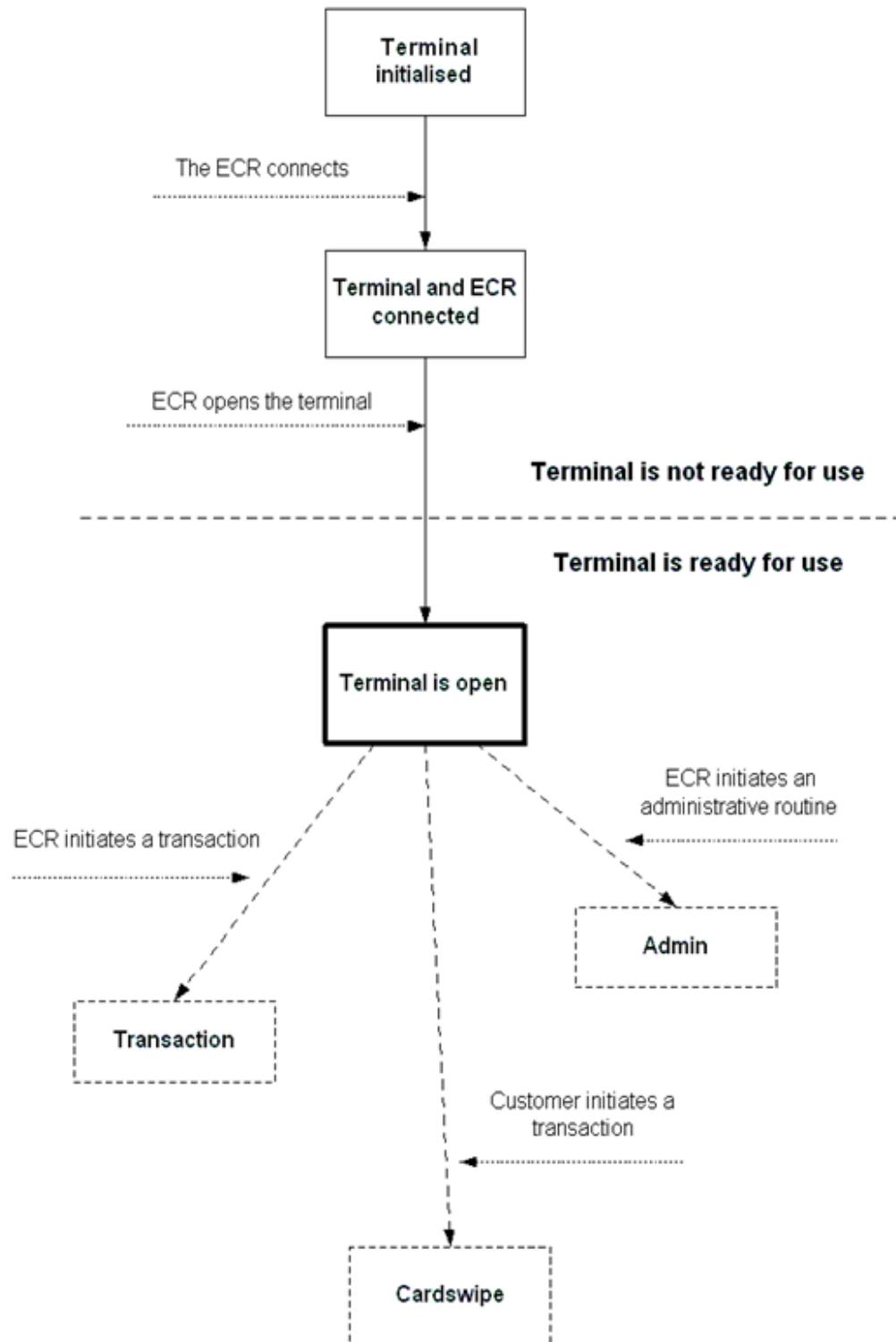
The clock synchronization function is very useful when testing the terminal's communication capabilities. E.g. to test the connection to Nets and/or Verifone Denmark when setting up the terminal in a store, or testing connection in case of communication errors during transactions.

#### **2.4.3.5 Debit/Credit Properties**

Debit/Credit properties return the status of previous transactions. This function is implemented from PSAM version 50.

#### **2.4.4 The Connect and Open procedure**

The terminal operates in many internal states, and it is only 'usable' for the customer and the ECR in the 'OPEN' state. Notice the sequence of connect and open.



## 2.5 The KISS protocol

The KISS protocol is used with communication to the terminal. The KISS protocol is serial bytes oriented protocol, full duplex, asynchronous, and with no notion of master or slave. It uses ASCII coding, and provides transparency over data field, by stuffing some characters (see Chapter Byte Stuffing). The connection is point-to-point. Messages are wrapped with a header and a tail to determine the beginning, the numbering, the end and the validity. A message has a maximum length of 2 KB. A message sent by the application is repeated up to 3 times (thus a total of 4 transmissions) by the protocol if the message is not acknowledged. If messages are lost or cannot be acknowledged the protocol will re-synchronize itself on the numbering of the first next correctly received message.

### 2.5.1 Approach – the KISS & BER–TLV interface

The rest of the document describes the direct approach to interfacing to the Smash terminal. This approach is based on the KISS protocol and TLV (Tag Length Value) objects.

### 2.5.2 Frame layout

There are two kinds of frames exchanged:

- Information frames
- Supervision frames

#### *Information frames*

Information frames consist of user data with a header, a trailer and a CRC (16-bit Cyclic Redundancy Check).

They have the following structure:

<b>STX</b>	<b>Sequence number</b>	<b>User data</b>	<b>ETX</b>	<b>CRC(LSB)</b>	<b>CRC(MSB)Name</b>
------------	------------------------	------------------	------------	-----------------	---------------------

#### *Supervision frames*

1. Affirmative acknowledge frame

<b>ACK</b>	<b>Sequence number</b>
------------	------------------------

2. Negative acknowledge frame

<b>NAK</b>
------------

3. Flow regulation frame (Wait for ACK)

<b>WACK</b>
-------------

All fields are described in detail in the next sections.

### 2.5.3 Control characters

#### ***STX (0x02) Start of Text***

##### *Definition*

The transmission control character STX precedes an information frame.

##### *Description of use*

1. An information frame must be preceded by STX.
2. STX resets the CRC computation to zero, when preceding an information frame.
3. If a User Data contains a character STX, a DLE character (Bytes stuffing) must precede it.
4. If STX appears in the sequence number, no stuffing is done.
5. STX is not included in the CRC computation.

#### ***ETX (0x03) End of Text***

##### *Definition*

A transmission control character, which terminates an information frame.

##### *Description of use*

1. ETX indicates the end of an information frame.
2. ETX calls for a reply ACK, NAK or WACK from the other party.
3. ETX signals that the next following 16 bits are the CRC characters.
4. ETX is included in the CRC computation.
5. If a text contains a character ETX, a DLE character (byte stuffing) must precede it.
6. If ETX appears in the sequence number, no stuffing is done.

#### ***DLE (0x10) Data Link Escape***

##### *Definition*

A transmission control character stuffs another control character in the data of an information frame.

##### *Description of use*

1. DLE informs the receiving station that the next character is to be treated as data and must not be considered as a control character.
2. DLE must precede STX, ETX and DLE when they are used in the data field of a message.
3. DLE is included in the CRC computation.
4. If DLE appears in the sequence number, no stuffing is done.

#### ***ACK (0x06) Affirmative Acknowledgment***

##### *Definition*

Transmission control character sent by a station as an affirmative response to the other station.

##### *Description of use*

1. ACK is transmitted by one station as an affirmative response to the other station.
2. ACK is used to indicate that the last transmitted information frame has been correctly received.
3. A sequence number follows ACK, which is identical to the number of the information frame to which an affirmative acknowledgment is given.

### **NAK (0x15) Negative Acknowledgment**

#### *Definition*

A transmission control character sent by a station as a negative response to the other station.

#### *Description of use*

1. NAK indicates that the last transmitted information frame was not received correctly, and the receiving station is ready to receive the same one. A sequence number does not follow NAK.

### **WACK (0x13) Wait for ACK (flow regulation)**

#### *Definition*

A transmission control character sent by a station as a request to the other station to wait for ACK. It is however possible that data is sent before this ACK.

#### *Description of use*

1. WACK is transmitted by one station as a request to wait for ACK
2. A sequence number does not follow WACK.
3. WACK is used to indicate that the last frame has been correctly received but has not yet been given to the application layer, because the buffers are not empty.
4. WACK will be repeated regularly (every time the receiving station tries to give the frame to the application layer).
5. There is no limit on the number of WACK's, which might be transmitted.
6. The time between successive WACK's must be less than the time-out on the ACK reception on the transmission station.
7. A WACK retriggers the ACK time-out at the transmission station.
8. When the frame is passed to the application layer, an ACK is sent.

### **Sequence Number (0x00 . . . 0xFF)**

#### *Definition*

A sequence number sent by both stations as a way to mark the frames. Once the sequence number has reached FF, it restarts at 00.

#### *Description of use*

1. The Sequence Number is a one-byte value, and is sent just after the STX in an information frame. The same value must be repeated in the affirmative acknowledgment.
2. The Sequence number is used to compute the CRC.
3. Each station manages its own transmission sequence numbering. It should be incremented after an Affirmative Acknowledgment or after an answer time-out.
4. The Sequence number is used to avoid that the same frame is given twice (or more) to the application.
5. Value 0xFF is the only value for which the receiving station always accepts the frame regardless of its previous sequence number.
6. No stuffing is done over the sequence number.
7. Upon reaching the value 0xFF, the sequence numbering should start at 0x00 again.



## **CRC computation**

### *Definition*

The CRC is computed over the sequence number and the ETX of a frame. It uses the polynomial  $X^{15} + X^{13} + 1$ , and is sent after the ETX closing the frame.

A frame is made of:

STX – Sequence number – data field – ETX – CRC (LSB) – CRC (MSB)

Upon reception of the STX, the LSB and MSB of the CRC field are initialised with zeroes. For each data byte between the Sequence Number (included) until and included the closing ETX, the following algorithm is applied:

table\_index = CRC(LSB) XOR data\_byte  
CRC(LSB) = CRC(MSB) XOR lsb(table\_index)  
CRC(MSB) = 0 XOR msb(table\_index)

Where:

LSB stands for Least Significant Byte

MSB stands for Most Significant Byte

XOR stands for eXclusive OR

Table is an array of 256 values, which is delivered together with the documentation package.

```
crctp[] = { 0x0, 0xc0c1, 0xc181, 0x140, 0xc301, 0x3c0, 0x280, 0xc241,
            0xc601, 0x6c0, 0x780, 0xc741, 0x500, 0xc5c1, 0xc481, 0x440,
            0xcc01, 0xcc0, 0xd80, 0xcd41, 0xf00, 0xcfc1, 0xce81, 0xe40,
            0xa00, 0xcac1, 0xcb81, 0xb40, 0xc901, 0x9c0, 0x880, 0xc841,
            0xd801, 0x18c0, 0x1980, 0xd941, 0x1b00, 0xdbc1, 0xda81, 0x1a40,
            0x1e00, 0xdec1, 0xdf81, 0x1f40, 0xdd01, 0x1dc0, 0x1c80, 0xdc41,
            0x1400, 0xd4c1, 0xd581, 0x1540, 0xd701, 0x17c0, 0x1680, 0xd641,
            0xd201, 0x12c0, 0x1380, 0xd341, 0x1100, 0xd1c1, 0xd081, 0x1040,
            0xf001, 0x30c0, 0x3180, 0xf141, 0x3300, 0xf3c1, 0xf281, 0x3240,
            0x3600, 0xf6c1, 0xf781, 0x3740, 0xf501, 0x35c0, 0x3480, 0xf441,
            0x3c00, 0xfcc1, 0xfd81, 0x3d40, 0xff01, 0x3fc0, 0x3e80, 0xfe41,
            0xfa01, 0x3ac0, 0x3b80, 0xfb41, 0x3900, 0xf9c1, 0xf881, 0x3840,
            0x2800, 0xe8c1, 0xe981, 0x2940, 0xeb01, 0x2bc0, 0x2a80, 0xea41,
            0xee01, 0x2ec0, 0x2f80, 0xef41, 0x2d00, 0x2dc1, 0x2c81, 0x2c40,
            0xe401, 0x24c0, 0x2580, 0xe541, 0x2700, 0xe7c1, 0xe681, 0x2640,
            0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0, 0x2080, 0xe041,
            0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281, 0x6240,
            0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441,
            0x6c00, 0xacc1, 0xad81, 0x6d40, 0xaf01, 0x6fc0, 0x6e80, 0xae41,
            0xaa01, 0x6ac0, 0x6b80, 0xab41, 0x6900, 0xa9c1, 0xa881, 0x6840,
            0x7800, 0xb8c1, 0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41,
```

```

0xbe01, 0x7ec0, 0x7f80, 0xbf41, 0x7d00, 0xbdc1, 0xbc81, 0x7c40,
0xb401, 0x74c0, 0x7580, 0xb541, 0x7700, 0xb7c1, 0xb681, 0x7640,
0x7200, 0xb2c1, 0xb381, 0x7340, 0xb101, 0x71c0, 0x7080, 0xb041,
0x5000, 0x90c1, 0x9181, 0x5140, 0x9301, 0x53c0, 0x5280, 0x9241,
0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x9481, 0x5440,
0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5f00, 0x9fc1, 0x9e81, 0x5e40,
0x5a00, 0x9ac1, 0x9b81, 0x5b40, 0x9901, 0x59c0, 0x5880, 0x9841,
0x8801, 0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x8a81, 0x4a40,
0x4e00, 0x8ec1, 0x8f81, 0x4f40, 0x8d01, 0x4dc0, 0x4c80, 0x8c41,
0x4400, 0x84c1, 0x8581, 0x4540, 0x8701, 0x47c0, 0x4680, 0x8641,
0x8201, 0x42c0, 0x4380, 0x8341, 0x4100, 0x81c1, 0x8081, 0x4040
}

```

Example of CRC computation:

station 1	0808A06D	1	07	
	28771383	5	6A	
station 2	08	0707A063	0707A063	
	68	2A781316	2A781316	
station 1	SSECC	N	AS	
	TE<data>	TRR	A	CE
	XQ	XCC	K	KQ
station 2	ASSS	ECC	SS	ECC
	CE	TE<data>	TRR	TE<data>
	KQ	XQ	XCC	XQ
				XCC

## 2.5.4 Time out controls

### ***Answer time-out***

The answer time-out is started at the transmitting station when a frame is sent. The answer time-out is re-triggered at the transmitting station when a WACK is received. The answer time-out is stopped at the transmitting station when an answer (ACK, NAK) is received from the other station. Note that the affirmative acknowledge is followed by the same sequence number as the frame just sent, to be considered as a valid affirmative acknowledgment. The time-out is 4 seconds.

### ***Inter character time-out***

The inter character time-out prevents a station having received a STX and some more characters, to stay in this state infinitely. The value of the inter character time-out is 2 ms.

### 2.5.5 Retry counter

The retry counter is the maximum number of times a message may be retransmitted in case of NAK response, or no answer at all. Value = 3.

### 2.5.6 Byte stuffing

The User data field allows transparent transmission, meaning that every byte in this field may have a valid value between 0 and 255. To avoid confusion between some values and control characters, a byte stuffing mechanism is implemented at both stations. The transmitting station scans every byte it received from its application, to find a STX, an ETX or a DLE control character. If it does, it stuffs this control character by adding a DLE just before this character. On the receiving station, once the leading STX and the Sequence Number are received, every DLE will be discarded, reconstituting by the same way the original User data. The DLE's added by this stuffing mechanism are used in the CRC computation.

### 2.5.7 Transmission Control Sequences

#### *Normal message transmission*

station 1	SSECC
TE<data>	TRR
XQ	XCC

station 2	AS
	CE
	KQ

#### *CRC error and retransmission accepted*

station 1	SSECC	SS	ECC	
	TE<data>	TRR	TE<data>	TRR
	XQ	XCC	XQ	XCC

station 2	N	AS
	A	CE
	K	KQ

#### *CRC error and re-transmissions refused*

station 1	SSECC	SS	ECC	SS	ECC	SS
	ECC					
	TE<data>	TRR	TE<data>	TRR		
	TE<data>	TRR				
	TE<data>	TRR				
	XQ	XCC	XQ	XCC	XQ	XCC
	XQ	XCC				

=>warns its application that the last message is lost

station 2	N	N	N	N
	A	A	A	A
	K	K	K	K

***Answer time-out and re-transmission accepted***

station 1	SSECC	SS		
	TE<data>	TRR<data>	TRR	
	XQ	XCC	XQ	XCC

=>time-out

station 2	AS
	CE
	KQ

station 1	SSECC	SS	ECC	SS	ECC	SS
	ECC					
	TE<data>	TRR	TE<data>	TRR		
	TE<data>	TRR				
	TE<data>	TRR				
	XQ	XCC	XQ	XCC	XQ	XCC
	XQ	XCC				

=>time-out   =>time-out   =>time-out   =>time-out

=>warns its application

station 2

***Inter character time-out and re-transmission accepted***

station 1	SSECC	ECC		
	TE<data>	TRR	TE<data>	TRR
	XQ	XQ	XCC	

station 2	N	AS
	A	CE
	K	KQ

=>inter character time out

***Normal full duplex exchange***

station 1	SSECC	AS	
	TE<data>	TRR	CE
	XQXCC	KQ	

station 2	SSECC	AS	
	TE<data>	TRR	CE
	XQ	XCC	KQ

***Information frame with out of order sequence number***

station 1	SSECC	SS	ECC	
	TE<data>	TRR	TE<data>	TRR
	XQ	XCC	XQ	XCC
		X		

station 2	ASAS		
	CE	CE	
	KQ	KQ	
	X		

=>warns its application for break in sequence numbering

station 1	SSECC	SS	ECC	
	TE<data>	TRR	TE<data>	TRR
	XQ	XCC	XQ	XCC
		X		

station 2	ASAS		
	CE	CE	
	KQ	KQ	
	X		

=>warns its application for break in sequence numbering

***ACK frame with out of order sequence number***

station 1	SSECC	SS	ECC	
	TE<data>	TRR	TE<data>	TRR
	XQ	XCC	XQ	XCC
		X		

=>invalid ACK. Fall in time out and re-send frame

station 2	ASAS		
	CE	CE	
	KQ	KQ	
	X		

***WACK frame followed by an ACK***

station 1	SSECC		
	TE<data>	TRR	
	XQ	XCC	

station 2	W W	AS
	A A	CE
	C C	KQ
	K K	

## 2.5.8 Error recovery

A negative (NAK) reply to an information frame causes re-transmission of the information frame. The maximal number of re-transmissions is equal to the message retry counter (see RETRY COUNTER). This implies that the information frame is sent a number of times equal to the retry counter plus one.

When the retry counter overflows, the sending station generates an alarm to its application layer, and takes the next information frame. The sequence number is incremented for this new information frame.

Station 1	SSECC	SS	ECC	SS	ECC	SS
	ECC	SS	ECC			
	TE<data>	TRR	TE<data>	TRR		
	TE<data>	TRR				
	TE<data>	TRR				
	XQ	XCC	XQ	XCC	XQ	XCC
	XQ	XCC	XQ	XCC		
	+					
	1					
	=>Warns application	retry counter exceeds				

Station 2	N	N	N	N	AS
	A A	A	A	CE	
	K K	K	K	KQ	
	+				
	1				
	=>Warns application	a frame has been lost			

An answer time-out on an information frame on the sending station, causes the re-transmission of the information frame a number of times equal to the value of the retry counter + 1. In case of unsuccessful re-sending, the sending station generates an alarm to its application layer, and take the next information frame. The sequence number is incremented for this new information frame.

Station 1	SSECC	SS	ECC	SS	ECC	SS
	ECC	SS	ECC			
	TE<data>	TRR	TE<data>	TRR		
	TE<data>	TRR				
	TE<data>	TRR				
	XQ	XCC	XQ	XCC	XQ	XCC
	XQ	XCC	XQ	XCC		

```

+
1
=>Warns to the application

```

```

Station 2      AS
                CE
                KQ
                +
                1
                =>Warns to the application a frame has been lost

```

An information frame having the same Sequence Number as the previous received information frame, must be acknowledged but discarded towards its application layer.

```

Station 1      SSECC    SS      ECC      SS      ECC
                TE<data> TRR      TE<data> TRR
                TE<new> TRR
                XQ      XCC      XQ      XCC      XQ      XCC
                +
                1

```

```

Station 2      ASAS      AS
                CE      CE      CE
                KQ      KQ      KQ
                +
                =>Give to application 1
                =>Discard! =>Give to application

```

An inter character time-out is treated in two different ways. The easiest way is not to answer and wait for the answer time-out to elapse, causing the re-transmission of the information frame.

A more efficient way to handle the inter character time-out is to generate a negative (NAK) reply, causing a faster re-transmission of the information frame. The final choice is left open for the designer.

## 2.5.9 Line supervision

To monitor the link between both stations, a so-called "I'm alive" mechanism is available. This is done by regularly sending an empty frame to the other station. Only one of both stations sends these messages: it is called the sender. The other station is called the receiver. Sender/receiver selection depends on the configuration.

Example:

The *External Device* is the sender. It sends an "I'm alive" message whenever it wants to know the connection is operational.

Message flow:

**The sender:**

Sends the "I'm alive" frames regularly

Reports to its application in case of no answer

After a reset, it sends “I’m alive” frames regularly and reports to its application when it receives the first ACK frame.

**The receiver:**

Answers all “I’m alive” frames with an ACK frame

Reports to its application in case of no reception.

Message layout for “I’m alive” frame:

STX	Sequence number	ETX	CRC(LSB)	CRC(MSB)
-----	-----------------	-----	----------	----------

The “I’m alive” frame is supported in the terminal, but it’s not mandatory for the ECR to send this frame.

## 2.5.10 Line characteristics

Asynchronous

Full duplex

1 start bit, 8 data bits, 1 stop bit

Parity: None

Speed: 4800 to 115200

Physical encoding method: Non Return to Zero (NRZ)

Byte serialisation: Least Significant Bit (LSB) first

Maximum user data length: Up to 2048 bytes

## 2.5.11 Function in C to calculate CRC

```
unsigned int CalcCRC(const unsigned char * pBuf, size_t sizeBuf)
{
    unsigned int i, cval, crc = 0;
    for(i=0; i<sizeBuf; i++)
    {
        cval = crc ^ (unsigned int) pBuf[i];
        crc = (crc >> 8) ^ crctp[cval & 0xff];
    }
    return crc; // crc & 0xFF is first byte of CRC
               // crc >> 8 is second byte of CRC
}
```

## 2.5.12 Functions in C to build KISS frame

```
#define PKISS_INIT_OUT 0x01
#define PKISS_INIT_IN 0x02
static int PKISS_CheckState(PKISS *pKiss, byte state)
{
    return pKiss->state & state;
}

static void PKISS_UnSetState(PKISS *pKiss, byte state)
```



```

{
    pKiss->state ^= (state & pKiss->state);
}

static byte PKISS_NewSeq(byte *pSeq)
{
    if (*pSeq >= 0xFE) {
        *pSeq = 0x00;
    } else {
        (*pSeq)++;
    }
    return *pSeq;
}

static int PKISS_Stuff(byte *pSBuf, const byte *pBuf, size_t bufSize)
{
    int res = 0;
    byte *pCursor, *pEnd, *pStart;
    if (bufSize)
    {
        pCursor = (byte*) pBuf;
        pStart = pSBuf;
        pEnd = pCursor + bufSize;
        while(pCursor < pEnd)
        {
            switch(*pCursor)
            {
                case STX :
                case ETX :
                case DLE :
                    *pSBuf++ = DLE;
                    *pSBuf++ = *pCursor++;
                    res+= 2;
                    break;
                default:
                    *pSBuf++ = *pCursor++;
                    res++;
                    break;
            }
        }
    }
    return res;
}

static void PKISS_BuildFrame(PKISS *pKiss, const byte *pData, size_t dataSize)
{
    uint fsize, crc;

    pKiss->oframe[0] = STX;
    if (PKISS_CheckState(pKiss, PKISS_INIT_OUT)) {
        PKISS_UnSetState(pKiss, PKISS_INIT_OUT);
        pKiss->oframe[1] = 0xFF;
        pKiss->oSeqNr = 0xFF;
    } else {

```

```

    pKiss->oframe[1] = PKISS_NewSeq(&pKiss->oSeqNr);
}
fsize = PKISS_Stuff(&pKiss->oframe[2], pData, dataSize);
pKiss->oframe[2 + fsize] = ETX;
crc = PCRC_CalcCRC(&pKiss->oframe[1], 2 + fsize);
pKiss->oframe[3 + fsize] = crc & 0xFF;
pKiss->oframe[4 + fsize] = crc >> 8;
pKiss->ofSize = fsize + 5;
pKiss->numberOfSends = 0;
}

```

### 2.5.13 Functions in C to verify KISS frame

```

static int PKISS_VerifySeq(PKISS *pKiss)
{
    int res = 0;
    byte recvSeq;

    recvSeq = pKiss->iframe[1];
    if (PKISS_CheckState(pKiss, PKISS_INIT_IN)) {
        res = 1;
    } else if (recvSeq == 0xFF) {
        res = 1;
    } else if ((recvSeq == 0x00) && (pKiss->iSeqNr >= 0xFE)) {
        res = 1;
    } else if (recvSeq == pKiss->iSeqNr) {
        res = 1;
    } else if ((recvSeq - 1) == pKiss->iSeqNr) {
        res = 1;
    }
    return res;
}

static int PKISS_VerifyFrame(PKISS *pKiss)
{
    int res = 0;
    uint calc_crc, recv_crc;

    calc_crc = PCRC_CalcCRC(&pKiss->iframe[1], pKiss->ifSize - 3);
    recv_crc = pKiss->iframe[pKiss->ifSize - 1] << 8;
    recv_crc += pKiss->iframe[pKiss->ifSize - 2];
    if (recv_crc == calc_crc)
    {
        pKiss->error = PKISS_OK;
        if (PKISS_VerifySeq(pKiss)) res = 1;
        else
        {
            if (pKiss->numberOfSends >= pKiss->maxNumberOfSends)
                pKiss->error = PKISS_SEQ_ERROR;
            fprintf(stderr, "Seq error...\n");
        }
    } else {
        if (pKiss->numberOfSends >= pKiss->maxNumberOfSends)
            pKiss->error = PKISS_CRC_ERROR;
    }
}

```

```
        fprintf(stderr,"CRC error... %x %x\n",recv_crc,calc_crc);
    }
    return res;
}
```

## 2.6 The BER–TLV principle

The data objects are encoded in BER–TLV – this section will give a brief description of the structure of TLV data objects.

### 2.6.1 Definition

A BER–TLV is an ISO standardized encoded representation of a Data Object. BER stands for Basic Rules of Encoding (of a data object) while TLV stands for Tag, Length and Value.

### 2.6.2 Properties

A TLV is a data object that encapsulates three properties Tag, Length and Value as follows:

TAG: a numerical value uniquely identifying the data object,

LEN: a numerical value specifying the length of data contained in the value property,

VAL: a data container for the actual value of the data object being encoded as a TLV.

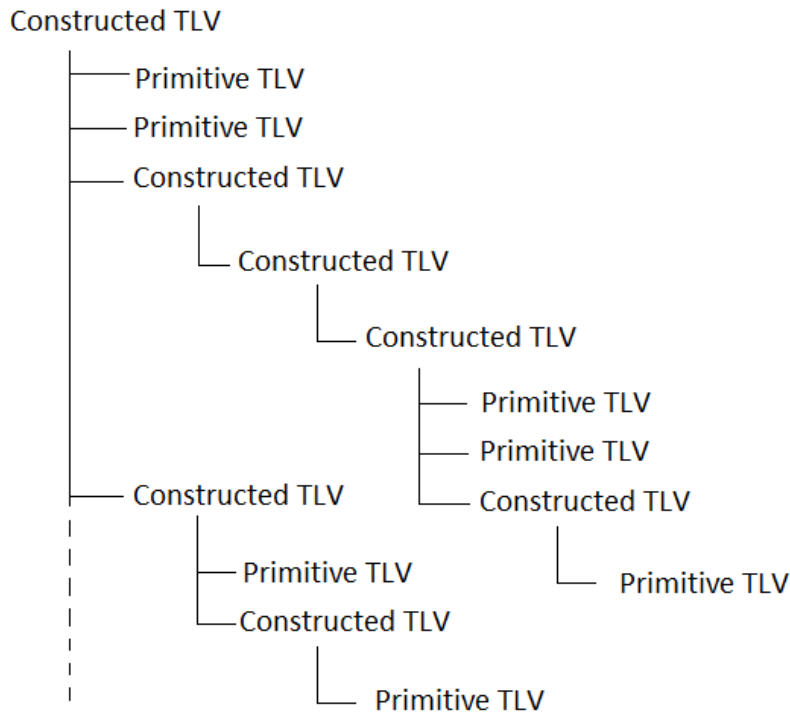
TLV's come in two forms: Constructed and Primitive.

**Constructed TLV:** is one that is used to embed one or more other TLV's in its value property. These embedded TLV's can themselves be either primitive or constructed.

A Constructed TLV can contain virtually any number of other constructed TLV's in there turn containing other embedded TLV's and so on.

A Constructed TLV could therefore be assimilated to a directory in a traditional file structure.

**Primitive TLV:** is one that is used to convey only simple data in its value property. It could be assimilated to a data File in a traditional file structure.



### 2.6.3 Motivation

The use of TLV to convey data between two devices is specified for the following reasons

- It is standardised following ISO specifications.
- Constructed TLV's can embed other TLV's which themselves if constructed may do the same, which allows dynamic use and assembling of DO's.
- Has been widely used, tested and documented.

### 2.6.4 Quick illustrated Description

The BER of TLV's are fully described in the SS-ISO 8825 series of standards. The reader is recommended therefore to read this document in conjunction with those standards.

The first byte in a TLV is the *Tag byte*, composed of 3 fields:

**Class** (2 bits) indicating its scope, **Form** (1 bit) indicating if it is constructed or primitive, and **Tagvalue** (5 bits) representing the actual value of the Tag.

**The Tag byte:** If the Tag value is greater than or equal to 31 then the first five bits of the first Tag byte will be 111112 indicating that there is at least one following *Tag byte*. This next *Tag byte* has a Flag bit (bit 8) indicating whether it is the last or not in a series of *Tag bytes*. The other bits (7 to 1) represent the actual *Tag value*.

**The Length byte:** has a flag bit (bit 8) set to zero when bits 7 to 1 suffice to represent the length otherwise set to 1 with bits 7 to 1 indicating the number of sub bytes representing the value of LEN. Sub bytes do not have flag bits. LEN indicates the number of Value bytes.

**The Value byte:** is only data, with Tag describing its content and Length its length in bytes.

*Tag octet - First Tag octet*

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Scope
								Class
0	0							Universal.
0	1							Application.
1	0							Context Specific.
1	1							Private.
								Form
		0						Primitive.
		1						Constructed.
								Value
			1	1	1	1	1	See next byte
			1 - 1E (hex)					EMV RESERVED
			0	0	0	0	0	Not used

*Tag octet - Following Tag octet*

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Description
0								Last Tag
1								See next Tag byte
1 - 7F (hex)								Tag ID or Value

*Length octet*

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Description
0								LEN < 128. Bits 7 → 1 = LEN
1								See next Tag byte
1 - 7F (hex)								Tag ID or Value

*Value octet*

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Description
0 - FF (hex)								Value

## 2.7 Data objects – Container tags

This section describes the various Container tag data objects, which are sent between the ECR and the terminal.

### 2.7.1 The INIT Container

The INIT container request from the ECR to the terminal is constructed like this:

Tag	Value		Presence
0x69	Init		Mandatory
	0x40	Connect	Mandatory*
	0x42	Disconnect	Mandatory*
	0x44	Open	Mandatory*
	0x46	Close	Mandatory*
	0x80	Binary	Optional

\*Only one of these four tags included. The Connect tag must include version number as four bytes (e.g. 0x02, 0x00, 0x00, 0x00).

The response from the terminal is constructed like this:

Tag	Value		Presence
0x69	Init		Mandatory
	0x84	Result	Mandatory
	0x80	Binary	Optional
	0x82	Text	Optional*
	0x90	ExtendedECR	Optional*
	0x86	State	Optional

\* The terminal id and extended ECR functions are included at Connect.

### 2.7.2 The DATA Container

The DATA container from the terminal is constructed like this:

Tag	Value		Presence
0x63	Data		Mandatory
	0x71	Local Card data	Mandatory <sup>2</sup>
	0x6F	Card data	Mandatory
	0x52	Card number	Mandatory <sup>1</sup>
	0x54	AID	Optional <sup>1</sup>
	0x86	State (binary)	Optional

<sup>1</sup> If the terminal supports Card Data Protection, the PAN returned will be truncated according to

the Card Schemes rules, i.e. leaving the first 6 and last 4 digits. The remaining digits may be replaced by “A”. The full PAN may be padded with a trailing “F” for byte boundary alignment if needed (see OTRS from Nets). <sup>2</sup> This tag is included in conjunction with Local Card transactions. The DATA Container reply from the ECR is constructed like this:

Tag	Value		Presence
<b>0x63</b>	<b>Data</b>		<b>Mandatory</b>
	<b>0x48</b>	<b>Amount</b>	<b>Optional<sup>1</sup></b>
	<b>0x6F</b>	<b>Card data</b>	<b>Mandatory</b>
	<b>0x71</b>	<b>Local Card data</b>	<b>Mandatory<sup>2</sup></b>
	<b>0x84</b>	<b>Result</b>	<b>Mandatory</b>

<sup>1</sup> If the transaction must continue at this point, the amount tag must be included and the result tag must be positive (0x00). If the transaction must be aborted the amount tag is optional but the result tag must be negative (0x01), result with value (0x03) will do a silent abort.

<sup>2</sup> This tag is included in conjunction with Local Card transactions.

### 2.7.3 The TRANSACTION Container

This container tag is used to initiate a transaction from the ECR, and can include several optional tags:

Tag	Value		Presence
<b>0x65</b>	<b>Transaction</b>		<b>Mandatory</b>
	<b>0x4E</b>	<b>MI</b>	<b>Optional<sup>1</sup></b>
	<b>0x4C</b>	<b>CU</b>	<b>Optional<sup>1</sup></b>
	<b>0x56</b>	<b>TT</b>	<b>Optional<sup>1</sup></b>
	<b>0x50</b>	<b>TR</b>	<b>Optional<sup>1</sup></b>
	<b>0x92</b>	<b>CARD_SOURCE</b>	<b>Optional<sup>1</sup></b>
	<b>0x93</b>	<b>PREPAID</b>	<b>Optional<sup>1</sup></b>
	<b>0x94</b>	<b>SCAN</b>	<b>Optional<sup>1</sup></b>
	<b>0x95</b>	<b>TERMINAL ENV</b>	<b>Optional<sup>1</sup></b>
	<b>0x4A</b>	<b>REF_NO</b>	<b>Optional<sup>1</sup></b>

<sup>1</sup> If not included, the terminal will use the default values defined in its param.ini file.

E.g. For purchase set MI to 0x00, TT to 0x00 and TR to 0x00; for refund set MI to 0x00, TT to 0x20 and TR to 0x01.

For KeyEntered transaction set CARD\_SOURCE to 0x10 0x02 (shows as DLE 02 in demo).



The reply from the terminal when the transaction finishes will be:

Tag	Value		Presence
<b>0x65</b>	<b>Transaction</b>		<b>Mandatory</b>
	<b>0x84</b>	<b>Result</b>	<b>Mandatory</b>
	<b>0x6B</b>	<b>Receipt</b>	<b>Mandatory</b> <sup>1</sup>
	<b>0x80</b>	<b>Binary</b>	<b>Optional</b>
	<b>0x86</b>	<b>State</b>	<b>Optional</b>
	<b>0x13</b>	<b>ASW1ASW2</b>	<b>Optional</b>
	<b>0x11</b>	<b>C9CA</b>	<b>Optional</b>
	<b>0x08</b>	<b>Acqmsg</b>	<b>Mandatory</b> <sup>1</sup>

<sup>1</sup> If the terminal have been communicating with the transaction acquirer, the receipt will always be present. If the user presses “Slet Alt” or the operator aborts the transaction before the transaction is send to the transaction inquirer, there will be no receipt. Advice Transfer will only appear if requested by Nets. The ECR may not run any further transactions before an endOfDay routine has been accomplished.

The reply can include an INFO container with host information (e.g. “LANGE SVARTIDER” from Nets).

NOTE: If the Terminal is configured to forward ip traffic to the ECR, the ENAI container will also appear in the reply.

## 2.7.4 The ADMIN Container

This container tag is used to initiate administrative functions on the terminal.

Tag	Value		Presence
<b>0x67</b>	<b>Admin</b>		<b>Mandatory</b>
	<b>0x88</b>	<b>Command</b>	<b>Mandatory</b>
	<b>0x10</b>	<b>STAN (bcd)</b>	<b>Optional</b> <sup>1</sup>
	<b>0x52</b>	<b>Card number</b>	<b>Optional</b> <sup>1</sup>

<sup>1</sup> Command values (0x80 and 0x81) will add either STAN or card number.

If the terminal supports Card Data Protection, the PAN returned will be truncated according to the Card Schemes rules, i.e. leaving the first 6 and last 4 digits. The remaining digits may be replaced by “A”. The full PAN may be padded with a trailing “F” for byte boundary alignment if needed (see OTRS from Nets).

The reply from the terminal when the administrative function finishes will be:

Tag	Value		Presence
0x67	Admin		Mandatory
	0x84	Result	Mandatory
	0x80	Binary	Optional
	0x86	State	Optional

NOTE: Reply for command value 0x80, 0x81 and 0x83, are embedded in an INFO container.

NOTE: If the Terminal is configured to forward ip traffic to the ECR, the ENAI container will also appear in the reply.

### 2.7.5 The INFO Container

This tag is used to carry information, which do not need any further action, e.g. status messages, and response to administrative command 0x80 and 0x81.

Tag	Value		Presence
0x60	Info		Mandatory
	0x00	TSI                      ascii	Optional
	0x82	Text                      ascii	Optional
	0x84	Result	Optional <sup>1</sup>
	0x06	Abort	Optional
	0x52	Card number <sup>2</sup> bcd	Optional
	0x10	Stan <sup>2,3</sup> bcd	Optional
	0x5C	Timestamp <sup>2,3</sup> DTHR format	Optional
	0x48	Amount <sup>3</sup>	Optional
	0x4C	Currency Code <sup>3</sup>	Optional
	0x80	Binary <sup>3</sup>	Optional
	0x86	State <sup>3</sup>	Optional

TSI: Transaction State Information.

<sup>1</sup> See transaction flow (transaction preresult).

<sup>2</sup> If the terminal supports Card Data Protection, the PAN returned will be truncated according to the Card Schemes rules, i.e. leaving the first 6 and last 4 digits. The remaining digits may be replaced by "A". The full PAN may be padded with a trailing "F" for byte boundary alignment if needed (see OTRS from Nets).

<sup>3</sup> These tags only when command 0x80 and 0x81.

<sup>3</sup> Currency Code (2 byte) embeds the currency exponent as 3. byte.

<sup>3</sup> When the ECR issues administrative command 0x80 and 0x81, the terminal may respond with Stan, Currency Code (+++) 2 byte, exponent 1 byte, Amount, Timestamp, or if requested transaction properties not found with ASW1-ASW2 code (usually 0x1033) as a binary tag.

## 2.7.6 The RECEIPT Container

This tag carries receipt information and it can be either nested in the transaction container or send as a standalone container.

Tag	Value		Presence
<b>0x6B</b>	<b>Receipt</b>		<b>Mandatory</b>
	<b>0x82</b>	Receipt Text	Mandatory
	<b>0x02</b>	Confirm <sup>2</sup>	Optional
	<b>0x95</b>	DCC rate <sup>3</sup>	Optional
	<b>0x0E</b>	Token (ascii)	Optional
	<b>0x80</b>	Binary Struct	Optional (2)
	<b>0x99</b>	E–receipt	Optional (8)
	<b>0x13</b>	Asw1Asw2	Optional (8)
	<b>0x5C</b>	Time (unix)	Optional (8)
	<b>0x52</b>	Card number (bcd)	Optional (8)
	<b>0x48</b>	Amount (binary)	Optional (8)
	<b>0x58</b>	Fee (binary)	Optional (8)
	<b>0x1C</b>	Extra (binary)	Optional (8)
	<b>0x59</b>	Gratuity (binary)	Optional (8)
	<b>0x4C</b>	Currency (bcd)	Optional (8)
	<b>0x54</b>	AID (ascii)	Optional (8)
	<b>0x0A</b>	ATC <sup>1</sup>	Optional (8)
	<b>0x0C</b>	AED <sup>1</sup>	Optional (8)
	<b>0x1E</b>	ARC <sup>1</sup>	Optional (8)
	<b>0x10</b>	STAN <sup>1</sup>	Optional (8)
	<b>0x12</b>	PSAMID <sup>1</sup>	Optional (8)
	<b>0x14</b>	ACODE <sup>1</sup>	Optional (8)
	<b>0x16</b>	CVM <sup>1</sup>	Optional (8)
	<b>0x18</b>	AutCode (ascii)	Optional (8)

Table continues on next page.

Tag	Value		Presence
	<b>0x5E</b>	Cardname (ascii)	Optional (8)
	<b>0x1E</b>	Terminal ID (ascii)	Optional (8)
	<b>0x4A</b>	Refnr (binary)	Mandatory
	<b>0x01</b>	Number (ascii)	Optional (4)
	<b>0x03</b>	Name (ascii)	Optional (4)
	<b>0x05</b>	City (ascii)	Optional (4)
	<b>0x07</b>	Address (ascii)	Optional (4)
	<b>0x09</b>	Zip (ascii)	Optional (4)
	<b>0x0B</b>	Phone (ascii)	Optional (4)
	<b>0x0B</b>	Brn. (ascii)	Optional (4)
	<b>0x86</b>	State	Optional
	<b>0x08</b>	Host msg	Optional
	<b>0x97</b>	Batch Number	Optional
	<b>0x98</b>	DCC currency	Optional
	<b>0x96</b>	Cancellation allowed	Optional
	<b>0x0E</b>	Token	Optional
	<b>0x0F</b>	PSAM Creator	Optional
	<b>0x92</b>	Card source	Optional
	<b>0x5A</b>	Card Reconciliation Counter id CRC	Optional
	<b>0xC6</b>	Nets Hash values from PSAM	see OTRS "Retrieve Hash Values" response
	<b>0xC7</b>	Storebox	

Optional (4): Merchant info bit set in terminal, Optional (8): Receipt information bit set in terminal.

If the terminal supports Card Data Protection, the PAN returned will be truncated according to the Card Schemes rules, i.e. leaving the first 6 and last 4 digits. The remaining digits may be replaced by "A". The full PAN may be padded with a trailing "F" for byte boundary alignment if needed (see OTRS from Nets).

1) EMV specific value (information in EMV specification).

2) See transaction flow, 3) DCC rate e.g. "1.000000".

The answer to a receipt must always look like this:

Tag	Value		Presence
<b>0x6B</b>	<b>Receipt</b>		<b>Mandatory</b>
	<b>0x84</b>	<b>Result</b>	<b>Optional</b>

### 2.7.7 The ERROR Container

This tag is used to carry error information from the terminal to the ECR.

Tag	Value		Presence
<b>0x6C</b>	<b>Error</b>		<b>Mandatory</b>
	<b>0x80</b>	<b>Error code</b>	<b>Mandatory</b>
	<b>0x86</b>	<b>State</b>	<b>Mandatory</b>

ECR must only handle Error codes 3, depending on the terminal state. Error codes other than 3 are for internal use.

## 2.7.8 The HANDLERSTR Container

This tag is used to carry handler string information.

Tag	Value		Presence
<b>0x73</b>	<b>HandlerString</b>		<b>Mandatory</b>
	<b>0x80</b>	<b>Data*</b>	<b>Mandatory</b>

\* Variable length host data.

The ECR does not answer this operation.

## 2.7.9 The HOSTDATA Container

This tag is used to carry host data information.

Tag	Value		Presence
<b>0x73</b>	<b>HostData</b>		<b>Mandatory</b>
	<b>0x80</b>	<b>Data</b>	<b>Mandatory</b>

See below host data flow.

## 2.7.10 The MENU Container

This tag is used to carry menu information.

The terminal menu request to the ECR is constructed like this:

Tag	Value		Presence
<b>0x77</b>	<b>Menu</b>		<b>Mandatory</b>
	<b>0x88</b>	<b>Command</b>	<b>Mandatory</b> <sup>1</sup>
	<b>0x82</b>	<b>Text</b>	<b>Mandatory</b>
	<b>0x80</b>	<b>Binary</b>	<b>Mandatory</b>
	<b>0x48</b>	<b>DCC CH Amount</b>	<b>Optional</b> <sup>2</sup>
	<b>0x4C</b>	<b>DCC CH Currency</b>	<b>Optional</b> <sup>2</sup>
	<b>0x58</b>	<b>DCC ME Amount</b>	<b>Optional</b> <sup>2</sup>
	<b>0x14</b>	<b>DCC CH Currency Exp</b>	<b>Optional</b> <sup>2</sup>
	<b>0x86</b>	<b>State</b>	<b>Mandatory</b>

The Command tag is 1 for Application list, 2 for ECR card inserted correct and 3 for DCC.

The Text tag holds menu text lines that are 20 chars long plus a termination zero, giving a total of 21 chars per line and maximum 10 lines (total 210 bytes).

For command 3 (DCC) text holds the DCC pre receipt text for printing (max 2048 bytes).

The Binary tag includes a timeout value. The response from the ECR must include the selected line number in the Result tag, within the time of the timeout value.

If the Command tag is 2 the text is "KORT ISAT KORREKT?", the ECR respond must include 1 for "JA" or 0 or -1 for "NEJ", within the time of the timeout value, otherwise the transaction is terminated ("AFBRUDT").

\*\* CH Amount, CH Currency, CH Currency ACode (ch currency exponent) and ME Amount (ME currency always DKK) only with DCC and command 3, and the ECR must respond 1 for selection of DCC currency or 0 for not selection, within the given timeout value.

The response from the ECR is constructed like this:

Tag	Value		Presence
<b>0x77</b>	<b>Menu</b>		<b>Mandatory</b>
	<b>0x84</b>	<b>Result</b>	<b>Optional</b>

### 2.7.11 The ENAI Container

This container is used to carry data information and commands for External Network Application Interface, e.g. routing of data to and from Nets through the ECR's serial connection. If configured the terminal will embed and route all IP traffic to the ECR through the serial interface. The ECR must create and IP connection with the specified parameters and forward the traffic to the connection.

The ENAI container will only be transmitted during transactions and administrative commands (clock sync).

The format from the Terminal:

Tag	Value		Presence
0x79	ENAI		Mandatory
	0x80	Binary	Optional
	0x8A	IP Address	Optional
	0x8C	IP Port	Optional
	0x8E	IP Timeout	Optional
	0x88	Command	Optional

The Terminal will initiate an IP transmission with the Command 0x01 (connect) and the IP Address, IP Port and IP Timeout are present.

IP Address is up to 15 chars e.g. 80.164.132.229.

IP Port is a 4 byte integer e.g. 22000 (decimal).

IP Timeout is a 4 byte integer e.g. 30000 (for 30 seconds).

The ECR must create a connection with the specified parameters and reply the Result to the Terminal.

The terminal will discontinue an IP connection with the Command 0x02 (disconnect). The ECR must close the connection and reply the result to the Terminal.

The Terminal will transmit data with the Binary tag, and the ECR must forward these data to the connection.

If the transmit data to be transmitted in the Binary tag exceed 1000 bytes, the Result is set to 0x5f to indicate more packets to come, after one or more packets of 1000 bytes, the remainder (last packet) is indicated by setting Result to 0x5a.

The ECR will forward received data with the Binary tag to the Terminal.

If the received data to be transmitted in the Binary tag exceed 1000 bytes, the Result is set to 0x5f to indicate more packets to come, after one or more packets of 1000 bytes, remainder (last packet) is indicated by setting Result to 0x5a.

MAX\_TCP\_WINDOW\_SIZE is 65535, so up to 65 packets of 1000 bytes and one of 535 should be allowed.

The reply format from the ECR:

Tag	Value		Presence
0x79	ENAI		Mandatory
	0x80	Binary	Optional
	0x84	Result 1 byte	Optional

#### 2.7.11.1 ENAI Connect reply

For the connect command Result equal 0x00 (one byte) indicate OK, Result equal 0x01 indicate an ERROR.

### 2.7.11.2 ENAI CardSwipe reply

If no Binary tag, and the Result is 0x5a a card has been inserted in the terminal (CardSwipe).

### 2.7.11.3 ENAI Card Removed reply

If no Binary tag, and the Result is 0x5b a card has been removed from the terminal, this will only be send if a transaction still running. Use the administrative function “check card state” to poll the status after transaction completes.

### 2.7.11.4 ENAI Send Data reply

If Binary tag, and Result is 0x5f more packets of data to come, ECR must assemble the full packet before passing it on to the TCP/IP connection.

If Binary tag and Result is 0x5a it's the last packet, ECR must pass on all the data received to the TCP/IP connection.

If Binary tag and Result is 0x5c the TCP/IP connection used internal in the terminal was lost.

### 2.7.11.5 ENAI Syncframe

If Result is 0x16 – Terminal send sync frame to indicate to the ECR it's started. The Binary tag contain four bytes 0x55, 0xa5, 0x5a, 0xaa. ECR should reply with ACK SEQ and an ENAI container with Result set to 0x16, and Binary tag set to the four bytes: 0x5a, 0xaa, 0xa5, 0x55 first time it receives the sync frame from the terminal, to indicate it's ready to make a IP routing connection.

Any next time it see the sync from the terminal, Binary tag must be set to the four bytes 0xaa, 0xa5, 0x55, 0x5a to indicate to the terminal ECR ready to process IP routing packets.

### 2.7.11.6 ENAI sync frame – start of program download example

```
TimeStamp: 09:14:52
STX       : 02
SEQ       : ff
  TAG      : 79 - PTAG_ENAI Container
  TAGlen   : 09
    TAG     : 80 - PTAG_BINARY
    TAGlen  : 04
    TAGvalue : 55 a5 5a aa 'UYZ'

    TAG     : 84 - PTAG_RESULT
    TAGlen  : 01
    TAGvalue : 16 - Unknown result '.'

ETX       : 03
CRC       : 0193
```



TimeStamp: 09:14:52  
ACK : 06  
SEQ : ff

TimeStamp: 09:14:52  
STX : 02  
SEQ : ff  
TAG : 79 - PTAG\_ENAI Container  
TAGlen : 09  
TAG : 84 - PTAG\_RESULT  
TAGlen : 01  
TAGvalue : 16 - Unknown result ',.'  
  
TAG : 80 - PTAG\_BINARY  
TAGlen : 04  
TAGvalue : 5a aa a5 55 'ZYU'

ETX : 03  
CRC : 7b61

TimeStamp: 09:14:52  
ACK : 06  
SEQ : ff

TimeStamp: 09:14:52  
STX : 02  
SEQ : ff  
TAG : 79 - PTAG\_ENAI Container  
TAGlen : 20  
TAG : 88 - PTAG\_COMMAND  
TAGlen : 01  
TAGvalue : 01 ',.'  
  
TAG : 8a - PTAG\_IPADDR  
TAGlen : 0f  
TAGvalue : 72 74 6c 2e 70 6f 69 6e 74 2d 74 73 2e 64 6b  
'rtl.point-ts.dk'  
  
TAG : 8c - PTAG\_IPPORT  
TAGlen : 04  
TAGvalue : 00 00 14 5e '...^'

TAG : 8e - PTAG\_IPTIMEOUT  
TAGlen : 04  
TAGvalue : 00 06 68 a0 '...h '

ETX : 03  
CRC : dbda

TimeStamp: 09:14:52  
ACK : 06  
SEQ : ff

TimeStamp: 09:15:19  
STX : 02  
SEQ : ff  
TAG : 79 - PTAG\_ENAI Container  
TAGlen : 09  
TAG : 80 - PTAG\_BINARY  
TAGlen : 04  
TAGvalue : 55 a5 5a aa 'UYZ'  
  
TAG : 84 - PTAG\_RESULT  
TAGlen : 01  
TAGvalue : 16 - Unknown result '.'

ETX : 03  
CRC : 0193

TimeStamp: 09:15:19  
ACK : 06  
SEQ : ff

TimeStamp: 09:15:19  
STX : 02  
SEQ : 00  
TAG : 79 - PTAG\_ENAI Container  
TAGlen : 09  
TAG : 84 - PTAG\_RESULT  
TAGlen : 01  
TAGvalue : 16 - Unknown result '.'  
  
TAG : 80 - PTAG\_BINARY

```

TAGlen      : 04
TAGvalue    : aa a5 55 5a          'YUZ'

ETX         : 03
CRC         : 695d

TimeStamp: 09:15:19
ACK         : 06
SEQ         : 00

Slam: ff - j

TimeStamp: 09:15:34
STX         : 02
SEQ         : ff
TAG         : 79 - PTAG_ENAI Container
TAGlen      : 09
TAG         : 80 - PTAG_BINARY
TAGlen      : 04
TAGvalue    : 55 a5 5a aa          'UYZ'
TAG         : 84 - PTAG_RESULT
TAGlen      : 01
TAGvalue    : 16 - Unknown result  ', '

ETX         : 03
CRC         : 0193

TimeStamp: 09:15:34
ACK         : 06
SEQ         : ff

TimeStamp: 09:15:34
STX         : 02
SEQ         : 01
TAG         : 79 - PTAG_ENAI Container
TAGlen      : 09
TAG         : 84 - PTAG_RESULT
TAGlen      : 01
TAGvalue    : 16 - Unknown result  ', '

TAG         : 80 - PTAG_BINARY
TAGlen      : 04
TAGvalue    : aa a5 55 5a          'YUZ'

```

ETX : 03  
CRC : 6bdc

TimeStamp: 09:15:34  
ACK : 06  
SEQ : 01

### 2.7.12 The STOPLIST tag

The stoplist tag is used to inform the ECR to return an authorization code (issued by Nets, and usually obtained before start of the transaction). The stoplist is checked for transaction type offline. The ECR must return the authorization code using the stoplist tag and including 8 bytes (the tag length must be eight), where the first six chars are the authorization code received from Nets followed by two zeros.

The default code is defined as six spaces.

Tag	Value	Presence
0x5B	STOPLIST 0-8 bytes	Mandatory

## 2.8 Flows of TLV data objects

### 2.8.1 The flow of a Connect command

To connect to the terminal, the following data must be sent to the terminal:

*INIT*

|

*CONNECT ( value is e.g. 0x00 and 0x01 ) – see INIT for more detail.*

The immediate feedback from the terminal will be:

*INIT*

|

*RESULT ( value is e.g. 0x00 )*

*BINARY ( value is e.g. 0x00 and 0x01 )*

*TEXT ( value is terminal id e.g. '00990071' )*

*ExtendedECR (value is e.g. 0x00 0x00 0x00 0x00)*

*STATE ( optional )*

The result will be either 0x00 for successful or 0x01 for unsuccessful.

### 2.8.2 The flow of a Disconnect command

To disconnect from the terminal, the following data must be sent to the terminal:

*INIT*

|

*DISCONNECT (empty)*

The immediate feedback from the terminal will be:

*INIT*

|

*RESULT*

*STATE (optional)*

The result will be either 0x00 for successful or 0x01 for unsuccessful.

### 2.8.3 The flow of an Open command

To open the terminal, the following data must be sent to the terminal:

*INIT*

|

*OPEN (empty)*

*BINARY (optional)*

The immediate feedback from the terminal will be:

*INIT*

|

*RESULT*

*BINARY (optional)*

*STATE (optional)*

The result will be either 0x00 for successful or 0x01 for unsuccessful.

## 2.8.4 The flow of a Close command

To close the terminal, the following data must be sent to the terminal:

*INIT*

|

*CLOSE (empty)*

The immediate feedback from the terminal will be:

*INIT*

|

*RESULT*

*STATE (optional)*

The result will be either 0x00 for successful or 0x01 for unsuccessful.

## 2.8.5 The flow of a PIN Purchase

To initiate a transaction, the following data must be sent to the terminal:

*TRANSACTION*

|

*MI ( e.g. 0x0 for default transaction type )*

*CU ( e.g. 208 for DKK )*

*TT ( purchase or refund )*

*TR ( purchase or refund )*

*AMOUNT ( purchase or refund amount)*

*GRATUITY*

*VAT*

*BACK ( cashback amount )*

*REF\_NO (optional )*

The MI, CU, TT, TR and the REF\_NO tags are all optional. If not sent, default values can be found in the appendix.

If terminal configured for contactless transactions

The AMOUNT is the minimum required

GRATUITY, VAT and BACK are all optional but required if used later.

This speed up the flow of a contactless transaction, now the terminal know the amount, gratuity, vat and cashback amount at transaction start. Enabling the terminal to present the customer with the full amount before customer tap a contactless card, inserting a ICC card or swipe a MSC card.

The immediate feedback from the terminal on the TRANSACTION tag will either be:

*INFO*

|

*TSI ( "Afventer kort" )*

*BINARY ( 0x01 → status line 1 )*

*STATE ( optional )*

If the card hasn't been swiped before sending a purchase request. The BINARY tag is optional. If

the card has been swiped before sending a purchase request:

Note that in certain fallback situations TSI is sent twice.

*DATA*

|

*CARDDATA*

|

*CARDNUMBER ( e.g. 4571... ) PCI padded*

*AID ( optional and only if applicable (EMV))*

|

*STATE (optional)*

If the card is accepted, the ECR must respond with the 4-byte AMOUNT tag:

*DATA*

|

*CARDDATA*

|

*RESULT ( 0x00 )*

|

*AMOUNT ( e.g. 19995 )*

*FEE (4 bytes)*

The FEE tag (optional) is directly related to the fee amount (DKK øre), which must be added to the transaction.

And the terminal replies:

*INFO*

|

*TSI ( "Afventer PIN/beløb" )*

*BINARY ( terminal suggests: status line 1 )*

*STATE ( optional )*

Note that in certain fallback situations CARDDATA is sent twice.

Or if the card is not accepted, the ECR must reply negative to the DATA tag, like this:

*DATA*

|

*CARDDATA*

|

*RESULT ( 0x01 )*

The terminal replies a termination tag:

## TRANSACTION

- |
  - RESULT (0x01)
  - RECEIPT (0x01)
- |
  - TEXT ("OLES PIZZABAR... etc.")
- STATE (optional)
- REF\_NO (optional)

The ECR must acknowledge the receipt by sending this tag:

### RECEIPT

- |
  - RESULT (0x00 or 0x01)

If the ECR application accepts the card, the terminal will continue in the PIN ENTRY state and the state changes when the user presses Godkend:

### INFO

- |
  - TSI ( "Vent (arbejder)" )
  - BINARY ( status line 1 )
  - STATE ( optional )

Followed by several status messages:

### INFO

- |
  - TEXT ( "Opkald..." )
  - PAN (the full primary account number)
  - STAN (a unique Nets reference number)
  - TIME (transaction timestamp in DTHR format)
  - BINARY ( status line 2)
  - STATE ( optional )

### INFO

- |
  - TEXT ( "Sender..." )
  - BINARY ( status line 2)
  - STATE ( optional )

Etc...

To make sure that the ECR receives the result of the transaction as soon as possible, the terminal sends the expected(!) transaction result, before completing the transaction with the PSAM.

### INFO

- |
  - RESULT(expected(!) transaction result)

At this point, the terminal will always generate a receipt, which also enclosed the final result of the transaction:



*TRANSACTION*

```
|  
RESULT (0x00 or 0x01)  
RECEIPT  
|  
TEXT ("SMASH Terminal... etc.")  
|  
STATE (optional)  
BINARY (optional)  
REF_NO (optional)
```

Note that in certain fallback situations the TRANSACTION container is sent twice.  
The ECR must, as always, acknowledge the receipt by sending this tag:

*RECEIPT*

```
|  
RESULT( 0x00 or 0x01 )
```

If the TRANSACTION tag isn't received by the ECR, the expected transaction result can be used to complete the transaction on the ECR. However, the transaction must be handled manually (when convenient), to verify the result. If the RECEIPT tag is not sent from the ECR (cable error etc.), the terminal will enter the "Ingen Kvittering" status, and can only be unlocked if the ECR requests a receipt unlock (ADMIN with "unlock receipt" as the COMMAND). Note: This "Ingen Kvittering" state has been removed from terminal SW version 3.4.

## 2.8.6 The flow of a Signature Purchase

To initiate a signature transaction, the following data must be sent to the terminal:

*TRANSACTION*

```
|  
MI ( e.g. 0x82 for Signature transaction type )  
CU ( Optional )  
TT ( 0x00 purchase )  
REF_NO (optional )
```

The CU and the REF\_NO tags are all optional. If not sent, default values are used (can be found in the appendix). The immediate feedback from the terminal on the TRANSACTION tag will be:

*INFO*

```
|  
TSI ( "Afventer kort" )  
BINARY ( 0x01 → status line 1 )  
STATE ( optional )
```

When the card is swiped, the following data is sent to the ECR:

DATA

- | CARDDATA
  - | CARDNUMBER ( e.g. 4571... )
  - | AID ( optional and only if applicable (EMV) )
- | STATE ( optional )

If the card is accepted, the ECR must respond with the 4-byte AMOUNT tag:

DATA

- | CARDDATA
  - | RESULT ( 0x00 )
- | AMOUNT ( e.g. 19995 )
- | FEE ( 4 bytes )

And the terminal replies:

INFO

- | TSI ( "Afventer beløb" )
- | BINARY ( terminal suggests: status line 1 )
- | STATE ( optional )

Or if the card is not accepted, the ECR must reply negative to the DATA tag, like a PIN transaction.

If the ECR application accepts the card, the terminal will continue in the PIN ENTRY state and the state changes when the user presses "Godkend".

INFO

- | TSI ( "Vent (arbejder)" )
- | BINARY ( status line 1 )
- | STATE ( optional )

Followed by several status messages:

*INFO*

|  
*TEXT* ( "Opkald..." )  
*BINARY* ( *status line 2* )  
*STATE* ( *optional* )

*INFO*

|  
*TEXT* ( "Sender..." )  
*BINARY* ( *status line 2* )  
*STATE* ( *optional* )

*Etc...*

The first receipt, which the customer must sign is sent first:

*RECEIPT*

|  
*TEXT* ( "SMASH Terminal...etc." )  
*CONFIRM* ( *optional* )  
*STATE* ( *optional* )  
*BINARY* ( *optional* )  
*REF\_NO* ( *optional* )

The answer to this receipt must be positive (0x00) to accept the customer signature and negative (0x01) to reject the transaction. If the CONFIRM tag appears, then the ECR/operator is requested to acknowledge the signature. This must be done within 3 minutes.

*RECEIPT*

|  
*RESULT* ( *0x00 or 0x01* )

The terminal will reply with either an approved receipt or a rejected receipt.

*TRANSACTION*

|  
*RESULT* ( *0x00 or 0x01* )  
*RECEIPT*  
|  
*TEXT* ( "SMASH Terminal...etc." )  
|  
*STATE* ( *optional* )  
*BINARY* ( *optional* )  
*REF\_NO* ( *optional* )

The ECR must also acknowledge the receipt by sending this tag:

*RECEIPT*

|  
*RESULT* ( *0x00 or 0x01* )

### 2.8.7 The application menu functionality

This functionality is used during refund transactions when the card requests a menu to be shown on the ECR display. The OTRS Specification specifies at least 10 lines to be shown. The MENU container is also used if the terminal detects an inserted chip card, but cannot read/write from the card. A text message “Card inserted correctly?” is then to be shown on the ECR display.

*MENU*

└  
*COMMAND, 1 byte*  
*TEXT, 21 bytes pr. menu*  
*BINARY*

COMMAND holds the line number to be displayed. TEXT holds 21 bytes per line (‘\ 0’ terminated text).

The binary tag holds the timeout (to display the menu) in milliseconds. If the value is 0, no timeout is used (infinity). The binary tag is optional.

The ECR must reply with the following tag:

*MENU*

└  
*RESULT*

The RESULT tag must hold the value for the selected menu point, e.g. if the first line is selected RESULT is 1. If timeout is expired RESULT is -2. If RETURN is not within the number of lines, the terminal response is undefined.

### 2.8.8 The Advice log functionality

The terminal is able to send copies of datastore operations to the ECR. This is done via the HANDLESTR tag. The ECR can then save a copy of the terminal datastore to be used in conjunction with error handling (primarily with offline transactions). If this functionality is activated (terminal parameter) the transaction time is increased by half approx a second. The information is given directly to the ECR. No reply is needed.

*HANDLERSTR*

└  
*BINARY*

### 2.8.9 Fee functionality

If the terminal must calculate a fee, instead of the ECR. The calculated fee is sent directly to ECR when the amount is known by the terminal. No reply is needed (DATA tag is even).

For return transactions with gratuity this amount can be added.

*DATA ( 0x62 !! )*

|  
*FEE ( 4 bytes )*  
*GRATUITY ( 4 bytes )*

### 2.8.10 Get terminal files functionality

The ECR is able to request certain files from the terminal. This functionality can only be used if the terminal is in IDLE state/mode, hence the ECR is NOT “connected” to the terminal.

The ECR must send the following tags to the terminal:

*DATAGRAM*

|  
*COMMAND (“GET; <remote filename >”)*

The COMMAND tag must include the name on the file to get, e.g. “GET;/dk/otrs.log”.

The terminal will reply with the following tags:

*DATAGRAM*

|  
*COMMAND (“PUT;<filename >” or “PUTF;<filename >”)*  
*BINARY ( binary data segment )*

If the terminal replies “PUT;<filnavn >”, it means that there are more data segments in the file.

The ECR must then reply:

*DATAGRAM*

|  
*RESULT (0x00)*

The flow loops until the ECR receives a “PUTF”, which means that the entire file has been received.

### 2.8.11 Abort from ECR

It is possible to abort the transaction from the ECR at any time by sending the abort tag (ABORT tags from the ECR are ignored by the terminal after the user presses “Godkend” and the transaction is transmitted to the host):

*INFO*

|  
*ABORT (empty)*

The terminal will reply with the termination tag:

*TRANSACTION*

|  
*RESULT ( 0x01 )*  
*BINARY ( optional )*  
*STATE ( optional )*  
*REF\_NO ( optional )*

### **2.8.12 The flow of a Local Card transaction (initiated by ECR)**

To initiate a Local Card transaction, the following data must be sent to the terminal, similar to a normal transaction:

*TRANSACTION*

|  
*MI ( e.g. 0x0 for default transaction type )*  
*CU ( e.g. 208 for DKK )*  
*TT ( purchase or refund )*  
*TR ( purchase or refund )*  
*REF\_NO (optional )*

The MI, CU, TT and the REF\_NO tags are all optional. If not sent, default values can be found in the appendix. The immediate feedback from the terminal on the TRANSACTION tag will either be:

*INFO*

|  
*TSI ( "Afventer kort" )*  
*BINARY ( 0x01 → status line 1 )*  
*STATE ( optional )*

If the card have NOT been swiped before sending a purchase request. The BINARY tag is optional. If the card have been swiped before sending a purchase request:

*DATA*

|  
*LOCALCARDATA*  
|  
*CARDNUMBER*  
*AID ( optional and only if applicable (EMV))*  
|  
*STATE (optional)*

The LOCALCARDATA tag states that a Local Card have been swiped. Now the ECR must reply with the DATA tag. If the amount must be shown on the terminal, the ECR must attach the AMOUNT tag. Otherwise no amount is shown on the terminal:

DATA

```
└─ LOCALCARDDATA
   └─ RESULT ( 0x00 )
      └─ AMOUNT (optional)
```

And the terminal replies:

```
INFO
└─ TSI ( "Afventer beløb" )
   └─ BINARY ( terminal suggests: status line 1 )
      └─ STATE ( optional )
```

Or if the card is not accepted, the ECR must reply negative to the DATA tag, like this:

```
DATA
└─ LOCALCARDDATA
   └─ RESULT ( 0x01 )
```

The terminal will reply with a termination tag:

```
TRANSACTION
└─ RESULT (0x01)
   └─ RECEIPT
      └─ TEXT (empty)
         └─ STATE (optional)
            └─ BINARY (optional)
               └─ REF_NO (optional)
```

The ECR must acknowledge the receipt by sending this tag:

```
RECEIPT
└─ RESULT ( 0x00 or 0x01 )
```

If the ECR application accepts the card, the terminal will show the amount and continue sending the following tag when the user presses "Godkend" or "Slet Alt".

```
HOSTDATA
└─ BINARY ( 0x00 if Godkend and 0x01 if Slet Alt )
```

The HOSTDATA tag means that the terminal is now waiting for the ECR to approve/reject the transaction. The ECR must reply with the HOSTDATA tag hold the value of the "transaction result".

*HOSTDATA*

└  
*BINARY ( TAPA message )*

The TAPA message 0x03 is approved and 0x07 is rejected The terminal will generate beeps and display texts similar to a PIN transaction.

*TRANSACTION*

└  
*RESULT ( 0x00 or 0x01 )*  
*STATE ( optional )*  
*BINARY ( optional )*  
*REF\_NO ( optional )*

### 2.8.13 The flow of a Local Card transaction (initiated by the user)

If the cardholder swipes the card before the transaction is initiated by the ECR, the following tags are sent to the ECR (terminal parameter specific). Please notice that the two container tags DATA and LOCALCARDATA has different values, because no reply is needed:

*DATA (0x62)*

└  
*LOCALCARDATA (0x70)*

└  
*CARDNUMBER*  
*AID ( optional and only if applicable (EMV))*

└  
*STATE (optional)*

The ECR can now initiate the transaction:

*TRANSACTION*

└  
*MI ( e.g. 0x0 for default transaction type )*  
*CU ( e.g. 208 for DKK )*  
*TT ( purchase or refund )*  
*TR ( purchase or refund )*  
*REF\_NO ( optional )*

The MI, CU, TT and the REF\_NO tags are all optional. If not sent, default values can be found in the appendix. The immediate feedback from the terminal on the TRANSACTION tag will either be:

*INFO*

└  
*TSI ( "Afventer kort" )*  
*BINARY ( 0x01 → status line 1 )*  
*STATE ( optional )*

If the card have NOT been swiped before sending a purchase request. The BINARY tag is optional. If the card have been swiped before sending a purchase request:



DATA

```
└
  LOCALCARDDATA
    └
      CARDNUMBER
      AID ( optional and only if applicable (EMV))
    └
      STATE ( optional)
```

The LOCALCARDDATA tag states that a Local Card have been swiped. Now the ECR must reply with the DATA tag. If the amount must be shown n the terminal, the ECR must attach the AMOUNT tag. Otherwise no amount is shown on the terminal:

DATA

```
└
  LOCALCARDDATA
    └
      RESULT ( 0x00 )
    └
      AMOUNT ( optional )
```

And the terminal replies:

INFO

```
└
  TSI ( "Afventer beløb" )
  BINARY ( terminal suggests: status line 1 )
  STATE ( optional )
```

Or if the card is not accepted, the ECR must reply negative to the DATA tag, like this:

DATA

```
└
  LOCALCARDDATA
    └
      RESULT ( 0x01 )
```

The terminal will reply with a termination tag:

TRANSACTION

```
└
  RESULT ( 0x01 )
  RECEIPT
    └
      TEXT ( empty)
  STATE ( optional )
  REF_NO ( optional )
```

The ECR must acknowledge the receipt by sending this tag:

RECEIPT

```
└
  RESULT ( 0x00 or 0x01 )
```

If the ECR application accepts the card, the terminal will show the amount and continue sending the following tag when the user presses “Godkend” or “Slet Alt”.

*HOSTDATA*

└  
*BINARY ( 0x00 if Godkend and 0x01 if Slet Alt )*

The HOSTDATA tag means that the terminal is now waiting for the ECR to approve/reject the transaction. The ECR must reply with the HOSTDATA tag hold the value of the “transaction result”.

*HOSTDATA*

└  
*BINARY ( TAPA message )*

The TAPA message 0x03 is approved and 0x07 is rejected the terminal will generate beeps and display texts similar to a PIN transaction.

*TRANSACTION*

└  
*RESULT ( 0x00 or 0x01 )*  
*STATE ( optional )*  
*BINARY ( optional )*  
*REF\_NO (optional )*

## **2.8.14 The flow of an endofday**

To initiate an “End of Day” (balancing) request, the following data must be sent to the terminal:

*ADMIN*

└  
*COMMAND ( 0x02 for End of Day )*

The immediate feedback from the terminal will be:

*INFO*

└  
*TEXT ( “Afstemning” )*  
*BINARY ( status line 1 )*  
*STATE ( optional )*

Followed by several status messages:

*INFO*

└  
*TEXT ( “Afstemning” )*  
*BINARY ( status line 1 )*  
*STATE ( optional )*

Followed by several status messages:

*INFO*

└  
*TEXT ( “Opkald...” )*  
*BINARY ( terminal suggests status line 2 )*  
*STATE ( optional )*

INFO

└  
TEXT ( "Sender..." )  
BINARY ( terminal suggests status line 2 )  
STATE ( optional )

Etc...

The end of day routine always delivers a receipt:

RECEIPT

└  
TEXT ( "SMASH Terminal...etc." )  
BINARY ( part of a larger segment, or final segment )  
STATE ( optional )

Etc...

When the receipt is safe (on your harddrive), you need to send this tag within 3 seconds from receiving the receipt:

RECEIPT (empty)

└  
RESULT ( 0x00 or 0x01 )

The immediate terminal answer will be either a new receipt:

RECEIPT

└  
RESULT ( 0x00 or 0x01 )

The immediate terminal answer will be either a new receipt:

RECEIPT

└  
TEXT ( "SMASH Terminal...etc." )  
RESULT ( 0x00 or 0x01 )  
BINARY ( part of a larger segment, or final segment )  
STATE ( optional )

Or a termination tag:

ADMIN

└  
RESULT ( 0x00 or 0x01 )  
STATE ( optional )

The ECR needs to acknowledge every receipt (if several) within 3 seconds. However, the termination tag (including result) is automatically sent from the terminal after 6 seconds even though the receipt isn't acknowledged.

## 2.8.15 The flow of a Terminal report

To initiate a terminal report request, the following data must be sent to the terminal:

ADMIN

└  
COMMAND ( 0x01 for terminal report )

The immediate feedback from the terminal will be:

*INFO*

|  
*TEXT* ( *"Terminal rapport"* )  
*BINARY* ( *terminal suggests status line 1* )  
*STATE* ( *optional* )

Always followed by a receipt:

*RECEIPT*

|  
*TEXT* ( *"SMASH Terminal...etc."* )  
*STATE* ( *optional* )

When the receipt is safe (e.g. on the harddrive), you need to sent this tag within 5 seconds from receiving the receipt:

*RECEIPT* (empty)

The immediate answer will be:

*ADMIN*

|  
*RESULT* ( *0x00 or 0x01* )  
*BINARY* ( *optional* )  
*STATE* ( *optional* )

## **2.8.16 The flow of a Last transaction receipt request**

The last transaction receipt functionality can only be used in conjunction with transactions. Which means that it cannot be used to fetch old terminal reports or logs. To initiate a last request, the following data must be sent to the terminal:

*ADMIN*

|  
*COMMAND* ( *0x0F for "last receipt"* )

The immediate feedback from the terminal will be:

*INFO*

|  
*TEXT* ( *"Sidste kvittering"* )  
*BINARY* ( *terminal suggests status line1* )  
*STATE* ( *optional* )

Always followed by a receipt including a result tag which holds the result of that particular transaction:

*RECEIPT*

|  
*TEXT*  
*RESULT* ( *0x00 or 0x01* )  
*REF\_NO* ( *optional* )  
*STATE* ( *optional* )

When the receipt is safe (on a harddrive), the ECR must respond by sending this tag within 5 seconds from receiving the receipt:

*RECEIPT*

|  
*RESULT*

The immediate answer from the terminal (in case of a PIN based transaction) will be:

*ADMIN*

|  
*RESULT ( 0x00 or 0x01 )*  
*BINARY ( optional )*  
*STATE ( optional )*

Or if the last transaction generated two receipts ( signature or refund ):

*RECEIPT*

|  
*TEXT*  
*REF\_NO ( optional )*  
*STATE ( optional )*

This must be acknowledged with an empty RECEIPT tag. If the terminal is locked in “Ingen kvittering” state when requesting the last receipt, the terminal will only unlock if the receipts are acknowledged. The “Ingen Kvittering” state has been removed from terminal SW version 3.4.

### **2.8.17 Advice transfer flag**

The Advice Transfer flag is sent from the ECR when the transaction inquires requires an endOfDay routine. The Advice Transfer flag is sent (if sent) together with the TRANSACTION TAG, like this:

*TRANSACTION*

|  
*RESULT ( 0x00 or 0x01 )*  
*RECEIPT*  
  
|  
*TEXT ( “SMASH Terminal... etc. ” )*  
  
|  
*STATE ( optional )*  
*BINARY ( optional )*  
*REF\_NO (optional )*  
*ACQMSG ( mandatory, if present )*

### **2.8.18 The flow of the Get DC properties**

To initiate a get dc properties request, the following data must be sent to the terminal (in open state):

*ADMIN*

|  
*COMMAND ( 0x80 STAN search key or 0x81 PAN is search key )*  
*STAN (or PAN)*

The immediate feedback from the terminal will be:

INFO

```
|  
STAN (optional)  
AMOUNT (optional)  
CU (optional)  
TIME (optional)  
BINARY (0 for OK 2 for "PSAM Not Ready/wrong version")  
STATE ( optional )
```

STAN, AMOUNT, CU (with exponent embedded) and TIME is only present if OK.

The immediate answer will be:

ADMIN

```
|  
RESULT ( 0x00 or 0x01 )
```

### 2.8.19 The flow of the ENAI

The Terminal can be configured to forward (route) IP traffic through the serial connection to the ECR. If so the ECR must be able to handle transmission and receiving if the ip traffic.

The traffic is embedded in the ENAI container and appears in the transaction flow and for some selected administrative functions (clock sync).

This example illustrates the flow during a transaction. Note that the example only covers the ENAI flow, other containers can mix into the flow.

The Terminal sends:

ENAI

```
|  
COMMAND (0x01 for connect)  
IPADDR ("90.164.132.229")  
IPPORT (22000)  
IPTIMEOUT (30000)
```

The immediate feedback from the ECR may be:

ENAI

```
|  
RESULT (0x00)
```

During the transaction the Terminal will send one or more data packets.

ENAI

```
|  
BINARY (..... data to send .....)
```

ENAI

```
|  
BINARY (..... data to send .....)
```

When ip traffic is received by the ECR it will send:

ENAI

```
|  
BINARY (.....received data.....)  
RESULT ( 0x5A )
```

When the session is ended the Terminal sends:

*ENAI*

|

*COMMAND (0x2 for disconnect)*

The ECR need not answer, but must disconnect the session immediately.

## 2.9 Extra Application Protocol

Extra application uses new tag PTAG\_EXTRA\_APP – 0x53 to communicate the extra application data to/from the terminal.

### 2.9.1 Extra application data from terminal to ECR

Data is put into a PTAG\_DATA container, this have two fields PTAG\_BINARY is used to hold a copy of the app\_no PTAG\_EXTRA\_APP hold the 4 byte header and up to 1000 bytes data.

### 2.9.2 Example Extra application data from terminal to ECR

Here app\_no 12, version 0x00, error 0x00, last\_block 0xff

```
TimeStamp: 12:33:28
STX       : 02
SEQ       : 05
TAG       : 63 - PTAG_DATA Container
TAGlen    : 59
TAG       : 80 - PTAG_BINARY
TAGlen    : 01
TAGvalue  : 0c                ', '

TAG       : 53 - PTAG_EXTRA_APP
TAGlen    : 48
TAGvalue  : 0c 00 00 ff 45 78 74 72 61 20 61 70 70 20 68 61
'...ÿExtra app ha'
TAGvalue  : 73 20 72 65 63 65 69 76 65 64 20 34 34 20 64 61 's
received 44 da'
TAGvalue  : 74 61 20 62 79 74 65 73 20 61 6e 64 20 70 65 72
'ta bytes and per'
TAGvalue  : 66 6f 72 6d 65 64 20 73 6f 6d 65 20 61 63 74 69
'formed some acti'
TAGvalue  : 6f 6e 20 6f 6e 20 69 74
'on on it'

TAG       : 89 - PTAG_SEQNO
TAGlen    : 01
TAGvalue  : 01                ', '

TAG       : 85 - PTAG_FROMSTATE
TAGlen    : 01
TAGvalue  : 07 - PSTATE_OPEN    ', '

TAG       : 87 - PTAG_EVENT
TAGlen    : 01
TAGvalue  : 6c - PEVENT_SIG_EXTRA2HOST '1'

TAG       : 86 - PTAG_STATE
TAGlen    : 01
TAGvalue  : 07 - PSTATE_OPEN    ', '

ETX       : 03
```



CRC : ae35

### 2.9.3 Extra application data from ECR to terminal

Data is put into a PTAG\_DATA\_1 container, this have one field PTAG\_EXTRA\_APP this hold the 4 byte header and up to 1000 bytes data.

### 2.9.4 Enable ExtraReply in the terminal

Use the ADMIN\_GET\_ECR\_EXTENDED\_FUNCTIONS to get current EcrExtendedFunctions value or the value with 0x0100 and use the ADMIN\_SET\_ECR\_EXTENDED\_FUNCTIONS.

TimeStamp: 09:23:57

STX : 02

SEQ : 03

TAG : 67 - PTAG\_ADMIN Container

TAGlen : DLE 03

TAG : 88 - PTAG\_COMMAND

TAGlen : 01

TAGvalue : 85 - ADMIN\_GET\_ECR\_EXTENDED\_FUNCTIONS , ,

ETX : 03

CRC : 48b2

TimeStamp: 09:23:57

ACK : 06

SEQ : 03

TimeStamp: 09:23:57

STX : 02

SEQ : 04

TAG : 60 - PTAG\_INFO Container

TAGlen : 17

TAG : 82 - PTAG\_TEXT

TAGlen :

00 , ,

TAG : 90 - PTAG\_ECREXTENDEDFUNCTIONS

TAGlen : 04

TAGvalue : 00 00 DLE 02

04 , . . . . ,

TAG : 80 - PTAG\_BINARY

TAGlen : 01

TAGvalue : 00 , ,

TAG : 89 - PTAG\_SEQNO

TAGlen : 01

TAGvalue : DLE 03 , . ,

TAG : 85 - PTAG\_FROMSTATE

TAGlen : 01

TAGvalue : 07 - PSTATE\_OPEN , ,

```

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 1f - PEVENT_ECR_ADMIN          ' .'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 2b - PSTATE_ADMIN              ' +'

ETX      : 03
CRC      : 2192

TimeStamp: 09:23:57
ACK      : 06
SEQ      : 04

TimeStamp: 09:23:57
STX      : 02
SEQ      : 05
TAG      : 67 - PTAG_ADMIN Container
TAGlen   : 0f
TAG      : 84 - PTAG_RESULT
TAGlen   : 01
TAGvalue : 00 - OK                        ' .'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : DLE 03                         ' ..'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 2b - PSTATE_ADMIN              ' +'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 1e - PEVENT_ECR_TRANSACTION_COMPLETED ' .'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 07 - PSTATE_OPEN               ' .'

ETX      : 03
CRC      : 4f55

Now we have the current EcrExtendedFunctions value 0x0204 and need to set the 0x0100 bit by
making EcrExtendedFunctions = EcrExtendedFunctions | 0x0100 giving the result 0x0304 and we
call the admin function ADMIN_SET_ECR_EXTENDED_FUNCTIONS to set it.

TimeStamp: 09:23:57
STX      : 02
SEQ      : 04
TAG      : 67 - PTAG_ADMIN Container
TAGlen   : 09
TAG      : 88 - PTAG_COMMAND
TAGlen   : 01
TAGvalue : 84 - ADMIN_SET_ECR_EXTENDED_FUNCTIONS ' .'

```

```

TAG      : 90 - PTAG_ECREXTENDEDFUNCTIONS
TAGlen   : 04
TAGvalue : 00 00 DLE 03
04                                     '....'

ETX      : 03
CRC      : e8f3

TimeStamp: 09:23:57
ACK      : 06
SEQ      : 04

TimeStamp: 09:23:57
STX      : 02
SEQ      : 06
TAG      : 67 - PTAG_ADMIN Container
TAGlen   : 0f
TAG      : 84 - PTAG_RESULT
TAGlen   : 01
TAGvalue : 00 - OK                      ',.'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 04                          ',.'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 2b - PSTATE_ADMIN           ',+'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 1e - PEVENT_ECR_TRANSACTION_COMPLETED ',.'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 07 - PSTATE_OPEN            ',.'

ETX      : 03
CRC      : 8729

```

## 2.9.5 Tell terminal ECR listen for extra app data

We use a special version 0xfd and a status by to tell the terminal ECR listen for data or has stopped to listen for data from the extra apps in the terminal.

Status 0x00: ECR stopped listen for extra apps data

Status 0x01: ECR listen for extra apps data

Ex. ECR listen for extra apps data

Here app\_no 12, version 0xfd, error 0x00, last\_block 0xff and Status 0x01

TimeStamp: 14:53:19

STX : 02

SEQ : 05

```

TAG      : 62 - PTAG_DATA_1 Container
TAGlen   : 07
  TAG     : 53 - PTAG_EXTRA_APP
  TAGlen  : 05
  TAGvalue : 0c fd 00 ff
           01                                     '.ÿ.ÿ.'
```

```

ETX      : 03
CRC      : 9918
```

```

TimeStamp: 14:53:19
ACK      : 06
SEQ      : 05
```

## 2.9.6 Example LPP Extra application data from ECR to terminal

Here app\_no 12, version 0x00, error 0x00, last\_block 0xff

```

TimeStamp: 12:33:28
STX      : 02
SEQ      : 03
  TAG     : 62 - PTAG_DATA_1 Container
  TAGlen  : 33
    TAG   : 53 - PTAG_EXTRA_APP
    TAGlen : 31
    TAGvalue : 0c 00 00 ff 53 6f 6d 65 20 74 65 73 74 20 64 61
              '...ÿSome test da'
    TAGvalue : 74 61 20 74 6f 20 45 78 74 72 61 20 74 65 72 6d
              'ta to Extra term'
    TAGvalue : 69 6e 61 6c 20 61 70 70 6c 69 63 61 74 69 6f 6e
              'inal application'
    TAGvalue : 00                                     ',.'
```

```

ETX      : 03
CRC      : 13d2
```

## 2.10 Examples

Examples on how to communicate with the terminal, made with the C# PointKasseDemoLPP\_TAPA3\_Thread demo - no longer supported.

1. Connect
2. Open
3. Close
4. Disconnect
5. ClockSyncNets
6. Extended Issuer Envelope

### 2.10.1 Connect

Send to terminal:

```
STX : 02
SEQ : FF
TAG  : 69 - PTAG_INIT Container
TAGlen : DLE 10
TAG  : 40 - PTAG_CONNECT
TAGlen : 04
TAGvalue : DLE 03 01 00 00 '....'
TAG  : 80 - PTAG_BINARY
TAGlen : DLE 02
TAGvalue : 00 00 '..'
TAG  : 90 - PTAG_ECREXTENDEDFUNCTIONS
TAGlen : 04
TAGvalue : 00 00 00 04 '....'
ETX : 03
CRC : 1944
```

Acknowledge from terminal:

```
ACK : 06
SEQ : FF
```

Terminal response to connect:

```
STX : 02
SEQ : FF
TAG      : 69 - PTAG_INIT Container
TAGlen    : 23
TAG      : 84 - PTAG_RESULT
TAGlen    : 01
TAGvalue  : 00 - OK
TAG      : 80 - PTAG_BINARY
```

```

TAGlen      : DLE 02
TAGvalue    : 00 01                                '...'

TAG         : 82 - PTAG_TEXT
TAGlen      : 08
TAGvalue    : 39 39 30 33 36 39 00 00            '990369..'

TAG         : 90 - PTAG_ECREXTENDEDFUNCTIONS
TAGlen      : 04
TAGvalue    : 00 00 00 04                        '....'

TAG         : 89 - PTAG_SEQNO
TAGlen      : 01
TAGvalue    : FF                                'ÿ'

TAG         : 85 - PTAG_FROMSTATE
TAGlen      : 01
TAGvalue    : DLE 03 - PSTATE_IDLE '...'

TAG         : 87 - PTAG_EVENT
TAGlen      : 01
TAGvalue    : 19 - PEVENT_ECR_CONNECT_OK '...'

TAG         : 86 - PTAG_STATE
TAGlen      : 01
TAGvalue    : 05 - PSTATE_CONNECTED '...'

```

```

ETX : 03
CRC : 86C7

```

Send acknowledge to terminal

```

ACK : 06
SEQ : FF

```

## 2.10.2 Open

Send to terminal:

```

STX : 02
SEQ : 00
TAG      : 69 - PTAG_INIT Container
TAGlen   : DLE 02
TAG      : 44 - PTAG_OPEN

```

```

TAGlen      : 00 ''

ETX : 03
CRC : c956

ACK : 06
SEQ : 00

```

Terminal response to open with configuration parameters:

```

STX : 02
SEQ : 00

TAG      : 69 - PTAG_INIT Container
TAGlen   : 81 dc
TAG      : 84 - PTAG_RESULT
TAGlen   : 01
TAGvalue : 00 - OK '. '

TAG      : 80 - PTAG_BINARY
TAGlen   : DLE 02
TAGvalue : 00 00 '.. '

TAG      : 82 - PTAG_TEXT
TAGlen   : 81 c0
TAGvalue : 39 39 30 33 36 39 2c 31 37 2c 33 35 2c 31 2c 31 '990369,17,35,1,1'
TAGvalue : 2c 31 2c 30 2c 31 2c 30 2c 30 30 30 30 31 37 ',1,0,1,0,0000017'
TAGvalue : 33 35 33 2c 54 65 73 74 20 52 65 74 61 69 6c 20 '353,Test Retail'
TAGvalue : 44 43 43 20 20 20 2c 42 61 6c 6c 65 72 75 70 20 'DCC ,Ballerup '
TAGvalue : 20 20 20 20 20 20 2c 4c 61 75 74 72 75 70 62 ' ,Lautrupb'
TAGvalue : 6a 65 72 67 20 31 30 20 20 20 2c 2b 34 35 20 34 34 ',2750 ,+45 44'
TAGvalue : 36 38 34 34 36 38 20 20 20 20 20 20 20 20 20 20 '684468 '
TAGvalue : 20 20 2c 20 20 20 20 20 20 20 20 20 20 2c ' , , '
TAGvalue : 31 36 2c 20 64 61 2c 32 35 30 2c 32 35 35 2c 32 '16, da,250,255,2'
TAGvalue : 30 38 2c 32 30 38 2c 30 2c 33 34 2c 20 20 20 20 '08,208,0,34, '
TAGvalue : 34 2e 30 30 30 30 2c 30 2c 33 2e 33 2e 30 30 00 '4.0000,0,3.3.00.'

TAG      : 90 - PTAG_ECREXTENDEDFUNCTIONS
TAGlen   : 04
TAGvalue : 00 00 00 04          '....'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 00          ', '

```

TAG : 85 - PTAG\_FROMSTATE  
TAGlen : 01  
TAGvalue : 05 - PSTATE\_CONNECTED       ',.'

TAG : 87 - PTAG\_EVENT  
TAGlen : 01  
TAGvalue : 1a - PEVENT\_ECR\_OPEN       ',.'

TAG : 86 - PTAG\_STATE  
TAGlen : 01  
TAGvalue : 07 - PSTATE\_OPEN       ',.'

ETX : 03  
CRC : 9fa5  
Send acknowledge to terminal:  
ACK : 06  
SEQ : 00

### 2.10.3 Close

Send to terminal:

STX : 02  
SEQ : 01

TAG : 69 - PTAG\_INIT Container  
TAGlen : DLE 02  
TAG : 46 - PTAG\_CLOSE  
TAGlen : 00       '',

ETX : 03  
CRC : 7856

Acknowledge from terminal:

ACK : 06  
SEQ : 01

Terminal response to close:

STX : 02  
SEQ : 01  
TAG : 69 - PTAG\_INIT Container  
TAGlen : 0f



```

TAG      : 84 - PTAG_RESULT
TAGlen   : 01
TAGvalue : 00 - OK          ' .'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 01              ' .'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 07 - PSTATE_OPEN ' .'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 1c - PEVENT_ECR_CLOSE ' .'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 05 - PSTATE_CONNECTED ' .'

```

```

ETX : 03
CRC : 3b97

```

Send acknowledge to the terminal:

```

ACK : 06
SEQ : 01

```

#### 2.10.4 Disconnect

Send to terminal:

```

STX : 02
SEQ : 02
TAG      : 69 - PTAG_INIT Container
TAGlen   : DLE 02
TAG      : 42 - PTAG_DISCONNECT
TAGlen   : 00          ' '

```

```

ETX : 03
CRC : 0a97

```

### 2.10.5 Disconnect

Acknowledge from terminal:

ACK : 06

SEQ : 02

Terminal response to disconnect:

STX : 02

SEQ : 02

TAG : 69 - PTAG\_INIT Container

TAGlen : 0f

TAG : 84 - PTAG\_RESULT

TAGlen : 01

TAGvalue : 00 - OK '.'

TAG : 89 - PTAG\_SEQNO

TAGlen : 01

TAGvalue : DLE 02 '.'

TAG : 85 - PTAG\_FROMSTATE

TAGlen : 01

TAGvalue : 05 - PSTATE\_CONNECTED '.'

TAG : 87 - PTAG\_EVENT

TAGlen : 01

TAGvalue : 1b - PEVENT\_ECR\_DISCONNECT '.'

TAG : 86 - PTAG\_STATE

TAGlen : 01

TAGvalue : DLE 03 - PSTATE\_IDLE '.'

ETX : 03

CRC : d35e

Send acknowledge to the terminal:

ACK : 06

SEQ : 02

### 2.10.6 ClockSyncNETS

Send to terminal:

STX : 02

SEQ : 05  
TAG : 67 - PTAG\_ADMIN Container  
TAGlen : DLE 03  
TAG : 88 - PTAG\_COMMAND  
TAGlen : 01  
TAGvalue : 09 - ADMIN\_CLOCKSYNCSyncNets '.'

ETX : 03  
CRC : ac58

Send acknowledge to the terminal:

ACK : 06  
SEQ : 05

Terminal response INFO line 1 SYNKRONISER UR Nets:

STX : 02  
SEQ : 05  
TAG : 60 - PTAG\_INFO Container  
TAGlen : 23  
TAG : 00 - PTAG\_TSI  
TAGlen : 12  
TAGvalue : 53 59 4e 4b 52 4f 4e 49 53 45 52 20 55 52 20 50 'SYNKRONISER UR P'  
TAGvalue : 42 53 'BS'  
  
TAG : 80 - PTAG\_BINARY  
TAGlen : 01  
TAGvalue : 01 '.'  
  
TAG : 89 - PTAG\_SEQNO  
TAGlen : 01  
TAGvalue : 05 '.'  
  
TAG : 85 - PTAG\_FROMSTATE  
TAGlen : 01  
TAGvalue : 2b - PSTATE\_ADMIN '+'  
  
TAG : 87 - PTAG\_EVENT  
TAGlen : 01  
TAGvalue : 28 - PEVENT\_ECR\_INFO '('  
  
TAG : 86 - PTAG\_STATE  
TAGlen : 01  
TAGvalue : 2b - PSTATE\_ADMIN '+'

ETX : 03  
CRC : 1c77

Send acknowledge to the terminal:

ACK : 06  
SEQ : 05

Terminal response INFO line 1 VENT (ARBEJDER):

STX : 02  
SEQ : 06  
TAG : 60 - PTAG\_INFO Container  
TAGlen : 20  
TAG : 00 - PTAG\_TSI  
TAGlen : 0f  
TAGvalue : 56 45 4e 54 20 28 41 52 42 45 4a 44 45 52 29 'VENT (ARBEJDER)'  
  
TAG : 80 - PTAG\_BINARY  
TAGlen : 01  
TAGvalue : 01 ', '  
  
TAG : 89 - PTAG\_SEQNO  
TAGlen : 01  
TAGvalue : 05 ', '  
  
TAG : 85 - PTAG\_FROMSTATE  
TAGlen : 01  
TAGvalue : 2b - PSTATE\_ADMIN '+ '  
  
TAG : 87 - PTAG\_EVENT  
TAGlen : 01  
TAGvalue : 28 - PEVENT\_ECR\_INFO ' ('  
  
TAG : 86 - PTAG\_STATE  
TAGlen : 01  
TAGvalue : 2b - PSTATE\_ADMIN '+ '

ETX : 03  
CRC : bbcc

Send acknowledge to the terminal:

ACK : 06  
SEQ : 06

Terminal response INFO line 2 OPKALD...

```

STX : 02
SEQ : 07
  TAG      : 60 - PTAG_INFO Container
  TAGlen   : 1a
    TAG : 82 - PTAG_TEXT
    TAGlen : 09
    TAGvalue : 4f 50 4b 41 4c 44 2e 2e 2e      'OPKALD...'

    TAG : 80 - PTAG_BINARY
    TAGlen : 01
    TAGvalue : DLE 02                          ',.'

    TAG : 89 - PTAG_SEQNO
    TAGlen : 01
    TAGvalue : 05                              ',.'

    TAG : 85 - PTAG_FROMSTATE
    TAGlen : 01
    TAGvalue : 2b - PSTATE_ADMIN                '+ '

    TAG : 87 - PTAG_EVENT
    TAGlen : 01
    TAGvalue : 28 - PEVENT_ECR_INFO             '( '

    TAG : 86 - PTAG_STATE
    TAGlen : 01
    TAGvalue : 2b - PSTATE_ADMIN                '+ '

ETX : 03
CRC : 48a7

```

Send acknowledge to the terminal:

```

ACK : 06
SEQ : 07

```

Terminal response INFO line 2 SENDER...

```

STX : 02
SEQ : 08
  TAG      : 60 - PTAG_INFO Container
  TAGlen   : 1a
    TAG : 82 - PTAG_TEXT
    TAGlen : 09
    TAGvalue : 53 45 4e 44 45 52 2e 2e 2e 'SENDER...'

```

```

TAG      : 80 - PTAG_BINARY
TAGlen   : 01
TAGvalue : DLE 02 '.,'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 05 '.,'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 2b - PSTATE_ADMIN '+ '

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 28 - PEVENT_ECR_INFO '( '

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 2b - PSTATE_ADMIN '+ '

```

```

ETX : 03
CRC : 2be4

```

Send acknowledge to the terminal:

```

ACK : 06
SEQ : 08

```

Terminal response INFO line 2 MODTAGER...

```

STX : 02
SEQ : 09
TAG      : 60 - PTAG_INFO Container
TAGlen   : 1c
TAG      : 82 - PTAG_TEXT
TAGlen   : 0b
TAGvalue : 4d 4f 44 54 41 47 45 52 2e 2e 2e 'MODTAGER...'

TAG      : 80 - PTAG_BINARY
TAGlen   : 01
TAGvalue : DLE 02 '.,'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 05 '.,'

```

```

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 2b - PSTATE_ADMIN      '+'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 28 - PEVENT_ECR_INFO  '('

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 2b - PSTATE_ADMIN      '+'

```

```

ETX : 03
CRC : 41a8

```

Send acknowledge to the terminal:

```

ACK : 06
SEQ : 09

```

Terminal response INFO line 2 GODKENDT:

```

STX : 02
SEQ : 0a
TAG      : 60 - PTAG_INFO Container
TAGlen   : 19
TAG      : 82 - PTAG_TEXT
TAGlen   : 08
TAGvalue : 47 4f 44 4b 45 4e 44 54      'GODKENDT'

TAG      : 80 - PTAG_BINARY
TAGlen   : 01
TAGvalue : DLE 02                        ', '

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 05                          ', '

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 2b - PSTATE_ADMIN      '+'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 28 - PEVENT_ECR_INFO  '('

```

TAG : 86 - PTAG\_STATE  
TAGlen : 01  
TAGvalue : 2b - PSTATE\_ADMIN '+'

ETX : 03  
CRC : 647c

Send acknowledge to the terminal:

ACK : 06  
SEQ : 0A

Terminal response INFO line 2 AFBRYDER:

STX : 02  
SEQ : 0b  
TAG : 60 - PTAG\_INFO Container  
TAGlen : 19  
TAG : 82 - PTAG\_TEXT  
TAGlen : 08  
TAGvalue : 41 46 42 52 59 44 45 52 'AFBRYDER'  
  
TAG : 80 - PTAG\_BINARY  
TAGlen : 01  
TAGvalue : DLE 02 '.'  
  
TAG : 89 - PTAG\_SEQNO  
TAGlen : 01  
TAGvalue : 05 '.'  
  
TAG : 85 - PTAG\_FROMSTATE  
TAGlen : 01  
TAGvalue : 2b - PSTATE\_ADMIN '+'  
  
TAG : 87 - PTAG\_EVENT  
TAGlen : 01  
TAGvalue : 28 - PEVENT\_ECR\_INFO '('  
  
TAG : 86 - PTAG\_STATE  
TAGlen : 01  
TAGvalue : 2b - PSTATE\_ADMIN '+'

ETX : 03  
CRC : 59aa



Send acknowledge to terminal:

ACK : 06

SEQ : 0B

Terminal response to ADMIN command:

STX : 02

SEQ : 0c

```
    TAG      : 67 - PTAG_ADMIN Container
    TAGlen    : 0f
    TAG      : 84 - PTAG_RESULT
    TAGlen    : 01
    TAGvalue  : 00 - OK                      ',.'
```

```
    TAG      : 89 - PTAG_SEQNO
    TAGlen    : 01
    TAGvalue  : 05                      ',.'
```

```
    TAG      : 85 - PTAG_FROMSTATE
    TAGlen    : 01
    TAGvalue  : 2b - PSTATE_ADMIN          '+'
```

```
    TAG      : 87 - PTAG_EVENT
    TAGlen    : 01
    TAGvalue  : 1e - PEVENT_ECR_TRANSACTION_COMPLETED ',.'
```

```
    TAG      : 86 - PTAG_STATE
    TAGlen    : 01
    TAGvalue  : 07 - PSTATE_OPEN          ',.'
```

ETX : 03

CRC : 180c

Send acknowledge to terminal:

ACK : 06

SEQ : 0C

## 2.10.7 Extended Issuer Envelope (EIE)

The LPP ECR can transfer Extended Issuer Envelope (EIE) data along with the PTAG\_DATA container response (PTAG\_AMOUNT) to CARDNUMBER using the new PTAG\_EIE\_DATA 0x51. Data is stored and send to the PSAM before Payment command.

After Payment command, EIE data read from the PSAM is transmitted back to the ECR in a PTAG\_DATA container, using the PTAG\_EIE\_DATA.

The ECR must respond with PTAG\_DATA container with a PTAG\_EIE\_DATA, to be send to the PSAM before Complete Payment.

```
typedef struct
{
    byte resp_mode; // 0x00 Empty not used allow memset 0x00 to be used
                    // 0x01 Request Append EIE to PSAM (PSAM start with empty EIE)
                    // 0x02 Advice Clear EIE in PSAM before update with this
                    // 0x03 Advice Append to EIE hold by PSAM
                    // 0x04 Advice Clear EIE in PSAM only
                    // 0x05 Request Clear EIE in PSAM only
                    // 0x06 Response from Host to the Request EIE
    int length_ie; // Length of SWE additional data IE part of data 0-90
    int length_eie; // Length of EIE part of data
    byte data[0..512]; // 0..length_ie+length_eie
} PACKED EIE_DATA_BLOCK; // Extended Issuer Envelope Data Block used by PTAG_EIE_DATA
```

Observe: length\_ie and length\_eie is stored as little-endian. TAGlength in data as big-endian.

#### 2.10.7.1 SWE additional data IE part of data may contain:

Only TAGs allowed here are “Z7”, “Z8”, “Z2”, “Z3” and/or “Z4”.

See OTRS 3.3.0 Table 1-15.8 “Swedish additional transaction data in Issuer Information Envelope” data is added to the TAG “TZ” build by the terminal.

“TZ” build by terminal uses 18 bytes, “Z7”, “Z8”, “Z2”, “Z3” and/or “Z4” up to 74 bytes, missing tags or tags with zero length is omitted. Give a max of 92 bytes.

#### 2.10.7.2 DCC Transaction Information

DCC is setup by the terminal and uses 42 bytes, see OTRS 3.3.0 Table 1-10.12 “DCC data element in DCC-Transaction-Information”.

#### 2.10.7.3 Example:

ECR get the PTAG\_DATA with the PTAG\_CARDNUMBER.

```
TimeStamp: 08:10:09
STX       : 02
SEQ       : b8
TAG       : 63 - PTAG_DATA Container
TAGlen    : 26
TAG       : 6f - PTAG_CARDDATA Container
TAGlen    : 18
TAG       : 52 - PTAG_CARDNUMBER
TAGlen    : DLE 10
TAGvalue  : 34 35 37 31 38 31 41 41 41 41
           : 41 41 30 34 36 36 '457181AAAAAA0466'
TAG       : 5a - PTAG_CRCNUMBER
```

```

TAGlen      : 04
TAGvalue    : 00 00 DLE 03 e7          '....ζ'

TAG         : 89 - PTAG_SEQNO
TAGlen      : 01
TAGvalue    : 0a                      '.'

```

```

TimeStamp: 08:10:09
ACK       : 06
SEQ       : b8

```

ECR response with PTAG\_DATA container with PTAG\_CARDDATA container, PTAG\_AMOUNT and PTAG\_EIE\_DATA telling the host "Customer Reference Number" is 1234.

```

TimeStamp: 08:10:09
STX       : 02
SEQ       : 0b
TAG       : 63 - PTAG_DATA Container
TAGlen    : 1a
TAG       : 6f - PTAG_CARDDATA Container
TAGlen    : DLE 03
TAG       : 84 - PTAG_RESULT
TAGlen    : 01
TAGvalue  : 00 - OK                      '.'

```

```

TAG      : 51 - PTAG_EIE_DATA
TAGlen   : 11
TAGvalue : 01 00 00 00 00 08 00 00 00 34
          4E 00 04 31 32 32 '.....4N..123'
          resp_mode length_ie length_eie data
          34 '4'

```

```

ETX      : 03
CRC      : CRCCRC

```

```

TimeStamp: 08:10:09
ACK      : 06
SEQ      : 0b

```

The host response say "Receipt Number" 5678 is going to be used by the host for this transaction. (PSAM doesn't support response at this moment, and this is just an example using a known tag).

```

TimeStamp: 08:10:09
STX      : 02
SEQ      : b9
TAG      : 63 - PTAG_DATA Container
TAGlen   : 0f

```

```

TAG      : 51 - PTAG_EIE_DATA
TAGlen   : 11
TAGvalue : 06 00 00 00 00 08 00 00 00 34 4F
          00 04 35 36 37 '.....40..567'
          resp_mode length_ie length_eie data
          38 '4'

```

```

ETX      : 03
CRC      : CRCCRC

```

```

TimeStamp: 08:10:09
ACK      : 06
SEQ      : b9

```

ECR response with PTAG\_DATA container with PTAG\_EIE\_DATA.  
Containing "Customer Reference Number" 1234 and "Receipt Number" 5678 to be used with the "Financial Advice".

```

TimeStamp: 08:10:09
STX      : 02
SEQ      : 0c

```

```

TAG      : 63 - PTAG_DATA Container
TAGlen   : 17

TAG      : 51 - PTAG_EIE_DATA
TAGlen   : 18
TAGvalue : 02 00 00 00 00 10 00 00 00 00 34 4E
          00 04 31 32 32 '.....4N..123'
resp_mode length data..
43 34 4F 00 04 35 36 37 38 '40..5678'
data continue..
ETX      : 03
CRC      : CRCCRC

TimeStamp: 08:10:09
ACK      : 06
SEQ      : 0c

```

#### 2.10.7.4 On terminal boot we get from the PSAM with Identifier 0x0011

```

CARD Build Nov 19 2011 22:12:52 (c) 2004-11 Point Transaction Systems A/S
USER Build Nov 19 2011 22:13:24 (c) 2004-07 Point Transaction Systems A/S
POTRS_TERMINAL_SETTINGS - rc = 0 PSAMvers=71.8 rlen=8
Issuer Envelope, non-EMV          150
Issuer Envelope, EMV              150
Total Issuer and Extended Issuer Envelope, non-EMV  340
Total Issuer and Extended Issuer Envelope, EMV       160

```

```

DCC TRANSACTION INFO length 42 bytes (42 bytes used by terminal)
SW ADDITIONAL TRANS INFO len 92 bytes (18 bytes used by terminal)
Total 134 bytes (60 bytes used by terminal if DCC & SWE)

```

#### 2.10.7.5 The total envelope data is send to the ECR as part of OPEN parameters

```

EIE          128
PSAM_EIE     128
Issuer Envelope,non-EMV  150
Issuer Envelope, EMV     150
Total Issuer and Extended Issuer Envelope, non-EMV  340
Total Issuer and Extended Issuer Envelope, EMV       160

```

```

TimeStamp: 10:49:56
STX      : 02
SEQ      : 00
TAG      : 69 - PTAG_INIT Container

```

```

TAGlen : DLE 02
TAG : 44 - PTAG_OPEN
TAGlen : 00

ETX : 03
CRC : c956

TimeStamp: 10:49:56
ACK : 06
SEQ : 00

TimeStamp: 10:49:56
STX : 02
SEQ : 6a
TAG : 69 - PTAG_INIT Container
TAGlen : 81 fa
TAG : 84 - PTAG_RESULT
TAGlen : 01
TAGvalue : 00 - OK

TAG : 80 - PTAG_BINARY
TAGlen : DLE 02
TAGvalue : 00 00

TAG : 82 - PTAG_TEXT
TAGlen : 81 de
TAGvalue : 39 39 30 35 35 35 2c 31 33 2c 30 2c 30 2c 30 2c '90555,13,0,0,0,'
TAGvalue : 31 2c 30 2c 31 2c 30 2c 30 37 36 30 31 31 30 30 '1,0,1,0,07601100'
TAGvalue : 30 31 2c 50 42 53 20 50 53 41 4d 2d 31 34 36 20 '01,PBS PSAM-146 '
TAGvalue : 20 20 c6 d8 c5 2c 42 41 4c 4c 45 52 55 50 20 20 ' EØÅ,BALLERUP '
TAGvalue : 20 20 20 20 20 20 2c 4c 41 55 54 52 55 50 42 4a ' ,LAUTRUPBJ'
TAGvalue : 45 52 47 20 31 30 20 20 20 20 20 20 20 20 2c 'ERG 10 , '
TAGvalue : 44 4b 2d 32 37 35 30 20 2c 28 2b 34 35 29 20 34 'DK-2750 ,( +45) 4'
TAGvalue : 34 20 36 38 20 34 34 20 36 38 20 20 20 20 20 20 '4 68 44 68 '
TAGvalue : 20 2c 31 32 33 34 35 36 37 38 20 20 20 20 2c 34 ' ,12345678 ,4'
TAGvalue : 2c 20 64 61 2c 30 2c 32 35 35 2c 32 30 38 2c 32 ' , da,0,255,208,2'
TAGvalue : 30 38 2c 30 2c 33 34 2c 20 20 20 20 33 2e 30 30 '08,0,34, 3.00'
TAGvalue : 30 30 2c 30 2c 33 2e 33 2e 30 31 2c 30 2c 30 2c '00,0,3.3.01,0,0,'
TAGvalue : 58 45 4e 54 41 2c 31 32 38 2c 31 32 38 2c 31 35 'XENTA,128,128,15'
TAGvalue : 30 2c 31 35 30 2c 33 34 30 2c 31 36 30 00 '0,150,340,160.'

TAG : 90 - PTAG_ECREXTENDEDFUNCTIONS
TAGlen : 04
TAGvalue : 00 00 00 04

TAG : 89 - PTAG_SEQNO
TAGlen : 01
TAGvalue : 00

TAG : 85 - PTAG_FROMSTATE
TAGlen : 01
TAGvalue : 05 - PSTATE_CONNECTED

TAG : 87 - PTAG_EVENT

```

TAGlen : 01  
TAGvalue : 1a - PEVENT\_ECR\_OPEN , ,

TAG : 86 - PTAG\_STATE  
TAGlen : 01  
TAGvalue : 07 - PSTATE\_OPEN , ,

ETX : 03  
CRC : 0293

TimeStamp: 10:49:56  
ACK : 06  
SEQ : 6a

#### **2.10.7.6 Example DLL kasse demo - no longer supported:**

Printer selection: Only on display  
IP address 10.0.0.157 and IP Port 2000  
applying flxSetConfiguration(1, 2, 10.0.0.157, 2000)  
flxConnect(0x04): CONNECT\_OK  
flxOpen(): CONNECT\_OK

1:	TERMINALID	990555
2:	RECEIPTTYPE	13
3:	GRATUITYMODEL	0
4:	DCC	0
5:	TOKEN	0
6:	PREPAID	1
7:	KEYENTRY	XENTA
8:	OFFLINE	1
9:	IP ROUTING	0
10:	MERCHANTNUMBER	0760110001
11:	MERCHANTNAME	PBS PSAM-146 ÆØÅ
12:	MERCHANTCITY	BALLERUP
13:	MERCHANTADDRESS	LAUTRUPBJERG 10
14:	MERCHANTZIP	DK-2750
15:	MERCHANTPHONE	(+45) 44 68 44 68
16:	MERCHANTBRN	12345678
17:	FLXTRACELEVEL	4
18:	LANGUAGE	da
19:	VAT	0
20:	USERINPUT	255
21:	COUNTRYCODE	208
22:	DEFAULTCURRENCY	208
23:	CONTACTLESSFLAGS	0
24:	TERMINALTYPE	34
25:	DCCMARKUP	3.0000
26:	TCS	0
27:	SOFTWARE_VERSION	3.3.01
28:	CDP	12
29:	PSAM_CDP	4
30:	HARWARENAME	XENTA
31:	EIE	128
32:	PSAM_EIE	128
33:	ISSUER_ENVELOPE_NON_EMV_SIZE	150
34:	ISSUER_ENVELOPE_EMV_SIZE	150
35:	TOTAL_ISSUER_ENVELOPE_NON_EMV_SIZE	340
36:	TOTAL_ISSUER_ENVELOPE_EMV_SIZE	160

MI = MI\_PIN and transaction type = FLX\_TRANS\_PURCHASE

StartTid 2011-12-05 11:01.41

Status: AFVENTER KORT line: 1

Status: INDLÆS KORT line: 2

Status: AFVENTER VALIDERING line: 1



SET CARD DATA Cardnumber:457199AAAAAA9968,1, CardType:PBS\_CARD  
FLX\_CARDNUMBER\_CMD: 'FLX\_CARDNUMBER\_OK'

#### 2.10.7.7 EIEdata2host

Response mode 1  
Length Issuer Envelope 7  
Length Extended Issuer Envelope 8  
Issuer Envelope Data  
HexDump Size=7 - 0x0007  
HexCnt Dec 0 1 2 3 4 5 6 7 - 8 9 A B C D E F  
0000 - 000000: 34 4e 00 03 49 45 20 4N..IE  
Extended Issuer Envelope Data  
HexDump Size=8 - 0x0008  
HexCnt Dec 0 1 2 3 4 5 6 7 - 8 9 A B C D E F  
0000 - 000000: 34 4f 00 04 45 49 45 20 40..EIE  
  
GetAmountFee: 0  
Status: AFVENTER PIN/BELØB line: 1  
Status: VENT (ARBEJDER) line: 1  
Status: OPKALD... line: 2  
Early STAN PAN: 457199EEEEEE9968;000088;111205  
Status: SENDER... line: 2  
Status: MODTAGER... line: 2

#### 2.10.7.8 EIEdataReceived

Response mode 6  
Length Issuer Envelope 0  
Length Extended Issuer Envelope 0  
Issuer Envelope Data  
HexDump Size=0 - 0x0000  
HexCnt Dec 0 1 2 3 4 5 6 7 - 8 9 A B C D E F  
Extended Issuer Envelope Data  
HexDump Size=0 - 0x0000  
HexCnt Dec 0 1 2 3 4 5 6 7 - 8 9 A B C D E F

#### 2.10.7.9 EIEdata2host

Response mode 1  
Length Issuer Envelope 7

```

Length Extended Issuer Envelope      8
Issuer Envelope Data
HexDump Size=7 - 0x0007
HexCnt  Dec    0  1  2  3  4  5  6  7 -  8  9  A  B  C  D  E  F
0000 - 000000: 34 4e 00 03 49 45 20                                4N..IE
Extended Issuer Envelope Data
HexDump Size=8 - 0x0008
HexCnt  Dec    0  1  2  3  4  5  6  7 -  8  9  A  B  C  D  E  F
0000 - 000000: 34 4f 00 04 45 49 45 20                                4O..EIE

```

```

Status: GODKENDT line: 2
Status: KVITTERING UDSKRIVES line: 4

```

```

PBS PSAM-146 ÆØÅ
LAUTRUPBJERG 10
DK-2750 BALLERUP

```

```

TLF: (+45) 44 68 44 68

```

```

2011-12-05      10:58

```

```

KØB      DKK      35,00

```

```

-----
PIN KØB
VisaDankort      PSN: 00
AAAA AAAA AAAA 9968
TERM:      00990555-000088
IA1      PBS NR:0760110001
ATC:03601      AED:000000
AID:      A0000000031010
PSAM: 5374978-0000000540
ARC:00      STATUS:0000
AUT. KODE:      10:40K
REF:000088      AUTORISERET

```

```

PrintReceipt
receipt completed

```

```

BINARY RECEIPT:

```

```

2011-12-05 09:58;

```

457199EEEEEE9968;3500;0;0;0;208;88;2129592318;540;0000;0;11;10:40K;VisaDankort  
;00990555;760110001;PBS PSAM-146 EØÅ;BALLERUP ;LAUTRUPBJERG 10  
;DK-2750 ;(+45) 44 68 44 68 ;12345678 ;123456;208;1.000000;1;550030  
;1;0;;

Receipt:

Time (yyyy-mm-dd hh:mm) : 2011-12-05 09:58  
PAN : 457199EEEEEE9968  
total : 3500  
extra : 0  
fee : 0  
gratuity : 0  
currency : 208  
Stan : 88  
PSAM\_Creator : 2129592318  
PSAM\_ID : 540  
Action Code : 0000  
Asw1Asw2 : 0  
CvmStatus : 11  
Authorization Code : 10:40K  
Cardname : VisaDankort  
Terminal Ident : 00990555  
PBS Number : 760110001  
Name : PBS PSAM-146 EØÅ  
City : BALLERUP  
Address : LAUTRUPBJERG 10  
Zip : DK-2750  
Phone : (+45) 44 68 44 68  
CVR : 12345678  
refNr : 123456  
Dcc Currrency : 208  
Dcc Rate : 1.000000  
Card CRC : 1  
Batch Number : 550030  
Cancallation Allowed : 1  
Vat : 0  
Not used yet 1 :  
Not used yet 2 :  
flxCardTransaction result=SUCCESS  
flxCardTransaction error=0

StopTid 2011-12-05 11:02.08

### 2.10.7.10 Example LPP – from DLL kasse demo - no longer supported

```
TimeStamp: 11:49:22
STX      : 02
SEQ      : ff
TAG      : 69 - PTAG_INIT Container
TAGlen   : DLE 10
TAG      : 40 - PTAG_CONNECT
TAGlen   : 04
TAGvalue : DLE 03 04 00 00          '....'

TAG      : 80 - PTAG_BINARY
TAGlen   : DLE 02
TAGvalue : 00 DLE 02                '...'

TAG      : 90 - PTAG_ECREXTENDEDFUNCTIONS
TAGlen   : 04
TAGvalue : 00 00 00 04              '....'
```

```
ETX      : 03
CRC      : d451
```

```
TimeStamp: 11:49:22
ACK      : 06
SEQ      : ff
```

```
TimeStamp: 11:49:22
STX      : 02
SEQ      : 01
TAG      : 69 - PTAG_INIT Container
TAGlen   : 23
TAG      : 84 - PTAG_RESULT
TAGlen   : 01
TAGvalue : 00 - OK                  ', '

TAG      : 80 - PTAG_BINARY
TAGlen   : DLE 02
TAGvalue : 00 01                    '...'

TAG      : 82 - PTAG_TEXT
TAGlen   : 08
TAGvalue : 39 39 30 35 35 35 00 00  '990555..'

TAG      : 90 - PTAG_ECREXTENDEDFUNCTIONS
TAGlen   : 04
TAGvalue : 00 00 00 04              '....'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : ff                        'ÿ'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
```

```

TAGvalue : DLE 03 - PSTATE_IDLE          '...'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 19 - PEVENT_ECR_CONNECT_OK    '...'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 05 - PSTATE_CONNECTED        '...'

ETX      : 03
CRC      : a9ae

TimeStamp: 11:49:22
ACK      : 06
SEQ      : 01

TimeStamp: 11:49:23
STX      : 02
SEQ      : 00

TAG      : 69 - PTAG_INIT Container
TAGlen   : DLE 02
TAG      : 44 - PTAG_OPEN
TAGlen   : 00                                ''

ETX      : 03
CRC      : c956

TimeStamp: 11:49:23
ACK      : 06
SEQ      : 00

TimeStamp: 11:49:24
STX      : 02
SEQ      : 02
TAG      : 69 - PTAG_INIT Container
TAGlen   : 81 fa
TAG      : 84 - PTAG_RESULT
TAGlen   : 01
TAGvalue : 00 - OK                          ', '

TAG      : 80 - PTAG_BINARY
TAGlen   : DLE 02
TAGvalue : 00 00                          '...'

TAG      : 82 - PTAG_TEXT
TAGlen   : 81 de
TAGvalue : 39 39 30 35 35 35 2c 31 33 2c 30 2c 30 2c 30 2c '990555,13,0,0,0,'
TAGvalue : 31 2c 30 2c 31 2c 30 2c 30 37 36 30 31 31 30 30 '1,0,1,0,07601100'
TAGvalue : 30 31 2c 50 42 53 20 50 53 41 4d 2d 31 34 36 20 '01,PBS PSAM-146 '
TAGvalue : 20 20 c6 d8 c5 2c 42 41 4c 4c 45 52 55 50 20 20 '  EØÅ,BALLERUP '
TAGvalue : 20 20 20 20 20 20 2c 4c 41 55 54 52 55 50 42 4a ' ,LAUTRUPBJ'
TAGvalue : 45 52 47 20 31 30 20 20 20 20 20 20 20 20 2c 'ERG 10      ,'
```

```

TAGvalue : 44 4b 2d 32 37 35 30 20 2c 28 2b 34 35 29 20 34 'DK-2750 ,(+45) 4'
TAGvalue : 34 20 36 38 20 34 34 20 36 38 20 20 20 20 20 20 '4 68 44 68      '
TAGvalue : 20 2c 31 32 33 34 35 36 37 38 20 20 20 20 2c 34 ' ,12345678      ,4'
TAGvalue : 2c 20 64 61 2c 30 2c 32 35 35 2c 32 30 38 2c 32 ' , da,0,255,208,2'
TAGvalue : 30 38 2c 30 2c 33 34 2c 20 20 20 20 33 2e 30 30 '08,0,34,      3.00'
TAGvalue : 30 30 2c 30 2c 33 2e 33 2e 30 31 2c 30 2c 30 2c '00,0,3.3.01,0,0,'
TAGvalue : 58 45 4e 54 41 2c 31 32 38 2c 31 32 38 2c 31 35 'XENTA,128,128,15'
TAGvalue : 30 2c 31 35 30 2c 33 34 30 2c 31 36 30 00      '0,150,340,160.'

TAG      : 90 - PTAG_ECREXTENDEDFUNCTIONS
TAGlen   : 04
TAGvalue : 00 00 00 04                                     ',...,'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 00                                              ',.'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 05 - PSTATE_CONNECTED                          ',.'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 1a - PEVENT_ECR_OPEN                           ',.'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 07 - PSTATE_OPEN                               ',.'

ETX      : 03
CRC      : 1edd

TimeStamp: 11:49:24
ACK      : 06
SEQ      : 02

TimeStamp: 11:49:46
STX      : 02
SEQ      : 01
TAG      : 65 - PTAG_TRANSACTION Container
TAGlen   : 28
TAG      : 4e - PTAG_MI
TAGlen   : 01
TAGvalue : 00                                              ',.'

TAG      : 4c - PTAG_CU
TAGlen   : DLE 02
TAGvalue : 00 d0                                           ',.D'

TAG      : 56 - PTAG_TT
TAGlen   : 01
TAGvalue : 00                                              ',.'

TAG      : 50 - PTAG_TR

```

```

TAGlen   : 01
TAGvalue : 00                                     ', '

TAG      : 4a - PTAG_REF_NO
TAGlen   : 04
TAGvalue : 00 01 e2 40                           '..â@'

TAG      : 59 - PTAG_GRATUITY
TAGlen   : 04
TAGvalue : 00 00 00 00                           '....'

TAG      : 57 - PTAG_VAT
TAGlen   : 04
TAGvalue : 00 00 00 00                           '....'

TAG      : 94 - PTAG_TERM_ENV
TAGlen   : 01
TAGvalue : 00                                     ', '

TAG      : 92 - PTAG_CARD_SOURCE
TAGlen   : 01
TAGvalue : 00                                     ', '

TAG      : 93 - PTAG_PREPAID
TAGlen   : 01
TAGvalue : 00                                     ', '

ETX      : 03
CRC      : 1392

TimeStamp: 11:49:46
ACK      : 06
SEQ      : 01

TimeStamp: 11:49:46
STX      : 02
SEQ      : 03
  TAG     : 60 - PTAG_INFO Container
  TAGlen  : 1e
    TAG   : 00 - PTAG_TSI
    TAGlen : 0d
    TAGvalue : 41 46 56 45 4e 54 45 52 20 4b 4f 52 54 'AFVENTER KORT'

    TAG   : 80 - PTAG_BINARY
    TAGlen : 01
    TAGvalue : 01                                     ', '

    TAG   : 89 - PTAG_SEQNO
    TAGlen : 01
    TAGvalue : 01                                     ', '

    TAG   : 85 - PTAG_FROMSTATE
    TAGlen : 01
    TAGvalue : 07 - PSTATE_OPEN                       ', '

```

```

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 1d - PEVENT_ECR_TRANSACTION      ', '

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 09 - PSTATE_TRANS                ', '
ETX      : 03
CRC      : c9a5

TimeStamp: 11:49:46
ACK      : 06
SEQ      : 03

TimeStamp: 11:49:48
STX      : 02
SEQ      : 04
TAG      : 60 - PTAG_INFO Container
TAGlen   : 1c
TAG      : 82 - PTAG_TEXT
TAGlen   : 0b
TAGvalue : 49 4e 44 4c c6 53 20 4b 4f 52 54      'INDLES KORT'

TAG      : 80 - PTAG_BINARY
TAGlen   : 01
TAGvalue : DLE 02                             ', .'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 01                                 ', '

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 09 - PSTATE_TRANS                 ', '

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 28 - PEVENT_ECR_INFO              '( '

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 09 - PSTATE_TRANS                 ', '

ETX      : 03
CRC      : ca26

TimeStamp: 11:49:48
ACK      : 06
SEQ      : 04

TimeStamp: 11:49:51
STX      : 02
SEQ      : 05

```



```

TAG      : 79 - PTAG_ENAI Container
TAGlen   : 0f
    TAG   : 84 - PTAG_RESULT
    TAGlen : 01
    TAGvalue : 5a - Data from pbsIPhandle received ok      'Z'

    TAG   : 89 - PTAG_SEQNO
    TAGlen : 01
    TAGvalue : 01                                          ',.'

    TAG   : 85 - PTAG_FROMSTATE
    TAGlen : 01
    TAGvalue : 09 - PSTATE_TRANS                          ',.'

    TAG   : 87 - PTAG_EVENT
    TAGlen : 01
    TAGvalue : 67 - PEVENT_MAX                            'g'

    TAG   : 86 - PTAG_STATE
    TAGlen : 01
    TAGvalue : 09 - PSTATE_TRANS                          ',.'

ETX      : 03
CRC      : f4be

TimeStamp: 11:49:51
ACK      : 06
SEQ      : 05
TimeStamp: 11:49:53
STX      : 02
SEQ      : 06
    TAG   : 60 - PTAG_INFO Container
    TAGlen : 24
        TAG   : 00 - PTAG_TSI
        TAGlen : DLE 13
        TAGvalue : 41 46 56 45 4e 54 45 52 20 56 41 4c 49 44 45 52 'AFVENTER VALIDER'
        TAGvalue : 49 4e 47                                         'ING'

    TAG   : 80 - PTAG_BINARY
    TAGlen : 01
    TAGvalue : 01                                          ',.'

    TAG   : 89 - PTAG_SEQNO
    TAGlen : 01
    TAGvalue : 01                                          ',.'

    TAG   : 85 - PTAG_FROMSTATE
    TAGlen : 01
    TAGvalue : 09 - PSTATE_TRANS                          ',.'

    TAG   : 87 - PTAG_EVENT
    TAGlen : 01
    TAGvalue : 28 - PEVENT_ECR_INFO                        '('

```

```

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 09 - PSTATE_TRANS      ', '

ETX      : 03
CRC      : d3eb

TimeStamp: 11:49:53
ACK      : 06
SEQ      : 06

TimeStamp: 11:49:56
STX      : 02
SEQ      : 07
TAG      : 63 - PTAG_DATA Container
TAGlen   : 26
TAG      : 6f - PTAG_CARDDATA Container
TAGlen   : 18
TAG      : 52 - PTAG_CARDNUMBER
TAGlen   : DLE 10
TAGvalue : 34 35 37 31 39 39 41 41 41 41 41 41 39 39 36 38
          '457199AAAAAA9968'

TAG      : 5a - PTAG_CRCNUMBER
TAGlen   : 04
TAGvalue : 00 00 00 01            ', ....'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 01                    ', '

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 09 - PSTATE_TRANS      ', '

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 32 - PEVENT_TAPA_GET_AMOUNT '2'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 0f - PSTATE_TRANS_REQ_AMOUNT ', '

ETX      : 03
CRC      : d3b5
TimeStamp: 11:49:56
ACK      : 06
SEQ      : 07

TimeStamp: 11:49:56
STX      : 02
SEQ      : 02
TAG      : 63 - PTAG_DATA Container
TAGlen   : 2f

```

```

TAG      : 6f - PTAG_CARDDATA Container
TAGlen   : DLE 03
TAG      : 84 - PTAG_RESULT
TAGlen   : 01
TAGvalue : 00 - OK                                ',.'

TAG      : 48 - PTAG_AMOUNT
TAGlen   : 04
TAGvalue : 00 00 17 df                            '...f'

TAG      : 4c - PTAG_CU
TAGlen   : DLE 02
TAGvalue : 00 d0                                  ',D'

TAG      : 51 - PTAG_EIE
TAGlen   : 1e
TAGvalue : 00 0a 00 00 00 0b 00 00 00 31 32 33 34 35 36 37 '.....1234567'
TAGvalue : 38 39 30 61 62 63 64 65 66 67 68 69 6a 6b      '890abcdefghijk'

ETX      : 03
CRC      : e220

TimeStamp: 11:49:56
ACK      : 06
SEQ      : 02

TimeStamp: 11:49:57
STX      : 02
SEQ      : 08
TAG      : 62 - PTAG_DATA_1 Container
TAGlen   : 18
TAG      : 58 - PTAG_FEE
TAGlen   : 04
TAGvalue : 00 00 00 00                            '....'

TAG      : 59 - PTAG_GRATUITY
TAGlen   : 04
TAGvalue : 00 00 00 00                            '....'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 00                                      ',.'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 09 - PSTATE_TRANS                       ',.'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 56 - PEVENT_ECR_FEE                     'V'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 09 - PSTATE_TRANS                       ',.'

```

ETX : 03  
CRC : 671f

TimeStamp: 11:49:57  
ACK : 06  
SEQ : 08

TimeStamp: 11:49:57  
STX : 02  
SEQ : 09  
TAG : 60 - PTAG\_INFO Container

TAGlen : 23

TAG : 00 - PTAG\_TSI

TAGlen : 12

TAGvalue : 41 46 56 45 4e 54 45 52 20 50 49 4e 2f 42 45 4c 'AFVENTER PIN/BEL'

TAGvalue : d8 42 'ØB'

TAG : 80 - PTAG\_BINARY

TAGlen : 01

TAGvalue : 01

','

TAG : 89 - PTAG\_SEQNO

TAGlen : 01

TAGvalue : 00

','

TAG : 85 - PTAG\_FROMSTATE

TAGlen : 01

TAGvalue : 09 - PSTATE\_TRANS

','

TAG : 87 - PTAG\_EVENT

TAGlen : 01

TAGvalue : 28 - PEVENT\_ECR\_INFO

('

TAG : 86 - PTAG\_STATE

TAGlen : 01

TAGvalue : 09 - PSTATE\_TRANS

','

ETX : 03  
CRC : 617d

TimeStamp: 11:49:57  
ACK : 06  
SEQ : 09

TimeStamp: 11:50:02  
STX : 02  
SEQ : 03  
ETX : 03  
CRC : 40f1

TimeStamp: 11:50:02  
ACK : 06  
SEQ : 03

```

TimeStamp: 11:50:03
STX       : 02
SEQ       : 0a
  TAG      : 60 - PTAG_INFO Container
  TAGlen   : 20
    TAG     : 00 - PTAG_TSI
    TAGlen  : 0f
    TAGvalue : 56 45 4e 54 20 28 41 52 42 45 4a 44 45 52 29 'VENT (ARBEJDER)'

    TAG      : 80 - PTAG_BINARY
    TAGlen   : 01
    TAGvalue : 01 ','

    TAG      : 89 - PTAG_SEQNO
    TAGlen   : 01
    TAGvalue : 00 ','

    TAG      : 85 - PTAG_FROMSTATE
    TAGlen   : 01
    TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT '...'

    TAG      : 87 - PTAG_EVENT
    TAGlen   : 01
    TAGvalue : 28 - PEVENT_ECR_INFO '(('

    TAG      : 86 - PTAG_STATE
    TAGlen   : 01
    TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT '...'

```

```

ETX       : 03
CRC       : bbeb

```

```

TimeStamp: 11:50:03
ACK       : 06
SEQ       : 0a

```

```

TimeStamp: 11:50:05
STX       : 02
SEQ       : 0b
  TAG      : 60 - PTAG_INFO Container
  TAGlen   : 30
    TAG     : 82 - PTAG_TEXT
    TAGlen  : 09
    TAGvalue : 4f 50 4b 41 4c 44 2e 2e 2e 'OPKALD...'

    TAG      : 52 - PTAG_CARDNUMBER
    TAGlen   : 08
    TAGvalue : 45 71 99 ee ee ee 99 68 'Eq.îîî.h'

    TAG      : DLE 10 - PTAG_STAN
    TAGlen   : DLE 03
    TAGvalue : 00 DLE 02 37 '...7'

```

```

TAG      : 5c - PTAG_TIME
TAGlen   : 05
TAGvalue : 11 12 19 11 50          '....P'

TAG      : 80 - PTAG_BINARY
TAGlen   : 01
TAGvalue : DLE 02                  '...'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 00                      ' .'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT '...'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 28 - PEVENT_ECR_INFO       '( '

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT '...'

ETX      : 03
CRC      : ba95

TimeStamp: 11:50:05
ACK      : 06
SEQ      : 0b

TimeStamp: 11:50:07
STX      : 02
SEQ      : 0c
TAG      : 60 - PTAG_INFO Container
TAGlen   : 1a
TAG      : 82 - PTAG_TEXT
TAGlen   : 09
TAGvalue : 53 45 4e 44 45 52 2e 2e 2e 'SENDER...'

TAG      : 80 - PTAG_BINARY
TAGlen   : 01
TAGvalue : DLE 02                  '...'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 00                      ' .'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT '...'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01

```

```

TAGvalue : 28 - PEVENT_ECR_INFO          '( '

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT  '...'

ETX      : 03
CRC      : 6a98

TimeStamp: 11:50:07
ACK      : 06
SEQ      : 0c

TimeStamp: 11:50:08
STX      : 02
SEQ      : 0d
TAG      : 60 - PTAG_INFO Container
TAGlen   : 1c
TAG      : 82 - PTAG_TEXT
TAGlen   : 0b
TAGvalue : 4d 4f 44 54 41 47 45 52 2e 2e 2e  'MODTAGER...'

TAG      : 80 - PTAG_BINARY
TAGlen   : 01
TAGvalue : DLE 02                          '...'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 00                             '...'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT  '...'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 28 - PEVENT_ECR_INFO          '( '

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT  '...'

ETX      : 03
CRC      : f0d7

TimeStamp: 11:50:08
ACK      : 06
SEQ      : 0d

TimeStamp: 11:50:14
STX      : 02
SEQ      : 04
ETX      : 03
CRC      : 42c1

```

TimeStamp: 11:50:14

ACK : 06

SEQ : 04

TimeStamp: 11:50:18

STX : 02

SEQ : 0e

TAG : 63 - PTAG\_DATA Container

TAGlen : 4d

TAG : 51 - PTAG\_EIE

TAGlen : 3f

TAGvalue : DLE 06 00 00 00 00 36 00 00 00 11 22 33 44 55 66 '.....6...."3DUf'

TAGvalue : 77 88 99 11 22 33 44 55 66 77 88 99 11 22 33 44 'w..."3DUfw..."3D'

TAGvalue : 55 66 77 88 99 11 22 33 44 55 66 77 88 99 11 22 'Ufw..."3DUfw..."'

TAGvalue : 33 44 55 66 77 88 99 11 22 33 44 55 66 77 88 99 '3DUfw..."3DUfw..''

TAG : 89 - PTAG\_SEQNO

TAGlen : 01

TAGvalue : 00 '.,'

TAG : 85 - PTAG\_FROMSTATE

TAGlen : 01

TAGvalue : DLE 13 - PSTATE\_TRANS\_COMMIT '.,'

TAG : 87 - PTAG\_EVENT

TAGlen : 01

TAGvalue : 6a - Unknown 'j'

TAG : 86 - PTAG\_STATE

TAGlen : 01

TAGvalue : DLE 13 - PSTATE\_TRANS\_COMMIT '.,'

ETX : 03

CRC : 9958

TimeStamp: 11:50:18

ACK : 06

SEQ : 0e

TimeStamp: 11:50:18

STX : 02

SEQ : 05

TAG : 63 - PTAG\_DATA Container

TAGlen : 20

TAG : 51 - PTAG\_EIE

TAGlen : 1e

TAGvalue : 00 0a 00 00 00 0b 00 00 00 31 32 33 34 35 36 37 '.....1234567'

TAGvalue : 38 39 30 61 62 63 64 65 66 67 68 69 6a 6b '890abcdefghijk'

ETX : 03

CRC : b751

TimeStamp: 11:50:18



```

STX      : 02
SEQ      : 0f
  TAG     : 60 - PTAG_INFO Container
  TAGlen  : 0f
    TAG    : 84 - PTAG_RESULT
    TAGlen  : 01
    TAGvalue : 00 - OK                      ' .'

    TAG     : 89 - PTAG_SEQNO
    TAGlen  : 01
    TAGvalue : 00                          ' .'

    TAG     : 85 - PTAG_FROMSTATE
    TAGlen  : 01
    TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT '...'

    TAG     : 87 - PTAG_EVENT
    TAGlen  : 01
    TAGvalue : 45 - PEVENT_SIG_PRERESULT    'E'

    TAG     : 86 - PTAG_STATE
    TAGlen  : 01
    TAGvalue : DLE 13 - PSTATE_TRANS_COMMIT '...'

```

```

ETX      : 03
CRC      : xxxx

```

```

TimeStamp: 11:50:18
ACK      : 06
SEQ      : 0f

```

```

TimeStamp: 11:50:18
ACK      : 06
SEQ      : 05

```

```

TimeStamp: 11:50:19

```

```

STX      : 02
SEQ      : 10
  TAG     : 60 - PTAG_INFO Container
  TAGlen  : 19
    TAG    : 82 - PTAG_TEXT
    TAGlen  : 08
    TAGvalue : 47 4f 44 4b 45 4e 44 54      'GODKENDT'

    TAG     : 80 - PTAG_BINARY
    TAGlen  : 01
    TAGvalue : DLE 02                      '...'

    TAG     : 89 - PTAG_SEQNO
    TAGlen  : 01
    TAGvalue : 00                          ' .'

    TAG     : 85 - PTAG_FROMSTATE

```

```

TAGlen   : 01
TAGvalue : 42 - PSTATE_TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK      'B'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 28 - PEVENT_ECR_INFO                                  '( '

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 42 - PSTATE_TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK      'B'

ETX      : 03
CRC      : 8d02

TimeStamp: 11:50:19
ACK      : 06
SEQ      : 10

TimeStamp: 11:50:19
STX      : 02
SEQ      : 11
TAG      : 60 - PTAG_INFO Container
TAGlen   : 25
TAG      : 82 - PTAG_TEXT
TAGlen   : 14
TAGvalue : 4b 56 49 54 54 45 52 49 4e 47 20 55 44 53 4b 52 'KVITTERING UDSKR'
TAGvalue : 49 56 45 53                                       'IVES'

TAG      : 80 - PTAG_BINARY
TAGlen   : 01
TAGvalue : 04                                                ', '

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 00                                                ', '

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 42 - PSTATE_TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK      'B'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 28 - PEVENT_ECR_INFO                                  '( '

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 42 - PSTATE_TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK      'B'

ETX      : 03
CRC      : 20be

TimeStamp: 11:50:19
ACK      : 06
SEQ      : 11

```

TimeStamp: 11:50:21

STX : 02

SEQ : 12

TAG : 65 - PTAG\_TRANSACTION Container

TAGlen : 82 DLE 02 f1

TAG : 6b - PTAG\_RECEIPT Container

TAGlen : 82 DLE 02 de

TAG : 82 - PTAG\_TEXT

TAGlen : 82 01 9f

TAGvalue : 0a 0a 50 42 53 20 50 53 41 4d 2d 31 34 36 20 c6 '...PBS PSAM-146 E'  
TAGvalue : d8 c5 0a 4c 41 55 54 52 55 50 42 4a 45 52 47 20 'ØÅ.LAUTRUPBJERG '  
TAGvalue : 31 30 0a 44 4b 2d 32 37 35 30 20 42 41 4c 4c 45 '10.DK-2750 BALLE'  
TAGvalue : 52 55 50 0a 54 4c 46 3a 20 28 2b 34 35 29 20 34 'RUP.TLF: (+45) 4'  
TAGvalue : 34 20 36 38 20 34 34 20 36 38 20 20 0a 0a 0a 32 '4 68 44 68 ...2'  
TAGvalue : 30 31 31 2d 31 32 2d 31 39 20 20 20 20 20 20 20 '011-12-19 '  
TAGvalue : 20 20 31 31 3a 35 30 0a 0a 4b d8 42 20 20 20 20 ' 11:50..KØB '  
TAGvalue : 20 20 20 44 4b 4b 20 20 20 20 20 20 36 31 2c 31 ' DKK 61,1'  
TAGvalue : 31 0a 20 20 20 20 20 20 20 20 20 20 20 20 20 2d '1. -'  
TAGvalue : 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 0a 50 49 4e 20 4b '-----PIN K'  
TAGvalue : d8 42 0a 56 69 73 61 44 61 6e 6b 6f 72 74 20 20 'ØB.VisaDankort '  
TAGvalue : 20 20 20 20 50 53 4e 3a 20 30 30 0a 58 58 58 58 ' PSN: 00.XXXX'  
TAGvalue : 20 41 41 41 41 20 41 41 41 41 20 39 39 36 38 0a 'AAAA AAAA 9968.'  
TAGvalue : 54 45 52 4d 3a 20 20 20 20 30 30 39 39 30 35 35 'TERM: 0099055'  
TAGvalue : 35 2d 30 30 30 32 33 37 0a 49 41 31 20 20 20 20 '5-000237.IA1 '  
TAGvalue : 50 42 53 20 4e 52 3a 30 37 36 30 31 31 30 30 30 'PBS NR:076011000'  
TAGvalue : 31 0a 41 54 43 3a 30 33 37 34 30 20 20 20 20 20 '1.ATC:03740 '  
TAGvalue : 41 45 44 3a 30 30 30 30 30 30 0a 41 49 44 3a 20 'AED:000000.AID: '  
TAGvalue : 20 20 20 20 20 41 30 30 30 30 30 30 30 30 33 31 ' A0000000031'  
TAGvalue : 30 31 30 0a 50 53 41 4d 3a 20 35 33 37 34 39 37 '010.PSAM: 537497'  
TAGvalue : 38 2d 30 30 30 30 30 30 30 35 34 30 0a 41 52 43 '8-0000000540.ARC'  
TAGvalue : 3a 30 30 20 20 20 20 20 20 20 53 54 41 54 55 53 ':00 STATUS'  
TAGvalue : 3a 30 30 30 30 0a 41 55 54 2e 20 4b 4f 44 45 3a ':0000.AUT. KODE:'  
TAGvalue : 20 20 20 20 20 20 20 20 31 30 3a 34 4f 4b 0a 52 ' 10:40K.R'  
TAGvalue : 45 46 3a 30 30 30 32 33 37 20 20 20 41 55 54 4f 'EF:000237 AUTO'  
TAGvalue : 52 49 53 45 52 45 54 0a 0a 0a 0a 0a 0a 0a 0a 'RISERET.....'

TAG : 5c - PTAG\_TIME

TAGlen : 04

TAGvalue : 4e ef 16 f4 'Ni.ø'

TAG : 52 - PTAG\_CARDNUMBER

TAGlen : DLE 10

TAGvalue : 45 71 99 ee ee ee 99 68 00 00 00 00 00 00 00 'Eq.îîî.h.....'

TAG : 48 - PTAG\_AMOUNT

TAGlen : 04

TAGvalue : 00 00 17 df '...ß'

TAG : 58 - PTAG\_FEE

TAGlen : 04

TAGvalue : 00 00 00 00 '....'

TAG	:	1c - PTAG_EXTRA	
TAGlen	:	04	
TAGvalue	:	00 00 00 00	'....'
TAG	:	59 - PTAG_GRATUITY	
TAGlen	:	04	
TAGvalue	:	00 00 00 00	'....'
TAG	:	4c - PTAG_CU	
TAGlen	:	DLE 02	
TAGvalue	:	DLE 02 08	'...'
TAG	:	54 - PTAG_AID	
TAGlen	:	07	
TAGvalue	:	a0 00 00 00 DLE 03 DLE 10 DLE 10	' .....
TAG	:	0a - PTAG_ATC	
TAGlen	:	DLE 02	
TAGvalue	:	0e 9c	'..'
TAG	:	0c - PTAG_AED	
TAGlen	:	DLE 03	
TAGvalue	:	00 00 00	'...'
TAG	:	1e - PTAG_ARC	
TAGlen	:	DLE 02	
TAGvalue	:	30 30	'00'
TAG	:	DLE 13 - PTAG_ASW1ASW2	
TAGlen	:	DLE 02	
TAGvalue	:	00 00	'..'
TAG	:	DLE 10 - PTAG_STAN	
TAGlen	:	DLE 06	
TAGvalue	:	30 30 30 32 33 37	'000237'
TAG	:	12 - PTAG_PSAMID	
TAGlen	:	04	
TAGvalue	:	00 00 DLE 02 1c	'.....'
TAG	:	14 - PTAG_ACODE	
TAGlen	:	DLE 02	
TAGvalue	:	00 00	'..'
TAG	:	16 - PTAG_CVM	
TAGlen	:	01	
TAGvalue	:	0b	'.'
TAG	:	92 - PTAG_CARD_SOURCE	
TAGlen	:	01	
TAGvalue	:	00	'.'
TAG	:	18 - PTAG_AUTCODE	
TAGlen	:	DLE 06	

TAGvalue : 31 30 3a 34 4f 4b '10:40K'  
  
 TAG : 5e - PTAG\_CARDNAME  
 TAGlen : DLE 10  
 TAGvalue : 56 69 73 61 44 61 6e 6b 6f 72 74 20 20 20 20 20 'VisaDankort '  
  
 TAG : 1a - PTAG\_TERMID  
 TAGlen : 08  
 TAGvalue : 30 30 39 39 30 35 35 35 '00990555'  
  
 TAG : 0f - PTAG\_PSAM\_CREATOR  
 TAGlen : 04  
 TAGvalue : 81 11 00 DLE 02 '.....'  
  
 TAG : 95 - PTAG\_DCC\_RATE  
 TAGlen : 09  
 TAGvalue : 31 2e 30 30 30 30 30 30 00 '1.000000.'  
  
 TAG : 5a - PTAG\_CRCNUMBER  
 TAGlen : 04  
 TAGvalue : 00 00 00 01 '....'  
  
 TAG : 96 - PTAG\_CANCELLATION  
 TAGlen : 01  
 TAGvalue : 01 ',.'  
  
 TAG : 57 - PTAG\_VAT  
 TAGlen : 04  
 TAGvalue : 00 00 00 00 '....'  
  
 TAG : 97 - PTAG\_BATCHNUMBER  
 TAGlen : 0c  
 TAGvalue : 35 35 30 30 33 32 20 20 20 20 20 20 '550032 '  
  
 TAG : 98 - PTAG\_DCC\_CU  
 TAGlen : DLE 02  
 TAGvalue : DLE 02 08 '...'  
  
 TAG : 01 - PTAG\_MI\_NR  
 TAGlen : 05  
 TAGvalue : 07 60 11 00 01 ',...'  
  
 TAG : DLE 03 - PTAG\_MI\_NAME  
 TAGlen : 12  
 TAGvalue : 50 42 53 20 50 53 41 4d 2d 31 34 36 20 20 20 c6 'PBS PSAM-146 Æ'  
 TAGvalue : d8 c5 '0Å'  
  
 TAG : 05 - PTAG\_MI\_CITY  
 TAGlen : DLE 10  
 TAGvalue : 42 41 4c 4c 45 52 55 50 20 20 20 20 20 20 20 'BALLERUP '  
  
 TAG : 07 - PTAG\_MI\_ADDR  
 TAGlen : 18  
 TAGvalue : 4c 41 55 54 52 55 50 42 4a 45 52 47 20 31 30 20 'LAUTRUPBJERG 10 '

```

TAGvalue : 20 20 20 20 20 20 20 20 20      ,      ,

TAG      : 09 - PTAG_MI_ZIP
TAGlen   : 08
TAGvalue : 44 4b 2d 32 37 35 30 20          'DK-2750 '

TAG      : 0b - PTAG_MI_PHONE
TAGlen   : 18
TAGvalue : 28 2b 34 35 29 20 34 34 20 36 38 20 34 34 20 36 '(+45) 44 68 44 6'
TAGvalue : 38 20 20 20 20 20 20 20 20      '8      '

TAG      : 0d - PTAG_MI_BRN
TAGlen   : 0c
TAGvalue : 31 32 33 34 35 36 37 38 20 20 20 20      '12345678      '

TAG      : 4a - PTAG_REF_NO
TAGlen   : 04
TAGvalue : 00 01 e2 40                        '..â@'

TAG      : 84 - PTAG_RESULT
TAGlen   : 01
TAGvalue : 00 - OK                          ',.'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 00                              ',.'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : DLE 15 - PSTATE_PRINT_WAIT_ACK    '...'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 31 - PEVENT_TAPA_TRANSACTION_COMPLETE '1'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 17 - PSTATE_TRANS_WAIT_ACK        ',.'

ETX      : 03
CRC      : 22b7

TimeStamp: 11:50:22
ACK      : 06
SEQ      : 12

TimeStamp: 11:50:22
STX      : 02
SEQ      : 06
    TAG      : 6b - PTAG_RECEIPT Container
    TAGlen   : DLE 03
        TAG      : 84 - PTAG_RESULT
        TAGlen   : 01
        TAGvalue : 00 - OK                      ',.'

```

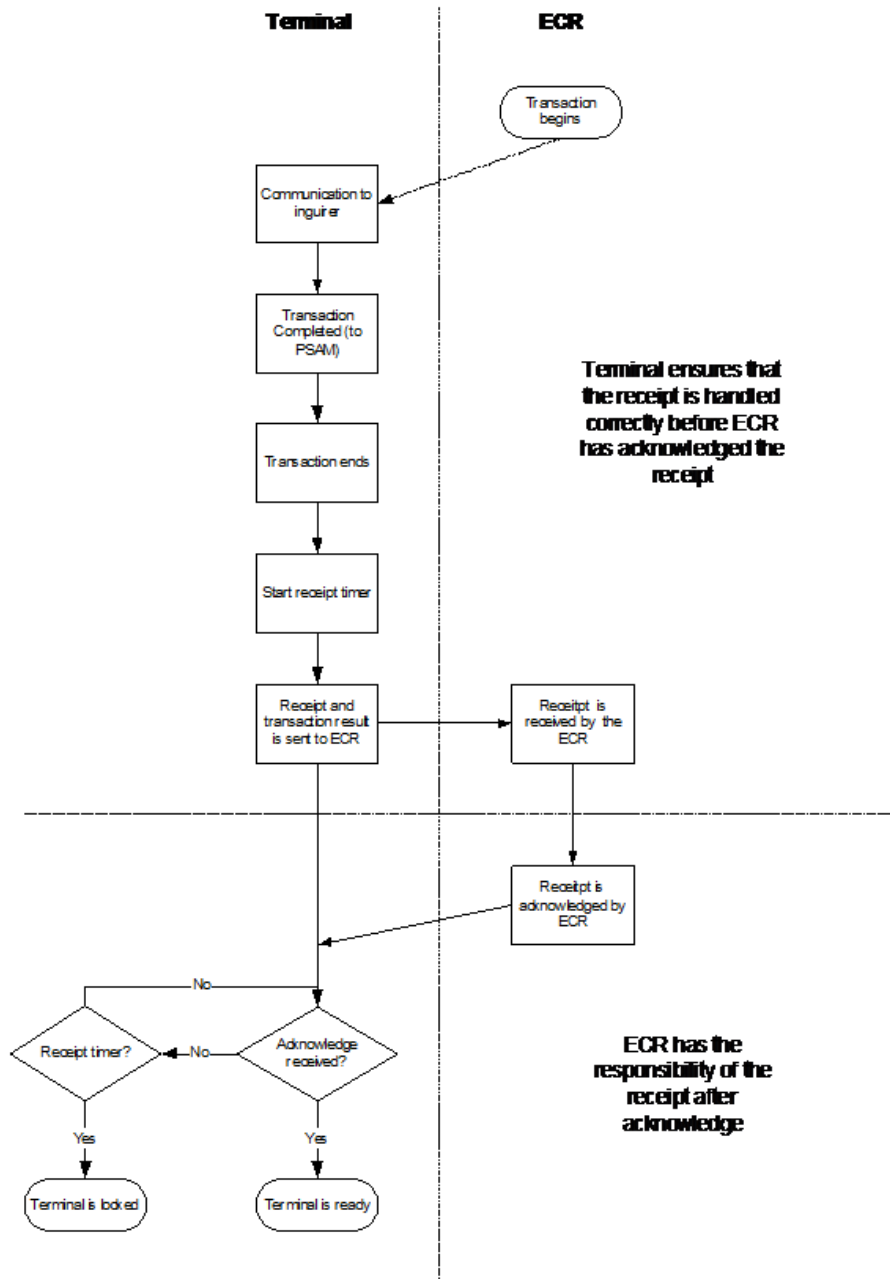
ETX : 03  
CRC : 254d

TimeStamp: 11:50:22  
ACK : 06  
SEQ : 06

## 2.11 General Flowcharts

### 2.11.1 The important flow of the receipt and the transaction result

As seen on the sketch below, the receipt is the responsibility of the ECR after acknowledging the receipt.





## **2.A Appendix A**

- 2.A.1 Tag definitions
- 2.A.2 Value definitions
  - 2.A.3 Transaction data
  - 2.A.4 Amount data
  - 2.A.5 Command data
- 2.A.6 Result data
  - 2.A.6.1 Result data with Transaction
  - 2.A.6.2 Result data with Admin
- 2.A.7 Binary data
  - 2.A.7.1 Binary data with Info–Text
  - 2.A.7.2 Binary data with Transaction–Result
  - 2.A.7.3 Binary data with Admin–Result
  - 2.A.8 Binary data with Receipt–Text
- 2.A.9 Acqmsg tags
  - 2.A.9.1 Acqmsg data with Transaction
- 2.A.10 Error tags
  - 2.A.10.1 Receiving Error data
- 2.A.11 Abort data
  - 2.A.11.1 Abort data with Info
- 2.A.12 CAC9 data
- 2.A.13 ExtendedECR data
  - 2.A.13.1 ExtendedECR functions
- 2.A.14 Terminal states and number
- 2.A.15 Description of locked states
- 2.A.16 The Merchant events and number
- 2.A.17 The Merchant state–event diagram
- 2.A.18 The Merchant state–events

### **2.A.1 Tag definitions**

The general rule is that an odd tag definition requires an answer, therefore all container tags will return at some point.

<b>Container tags (CDO)</b>	<b>Definition</b>	<b>Length</b>	<b>Value Description</b>
INFO	0x60	CDO	Contains info like status messages
DATA_1	0x62	CDO	Contains data needing no response
DATA	0x63	CDO	Contains data – needing response
CARDDATA	0x6F	CDO	Holding card data info
TRANSACTION	0x65	CDO	Contains transaction data
ADMIN	0x67	CDO	Contains data for administrative functions
INIT	0x69	CDO	Contains data for init functions
RECEIPT	0x6B	CDO	Contains receipt data
ERROR	0x6C	CDO	Contains error messages
ADVICE_RECON	0x6D	CDO	
HOSTDATA	0x73	CDO	Contains data to/from extern host
HANDLERSTR	0x74	CDO	
PRERESULT	0x75	CDO	
MENU	0x77	CDO	
ENAI	0x79	CDO	IP routing
<b>General tags</b>	<b>Definition</b>	<b>Length</b>	<b>Description</b>
STOPLIST	0x5B	8 bytes	Check Stop List/authorisation code
BINARY	0x80	Max 1000	Binary data
TEXT	0x82	Max 1000	Text data, e.g. status or receipt
RESULT	0x84	1 byte	Result data, e.g. transaction result info
FROM_STATE	0x85	1 byte	State machine data
STATE	0x86	1 byte	State machine data
EVENT	0x87	1 byte	Event in state machine
COMMAND	0x88	Max 1000	e.g. 1 for a admin terminal report
SEQNO	0x89	1 byte	Response to command with seq. no.
ExtendedECR	0x90	4 bytes	Extended ECR functions (bitmask)
<b>Connect tags</b>	<b>Definition</b>	<b>Length</b>	<b>Description</b>
CONNECT	0x40	2 bytes	First/second → SW compatibility/new functionality
DISCONNECT	0x42	Empty	
OPEN	0x44	Empty	
CLOSE	0x46	Empty	

<b>Transaction tags</b>	<b>Definition</b>	<b>Length</b>	<b>Description</b>
AMOUNT	0x48	4 bytes	
VAT	0x57	4 bytes	
FEE	0x58	4 bytes	
GRATUITY	0x59	4 bytes	
REF_NO	0x4A	4 bytes	
CU	0x4C	2 (in)/3 (out) bytes	Currency Code (+ exponent)
MI	0x4E	1 byte	Merchant Initiative
TR	0x50	1 byte	Transaction Type
TT	0x56	1 byte	Transaction request
TIME	0x5C	4/5 byte	Unix time stamp/DTHR format
ASW1ASW2	0x13	2 bytes	ASW1ASW2
CARD_SOURCE	0x92	1 byte	value 0,1,2,3,4
PREPAID	0x93	1 byte	value 0,1
TERMENV	0x94	1 byte	value 0,1,2
E_RECEIPT	0x99	Max 65 bytes	
DCCAMOUNT	0xC0	4 bytes	
DCCFEE	0xC1	4 bytes	
DCCGRATUITY	0xC2	4 bytes	
SALDO	0xC3	4 bytes	
BACK	0xC4	4 bytes	
EXP_DATE	0xC5	4 bytes	
NETS_HASH	0xC6	72 bytes	
K_RECEIPT	0xC7	64 bytes	

<b>Card number tags</b>	<b>Definition</b>	<b>Length</b>	<b>Description</b>
LOCALCARDATA	0x71	CDO	Holding the Local Card data info
EIE	0x51	Max 521 bytes	Extended Issuer Envelope data
CARDNUMBER	0x52	Max 37 bytes	Card number as ascii
EXTRA_APP	0X53	Max 1004 bytes	Loyalty equal Extra apps data
AID	0x54	1 byte	Application ID
CRCNUMBER	0x5A	4 bytes	Card Reconciliation Counter number
CARDNAME	0x5E	17 bytes	Card name as ascii

If the terminal supports Card Data Protection, the PAN returned will be truncated according to the Card Schemes rules, i.e. leaving the first 6 and last 4 digits. The remaining digits may be replaced by "A". The full PAN may be padded with a trailing "F" for byte boundary alignment if needed (see

OTRS from Nets).

<b>Info tags</b>	<b>Definition</b>	<b>Length</b>	<b>Description</b>
TSI	0x00	Max 24 bytes	Transaction State Information
CONFIRM	0x02	Empty	
KEYPRESS	0x04	1 byte	Number of terminal numeric key presses
ABORT	0x06	1 byte	
ACQMSG	0x08	Max 1000 bytes	Host advice
CAC9	0x11	1 byte	Enhanced Host advises

<b>Receipt tags</b>	<b>Definition</b>	<b>Length</b>	<b>Description</b>
ATC	0x0A	2 bytes	Consult EMV specification
AED	0x0C	3 bytes	Consult EMV specification
STAN	0x10	3 bytes	Nets reference number (bcd)
PSAMID	0x12	4 bytes	PSAM id.
ACODE	0x14	2 bytes	Action code
CVM	0x16	1 byte	Cardholder verification method, see EMV spec.
TCC	0x17	3 bytes	Transaction Condition Code, see OTRS
AUTCODE	0x18	6 bytes	Nets Authorisation code
VAS	0x19	X bytes	VAS Json receipt (Ex. Swipp transaction)
TERMID	0x1A	8 bytes	Terminal id
EXTRA	0x1C	4 bytes	-
ARC	0x1E	2 bytes	Consult EMV specification
MI_NR	0x01	5 bytes	Merchant info number ascii
MI_NAME	0x03	18 bytes	Merchant info name ascii
MI_CITY	0x05	16 bytes	Merchant info city ascii
MI_ADDR	0x07	24 bytes	Merchant info address ascii
MI_ZIP	0x09	8 bytes	Merchant info zip code ascii
MI_PHONE	0x0B	24 bytes	Merchant info phone number ascii
MI_BRN	0x0D	12 bytes	Merchant info business number (cvr) ascii
TOKEN	0x0E	Max 1024 bytes	Transaction token
CANCELLATION	0x96	1 byte	Cancellation
BATCHNUMBER	0x97	12 bytes	Batch number
PSAM_CREATOR	0x0F	4 bytes	PSAM Creator
DCC_RATE	0x95	8 bytes	DCC rate ascii "1.000000"
DCC_CURRENCY	0x98	2/3 bytes	DCC currency
<b>IP routing tags</b>	<b>Definition</b>	<b>Length</b>	<b>Description</b>
IPADDR	0x8A	X bytes	xxx.xxx.xxx.xxx or host.domain
IPPORT	0x8C	4 bytes	
IPTIMEOUT	0x8E	4 bytes	
<b>Admin tags</b>	<b>Definition</b>	<b>Length</b>	<b>Description</b>
PARAM_STR	0x49	X bytes	Parameters for admin function

### 2.A.2 Value definitions

These tables define the values used when initiating a transaction. These tags are optional to send, so if they not used the default values are:

Default terminal value	Value
MI (Merchant Initiative)	0x00 for Default
CU (Currency Code)	0x00D0 for DKK
TT (Transaction Type)	0x00 for Purchase
TR (Transaction request)	According to the Nets OTRS specification
CARD_SOURCE	0x00 default
REF_NO	Any value

### 2.A.3 Transaction data

If the transaction tags are used, they can be given the following values. The merchant initiative is a one-byte value, which can hold the following values:

MI (Merchant Initiative)	Value
Default value	0x00
Forced PIN	0x81
Signature transaction	0x82
No CVM	0x80

The currency code is a two-byte value. The terminal has by default 10 currency codes embedded. More can be added by appointment with Verifone Denmark A/S.

CU (Currency Code)	Value
DKK	0x00D0 (208d)
EUR	0x03D2 (978d)
Etc...	...

The Transaction Type is a one-byte value, which defines the transaction request (see Nets

OTRS).

<b>TR (Transaction Request)</b>	<b>Value</b>
Purchase Transaction	0x00
Refund Transaction	0x20

The Transaction Type is a one-byte value, which defines the transaction type to initiate.

<b>TT (Transaction Type)</b>	<b>Value</b>
Purchase Transaction	0x00
Refund Transaction	0x01

The reference number is a four-byte value chosen by the ECR and can be used to keep track of a transaction.

<b>REF_NO</b>	<b>Value</b>
e.g. keep track of purchase number 1	0x00000001

#### **2.A.4 Amount data**

The amount is given in the smallest possible unit (øre for DKK) consists of 4 bytes.

<b>Amount</b>	<b>Value</b>
e.g. 19995 (in DKK → 199 kroner and 95 øre)	0x00004E1B

## 2.A.5 Command data

When initiating an administrative function e.g. like this:

*ADMIN*

|  
  *COMMAND*  
  *[PARAM\_STR]*

*The command defines which function to initiate*  
*Optional parameters to selected commands*

Command	Value	ASCII CSV string parameter(s)
ADMIN_ENDOFDAY	0x01	
ADMIN_ENDOFDAYLOG	0x02	
ADMIN_REPORT_TERMINALREPORT	0x03	
ADMIN_REPORT_TOTALS	0x04	
ADMIN_REPORT_LOG	0x05	
ADMIN_REPORT_OLDLOG	0x06	
ADMIN_LASTRECEIPT	0x07	
ADMIN_UNLOCK_RECEIPT	0x08	
ADMIN_CLOCKSYNCPBS	0x09	
ADMIN_CLOCKSYNCPPOINT	0x0A	
ADMIN_SENDLOG	0x0B	
ADMIN_CLEARDATASTORE	0x0C	
ADMIN_DOWNLOADPROGRAM	0x0D	
ADMIN_DOWNLOADPARAM	0x0E	
ADMIN_DOWNLOADPAN	0x0F	
ADMIN_DOWNLOADTLCMDB	0x10	
ADMIN_RESTORETLCMDB	0x11	
ADMIN_CONTRASTUP	0x12	
ADMIN_CONTRASTDOWN	0x13	
ADMIN_RESTARTTERMINAL	0x14	



Command	Value	ASCII CSV string parameter(s)
ADMIN_EJECTCARD	0x15	
ADMIN_MSC (for future use)	0x16	
ADMIN_BACKLIGHT_ON	0x17	
ADMIN_BACKLIGHT_OFF	0x18	
ADMIN_REPORT_NETWORKREPORT	0x19	
ADMIN_REPORT_RATESREPORT	0x1A	
ADMIN GRATUITY RECEIPT	0x1B	<AMOUNT >,<CURRENCYCODE >
ADMIN_EXCLUDE_DATASTORE_RECORD_WITH_STAN	0x1C	<PSAMID>,<RECORDID>,<FILENUMBER>,<STAN>
ADMIN_ADVICE_FORWARDING	0x1D	
ADMIN_REPORT_FILE5STATUS	0x1E	
ADMIN_RESERVED_FOR_OCX	0x1F	
ADMIN_REPORT_PCT	0x20	
ADMIN_DOWNLOAD_DCC_RATES	0x21	
ADMIN_UPDATEPSAM	0x22	
ADMIN_UPDATEFEETABLE	0x23	
ADMIN_UPDATE_SALT	0x24	
ADMIN_GETADVICERECON	0x25	<ADVICE_FILE_INDEX>
ADMIN_SET_BATCH_NUMBERS	0x26	<BATCH_NUMBER>
ADMIN_REPORT_TCS	0x27	
ADMIN_REPORT_TPROPS_CVS	0x28	
ADMIN_EVENTLOG_PRINT	0x29	
ADMIN_EVENTLOG_SEND	0x2A	<IP ADDRESS>
ADMIN_EVENTLOG_DELETE	0x2B	<IP ADDRESS>
ADMIN_GETIP_SETTINGS	0x2C	
ADMIN_SETIP_SETTINGS	0x2D	<IP ADDRESS>, <SUBNET>, <GATEWAY>, <DNS1>,<DNS2>,<DOMAIN>
ADMIN_DOWNLOAD_IMAGES	0x2E	
ADMIN_CHECK_CARD	0x2F	
ADMIN_GETTELEDONE_SETTINGS	0x30	
ADMIN_SETTELEDONE_SETTINGS	0x31	<IP ADDRESS>,<PORT>
ADMIN_REPORT_CTLREPORT	0x32	
ADMIN_USER_INPUT	0x33	<MIN>,<MAX>,<TEXTID1>,<TEXTID2>,<TIMEOUT>
ADMIN_MAX	0x34	
ADMIN_GET_DC_PROPERTIES_STAN	0x80	
ADMIN_GET_ID	0x83	
ADMIN_SET_ECR_EXTENDED_FUNCTIONS	0x84	
ADMIN_GET_ECR_EXTENDED_FUNCTIONS	0x85	

#### Example: Get IP settings

TimeStamp: 11:43:07

STX : 02

SEQ : 01

```

TAG : 67 - PTAG_ADMIN Container
TAGlen : DLE 03
TAG : 88 - PTAG_COMMAND
TAGlen : 01
TAGvalue : 2c - ADMIN_GETIP_SETTINGS ', '

ETX : 03
CRC : b73b

TimeStamp: 11:43:07
ACK : 06
SEQ : 01

TimeStamp: 11:43:07
STX : 02
SEQ : 01
TAG : 60 - PTAG_INFO Container
TAGlen : 24
TAG : 00 - PTAG_TSI
TAGlen : 13
TAGvalue : 48 45 4e 54 45 52 20 49 50 20 4f 50 53 c6 54 4e 'HENTER IP OPSAETN'
TAGvalue : 49 4e 47 'ING'

TAG : 80 - PTAG_BINARY
TAGlen : 01
TAGvalue : 01 ' .'

TAG : 89 - PTAG_SEQNO
TAGlen : 01
TAGvalue : 01 ' .'

TAG : 85 - PTAG_FROMSTATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

TAG : 87 - PTAG_EVENT
TAGlen : 01
TAGvalue : 28 - PEVENT_ECR_INFO '('

TAG : 86 - PTAG_STATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

ETX : 03
CRC : ed15

TimeStamp: 11:43:07
ACK : 06
SEQ : 01

TimeStamp: 11:43:07
STX : 02

```

```

SEQ : 02
TAG : 6b - PTAG_RECEIPT Container
TAGlen : 82 01 41
TAG : 82 - PTAG_TEXT
TAGlen : 82 01 28
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 20 '*****. '
TAGvalue : 20 20 20 20 20 20 20 20 20 20 20 20 20 49 50 20 ' IP '
TAGvalue : 53 65 74 74 69 6e 67 73 20 52 61 70 70 6f 72 74 'Settings Rapport'
TAGvalue : 20 20 20 20 20 20 20 20 20 20 20 20 20 0a 2a 2a ' .**'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 49 50 3b '*****.IP;'
TAGvalue : 31 30 2e 30 2e 30 2e 31 35 36 0a 53 55 42 4e 45 '10.0.0.156.SUBNE'
TAGvalue : 54 3b 32 35 35 2e 32 35 35 2e 30 2e 30 0a 47 41 'T;255.255.0.0.GA'
TAGvalue : 54 45 57 41 59 3b 31 30 2e 30 2e 31 30 2e 31 0a 'TEWAY;10.0.10.1.'
TAGvalue : 44 4e 53 31 3b 31 30 2e 30 2e 31 30 2e 32 30 34 'DNS1;10.0.10.204'
TAGvalue : 0a 44 4e 53 32 3b 31 30 2e 31 2e 31 30 2e 31 30 '.DNS2;10.1.10.10'
TAGvalue : 0a 44 4f 4d 41 49 4e 3b 0a 44 48 43 50 3b 41 43 '.DOMAIN;.DHCP;AC'
TAGvalue : 54 49 56 45 0a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 'TIVE.*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 0a 0a 0a 0a 0a '****....'

TAG : 4a - PTAG_REF_NO
TAGlen : 04
TAGvalue : 00 00 00 00 '....'

TAG : 80 - PTAG_BINARY
TAGlen : 01
TAGvalue : 00 ' .'

TAG : 89 - PTAG_SEQNO
TAGlen : 01
TAGvalue : 01 ' .'

TAG : 85 - PTAG_FROMSTATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

TAG : 87 - PTAG_EVENT
TAGlen : 01
TAGvalue : 22 - PEVENT_ECR_LAST_RECEIPT ''

TAG : 86 - PTAG_STATE
TAGlen : 01
TAGvalue : 2d - PSTATE_ADMIN_RECEIPT_PRINTED '- '

```

```

ETX : 03
CRC : 6584

```

```

*****
IP Settings Rapport
*****

```

IP;10.0.0.156  
SUBNET;255.255.0.0  
GATEWAY;10.0.10.1  
DNS1;10.0.10.204  
DNS2;10.1.10.10  
DOMAIN;  
DHCP;ACTIVE  
\*\*\*\*\*

TimeStamp: 11:43:07  
ACK : 06  
SEQ : 02

TimeStamp: 11:43:07  
STX : 02  
SEQ : 02  
TAG : 6b - PTAG\_RECEIPT Container  
TAGlen : DLE 03  
TAG : 84 - PTAG\_RESULT  
TAGlen : 01  
TAGvalue : 00 - OK '.'

ETX : 03  
CRC : 24be

TimeStamp: 11:43:07  
ACK : 06  
SEQ : 02

TimeStamp: 11:43:07  
STX : 02  
SEQ : 03  
TAG : 67 - PTAG\_ADMIN Container  
TAGlen : 0f  
TAG : 84 - PTAG\_RESULT  
TAGlen : 01  
TAGvalue : 00 - OK '.'  
  
TAG : 89 - PTAG\_SEQNO  
TAGlen : 01  
TAGvalue : DLE 02 '..'  
  
TAG : 85 - PTAG\_FROMSTATE  
TAGlen : 01  
TAGvalue : 2b - PSTATE\_ADMIN '+'  
  
TAG : 87 - PTAG\_EVENT  
TAGlen : 01  
TAGvalue : 1e - PEVENT\_ECR\_TRANSACTION\_COMPLETED '.'  
  
TAG : 86 - PTAG\_STATE  
TAGlen : 01

```

TAGvalue : 07 - PSTATE_OPEN ','
ETX : 03
CRC : c3cf

TimeStamp: 11:43:07
ACK : 06
SEQ : 03

```

### Example: Set IP settings

```

TimeStamp: 11:43:20
STX : 02
SEQ : 03
  TAG : 67 - PTAG_ADMIN Container
  TAGlen : 3d
    TAG : 88 - PTAG_COMMAND
    TAGlen : 01
    TAGvalue : 2d - ADMIN_SETIP_SETTINGS '-'

    TAG : 49 - PTAG_PARAM_STR
    TAGlen : 38
    TAGvalue : 31 30 2e 30 2e 30 2e 31 35 36 2c 32 35 35 2e 32 '10.0.0.156,255.2'
    TAGvalue : 35 35 2e 30 2e 30 2c 31 30 2e 30 2e 31 30 2e 31 '55.0.0,10.0.10.1'
    TAGvalue : 2c 31 30 2e 30 2e 31 30 2e 32 30 34 2c 31 30 2e ' ,10.0.10.204,10.'
    TAGvalue : 31 2e 31 30 2e 31 30 2c '1.10.10,'

ETX : 03
CRC : ccbc

TimeStamp: 11:43:20
ACK : 06
SEQ : 03

TimeStamp: 11:43:20
STX : 02
SEQ : 04
  TAG : 60 - PTAG_INFO Container
  TAGlen : 24
    TAG : 00 - PTAG_TSI
    TAGlen : 13
    TAGvalue : 53 c6 54 54 45 52 20 49 50 20 4f 50 53 c6 54 4e 'SAETTER IP OPSAETN'
    TAGvalue : 49 4e 47 'ING'

    TAG : 80 - PTAG_BINARY
    TAGlen : 01
    TAGvalue : 01 ','

    TAG : 89 - PTAG_SEQNO
    TAGlen : 01
    TAGvalue : DLE 03 '..'

    TAG : 85 - PTAG_FROMSTATE

```



```

TAGvalue : 74 20 74 65 72 6d 69 6e 61 6c 20 0a 2a 2a 2a 2a 't terminal .****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 0a 0a 0a '*****...'

TAG : 4a - PTAG_REF_NO
TAGlen : 04
TAGvalue : 00 00 00 00 '....'

TAG : 80 - PTAG_BINARY
TAGlen : 01
TAGvalue : 00 '.'

TAG : 89 - PTAG_SEQNO
TAGlen : 01
TAGvalue : DLE 03 '..'

TAG : 85 - PTAG_FROMSTATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

TAG : 87 - PTAG_EVENT
TAGlen : 01
TAGvalue : 22 - PEVENT_ECR_LAST_RECEIPT '''

TAG : 86 - PTAG_STATE
TAGlen : 01
TAGvalue : 2d - PSTATE_ADMIN_RECEIPT_PRINTED '-'

ETX : 03
CRC : 7ae3
*****
Set IP Settings Rapport
*****
Success -Fixed IP selected
IP;10.0.0.156
SUBNET;255.255.0.0
GATEWAY;10.0.10.1
DNS1;10.0.10.204
DNS2;10.1.10.10
DOMAIN;

Changes will take effect after
next boot of terminal.

Use admin function 20 - Restart terminal
*****

TimeStamp: 11:43:20
ACK : 06
SEQ : 05

TimeStamp: 11:43:20

```

STX : 02  
SEQ : 04  
TAG : 6b - PTAG\_RECEIPT Container  
TAGlen : DLE 03  
TAG : 84 - PTAG\_RESULT  
TAGlen : 01  
TAGvalue : 00 - OK ','

ETX : 03  
CRC : a494

TimeStamp: 11:43:21  
ACK : 06  
SEQ : 04

TimeStamp: 11:43:21  
STX : 02  
SEQ : 06  
TAG : 67 - PTAG\_ADMIN Container  
TAGlen : 0f  
TAG : 84 - PTAG\_RESULT  
TAGlen : 01  
TAGvalue : 00 - OK ','  
  
TAG : 89 - PTAG\_SEQNO  
TAGlen : 01  
TAGvalue : 04 ','  
  
TAG : 85 - PTAG\_FROMSTATE  
TAGlen : 01  
TAGvalue : 2b - PSTATE\_ADMIN '+'  
  
TAG : 87 - PTAG\_EVENT  
TAGlen : 01  
TAGvalue : 1e - PEVENT\_ECR\_TRANSACTION\_COMPLETED ','  
  
TAG : 86 - PTAG\_STATE  
TAGlen : 01  
TAGvalue : 07 - PSTATE\_OPEN ','

ETX : 03  
CRC : 8729

TimeStamp: 11:43:21  
ACK : 06  
SEQ : 06

### Example: Set IP DHCP

TimeStamp: 12:01:53  
STX : 02  
SEQ : 05  
TAG : 67 - PTAG\_ADMIN Container  
TAGlen : 0a



```

TAG : 88 - PTAG_COMMAND
TAGlen : 01
TAGvalue : 2d - ADMIN_SETIP_SETTINGS '-'

TAG : 49 - PTAG_PARAM_STR
TAGlen : 05
TAGvalue : 2c 2c 2c 2c 2c ',,,,,'

ETX : 03
CRC : 1664

TimeStamp: 12:01:53
ACK : 06
SEQ : 05

TimeStamp: 12:01:54
STX : 02
SEQ : 07
TAG : 60 - PTAG_INFO Container
TAGlen : 24
TAG : 00 - PTAG_TSI
TAGlen : 13
TAGvalue : 53 c6 54 54 45 52 20 49 50 20 4f 50 53 c6 54 4e 'SAETTER IP OPSAETN'
TAGvalue : 49 4e 47 'ING'

TAG : 80 - PTAG_BINARY
TAGlen : 01
TAGvalue : 01 ' .'

TAG : 89 - PTAG_SEQNO
TAGlen : 01
TAGvalue : 05 ' .'

TAG : 85 - PTAG_FROMSTATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

TAG : 87 - PTAG_EVENT
TAGlen : 01
TAGvalue : 28 - PEVENT_ECR_INFO '('

TAG : 86 - PTAG_STATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

ETX : 03
CRC : 9143

TimeStamp: 12:01:54
ACK : 06
SEQ : 07

```

TimeStamp: 12:01:55

STX : 02

SEQ : 08

TAG : 6b - PTAG\_RECEIPT Container

TAGlen : 82 01 85

TAG : 82 - PTAG\_TEXT

TAGlen : 82 01 6c

TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 20 '\*\*\*\*\*. '  
TAGvalue : 20 20 20 20 20 20 20 20 20 20 53 65 74 20 49 50 20 ' Set IP '  
TAGvalue : 53 65 74 74 69 6e 67 73 20 52 61 70 70 6f 72 74 'Settings Rapport'  
TAGvalue : 20 20 20 20 20 20 20 20 20 20 20 20 20 0a 2a 2a ' .\*\*'  
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 20 20 20 '\*\*\*\*\*. '  
TAGvalue : 20 20 20 20 53 75 63 63 65 73 73 20 2d 20 44 48 ' Success - DH'  
TAGvalue : 43 50 20 73 65 6c 65 63 74 65 64 20 20 20 20 20 20 'CP selected '  
TAGvalue : 20 20 20 20 20 20 0a 0a 20 20 20 20 20 20 43 68 ' .. Ch'  
TAGvalue : 61 6e 67 65 73 20 77 69 6c 6c 20 74 61 6b 65 20 'anges will take '  
TAGvalue : 65 66 66 65 63 74 20 61 66 74 65 72 20 20 20 20 'effect after '  
TAGvalue : 20 20 0a 20 20 20 20 20 20 20 20 20 20 6e 65 78 74 ' . next'  
TAGvalue : 20 62 6f 6f 74 20 6f 66 20 74 65 72 6d 69 6e 61 ' boot of termina'  
TAGvalue : 6c 2e 20 20 20 20 20 20 20 20 20 20 20 0a 0a 20 'l. .. '  
TAGvalue : 55 73 65 20 61 64 6d 69 6e 20 66 75 6e 63 74 69 'Use admin functi'  
TAGvalue : 6f 6e 20 32 30 20 2d 20 52 65 73 74 61 72 74 20 'on 20 - Restart '  
TAGvalue : 74 65 72 6d 69 6e 61 6c 20 0a 2a 2a 2a 2a 2a 2a 'terminal .\*\*\*\*\*'  
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 0a 0a 0a 0a '\*\*\*\*\*...'

TAG : 4a - PTAG\_REF\_NO

TAGlen : 04

TAGvalue : 00 00 00 00 '....'

TAG : 80 - PTAG\_BINARY

TAGlen : 01

TAGvalue : 00 '.'

TAG : 89 - PTAG\_SEQNO

TAGlen : 01

TAGvalue : 05 '.'

TAG : 85 - PTAG\_FROMSTATE

TAGlen : 01

TAGvalue : 2b - PSTATE\_ADMIN '+'

TAG : 87 - PTAG\_EVENT

TAGlen : 01

TAGvalue : 22 - PEVENT\_ECR\_LAST\_RECEIPT ''

TAG : 86 - PTAG\_STATE

TAGlen : 01

TAGvalue : 2d - PSTATE\_ADMIN\_RECEIPT\_PRINTED '-'

```

ETX : 03
CRC : 92ac
*****
      Set IP Settings Rapport
*****
      Success - DHCP selected

      Changes will take effect after
      next boot of terminal.

Use admin function 20 - Restart terminal
*****

```

```

TimeStamp: 12:01:55
ACK : 06
SEQ : 08

```

```

TimeStamp: 12:01:55
STX : 02
SEQ : 06
  TAG : 6b - PTAG_RECEIPT Container
  TAGlen : DLE 03
    TAG : 84 - PTAG_RESULT
    TAGlen : 01
    TAGvalue : 00 - OK ','

```

```

ETX : 03
CRC : 254d

```

```

TimeStamp: 12:01:55
ACK : 06
SEQ : 06

```

```

TimeStamp: 12:01:55
STX : 02
SEQ : 09
  TAG : 67 - PTAG_ADMIN Container
  TAGlen : 0f
    TAG : 84 - PTAG_RESULT
    TAGlen : 01
    TAGvalue : 00 - OK ','

    TAG : 89 - PTAG_SEQNO
    TAGlen : 01
    TAGvalue : 06 ','

    TAG : 85 - PTAG_FROMSTATE
    TAGlen : 01
    TAGvalue : 2b - PSTATE_ADMIN '+'

    TAG : 87 - PTAG_EVENT

```

TAGlen : 01  
TAGvalue : 1e - PEVENT\_ECR\_TRANSACTION\_COMPLETED ',.'

TAG : 86 - PTAG\_STATE  
TAGlen : 01  
TAGvalue : 07 - PSTATE\_OPEN ',.'

ETX : 03  
CRC : db84

TimeStamp: 12:01:55  
ACK : 06  
SEQ : 09

Example: ADMIN\_SETTELEDONE\_SETTINGS IP address: 10.0.0.123 port: 2001

TimeStamp: 11:54:16  
STX : 02  
SEQ : 01  
TAG : 67 - PTAG\_ADMIN Container  
TAGlen : 14  
TAG : 88 - PTAG\_COMMAND  
TAGlen : 01  
TAGvalue : 31 - ADMIN\_SETTELEDONE\_SETTINGS '1'  
  
TAG : 49 - PTAG\_PARAM\_STR  
TAGlen : 0f  
TAGvalue : 31 30 2e 30 2e 30 2e 31 32 33 2c 32 30 30 31 '10.0.0.123,2001'

ETX : 03  
CRC : aa57

TimeStamp: 11:54:16  
ACK : 06  
SEQ : 01  
TimeStamp: 11:54:16  
STX : 02  
SEQ : 01  
TAG : 60 - PTAG\_INFO Container  
TAGlen : 24  
TAG : 00 - PTAG\_TSI  
TAGlen : 13  
TAGvalue : 53 c6 54 20 54 45 4c 45 44 4f 4e 45 20 4f 50 53 'SAET TELEDONE OPS'  
TAGvalue : c6 54 2e 'AET.'  
  
TAG : 80 - PTAG\_BINARY  
TAGlen : 01  
TAGvalue : 01 ',.'  
  
TAG : 89 - PTAG\_SEQNO  
TAGlen : 01  
TAGvalue : 01 ',.'  
  
TAG : 85 - PTAG\_FROMSTATE  
TAGlen : 01

```

TAGvalue : 2b - PSTATE_ADMIN '+'

TAG : 87 - PTAG_EVENT
TAGlen : 01
TAGvalue : 28 - PEVENT_ECR_INFO '('

TAG : 86 - PTAG_STATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

ETX : 03
CRC : c237

TimeStamp: 11:54:16
ACK : 06
SEQ : 01

TimeStamp: 11:54:16
STX : 02
SEQ : 02
TAG : 6b - PTAG_RECEIPT Container
TAGlen : 82 01 2d
TAG : 82 - PTAG_TEXT
TAGlen : 82 01 14
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 20 '*****. '
TAGvalue : 20 20 20 20 20 20 53 65 74 20 54 45 4c 45 44 4f ' Set TELED0'
TAGvalue : 4e 45 20 53 65 74 74 69 6e 67 73 20 52 61 70 70 'NE Settings Rapp'
TAGvalue : 6f 72 74 20 20 20 20 20 20 20 20 20 20 0a 2a 2a 'ort .**'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 20 20 20 '*****. '
TAGvalue : 20 20 53 75 63 63 65 73 73 20 2d 20 54 65 6c 65 ' Success - Tele'
TAGvalue : 44 6f 6e 65 20 73 65 6c 65 63 74 65 64 20 20 20 'Done selected '
TAGvalue : 20 20 20 20 20 20 0a 20 20 20 20 20 20 20 20 20 ' . '
TAGvalue : 49 50 3b 31 30 2e 30 2e 30 2e 31 32 33 0a 20 20 'IP;10.0.0.123. '
TAGvalue : 20 20 20 20 20 20 20 50 4f 52 54 3b 32 30 30 31 ' PORT;2001'
TAGvalue : 0a 0a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '.....'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 0a 0a 0a 0a '....'

TAG : 4a - PTAG_REF_NO
TAGlen : 04
TAGvalue : 00 00 00 00 '....'

TAG : 80 - PTAG_BINARY
TAGlen : 01
TAGvalue : 00 ' .'

TAG : 89 - PTAG_SEQNO
TAGlen : 01

```

```

TAGvalue : 01 ','
TAG : 85 - PTAG_FROMSTATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'
TAG : 87 - PTAG_EVENT
TAGlen : 01
TAGvalue : 22 - PEVENT_ECR_LAST_RECEIPT ''
TAG : 86 - PTAG_STATE
TAGlen : 01
TAGvalue : 2d - PSTATE_ADMIN_RECEIPT_PRINTED '- '
ETX : 03
CRC : Obf4

```

```

*****
Set TELEDONE Settings Rapport
*****
Success - TeleDone selected
IP;10.0.0.123
PORT;2001

```

```

*****
TimeStamp: 11:54:16
ACK : 06
SEQ : 02

```

```

TimeStamp: 11:54:16
STX : 02
SEQ : 02
TAG : 6b - PTAG_RECEIPT Container
TAGlen : DLE 03
TAG : 84 - PTAG_RESULT
TAGlen : 01
TAGvalue : 00 - OK ','

```

```

ETX : 03
CRC : 24be

```

```

TimeStamp: 11:54:16
ACK : 06
SEQ : 02

```

```

TimeStamp: 11:54:16
STX : 02
SEQ : 03
TAG : 67 - PTAG_ADMIN Container
TAGlen : 0f

```

```

TAG : 84 - PTAG_RESULT
TAGlen : 01
TAGvalue : 00 - OK ','

TAG : 89 - PTAG_SEQNO
TAGlen : 01
TAGvalue : DLE 02 '...'

TAG : 85 - PTAG_FROMSTATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

TAG : 87 - PTAG_EVENT
TAGlen : 01
TAGvalue : 1e - PEVENT_ECR_TRANSACTION_COMPLETED ','

TAG : 86 - PTAG_STATE
TAGlen : 01
TAGvalue : 07 - PSTATE_OPEN ','

```

```

ETX : 03
CRC : c3cf

```

```

TimeStamp: 11:54:16
ACK : 06
SEQ : 03

```

Example: ADMIN\_SETTELEDONE\_SETTINGS remove entry IP adresse: Empty Port: Empty

```

TimeStamp: 11:56:18
STX : 02
SEQ : 03
TAG : 67 - PTAG_ADMIN Container
TAGlen : 06
TAG : 88 - PTAG_COMMAND
TAGlen : 01
TAGvalue : 31 - ADMIN_SETTELEDONE_SETTINGS '1'

TAG : 49 - PTAG_PARAM_STR
TAGlen : 01
TAGvalue : 2c ',,'

```

```

ETX : 03
CRC : 9157

```

```

TimeStamp: 11:56:19
ACK : 06
SEQ : 03

```

```

TimeStamp: 11:56:19
STX : 02
SEQ : 04
TAG : 60 - PTAG_INFO Container

```

```

TAGlen : 24
TAG : 00 - PTAG_TSI
TAGlen : 13
TAGvalue : 53 c6 54 20 54 45 4c 45 44 4f 4e 45 20 4f 50 53 'SAET TELEDONE OPS'
TAGvalue : c6 54 2e 'AET.'

TAG : 80 - PTAG_BINARY
TAGlen : 01
TAGvalue : 01 '.,'

TAG : 89 - PTAG_SEQNO
TAGlen : 01
TAGvalue : DLE 03 '.,'

TAG : 85 - PTAG_FROMSTATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

TAG : 87 - PTAG_EVENT
TAGlen : 01
TAGvalue : 28 - PEVENT_ECR_INFO '('

TAG : 86 - PTAG_STATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

ETX : 03
CRC : d5cb

TimeStamp: 11:56:19
ACK : 06
SEQ : 04

TimeStamp: 11:56:19
STX : 02
SEQ : 05
TAG : 6b - PTAG_RECEIPT Container
TAGlen : 82 01 DLE 02
TAG : 82 - PTAG_TEXT
TAGlen : 81 ea
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 20 '*****. '
TAGvalue : 20 20 20 20 20 20 53 65 74 20 54 45 4c 45 44 4f ' Set TELEDONE'
TAGvalue : 4e 45 20 53 65 74 74 69 6e 67 73 20 52 61 70 70 'NE Settings Rapp'
TAGvalue : 6f 72 74 20 20 20 20 20 20 20 20 20 20 20 0a 2a 2a 'ort .**'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 53 75 63 '*****.Suc'
TAGvalue : 63 65 73 73 20 2d 20 54 65 6c 65 44 6f 6e 65 20 'cess - TeleDone '
TAGvalue : 54 4c 43 4d 44 42 20 65 6e 74 72 79 20 72 65 6d 'TLCMDB entry rem'
TAGvalue : 6f 76 65 64 20 20 0a 0a 2a 2a 2a 2a 2a 2a 2a 2a 'oved ..*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'

```



```

TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 0a 0a 0a 0a '*****...'

TAG : 4a - PTAG_REF_NO
TAGlen : 04
TAGvalue : 00 00 00 00 '....'

TAG : 80 - PTAG_BINARY
TAGlen : 01
TAGvalue : 00 '.'

```

```

*****
Set TELEDONE Settings Rapport
*****
Success - TeleDone TLCMDB entry removed
*****

```

```

TimeStamp: 11:56:19
ACK : 06
SEQ : 05

```

```

TimeStamp: 11:56:19
STX : 02
SEQ : 04
TAG : 6b - PTAG_RECEIPT Container
TAGlen : DLE 03
TAG : 84 - PTAG_RESULT
TAGlen : 01
TAGvalue : 00 - OK '.'

```

```

ETX : 03
CRC : a494

```

TimeStamp: 11:56:19

ACK : 06

SEQ : 04

TimeStamp: 11:56:19

STX : 02

SEQ : 06

TAG : 67 - PTAG\_ADMIN Container

TAGlen : 0f

TAG : 84 - PTAG\_RESULT

TAGlen : 01

TAGvalue : 00 - OK ','

TAG : 89 - PTAG\_SEQNO

TAGlen : 01

TAGvalue : 04 ','

TAG : 85 - PTAG\_FROMSTATE

TAGlen : 01

TAGvalue : 2b - PSTATE\_ADMIN '+'

TAG : 87 - PTAG\_EVENT

TAGlen : 01

TAGvalue : 1e - PEVENT\_ECR\_TRANSACTION\_COMPLETED ','

TAG : 86 - PTAG\_STATE

TAGlen : 01

TAGvalue : 07 - PSTATE\_OPEN ','

ETX : 03

CRC : 8729

TimeStamp: 11:56:19

ACK : 06

SEQ : 06

Example: ADMIN\_GETTELEDONE\_SETTINGS Terminal have entry

TimeStamp: 11:57:34

STX : 02

SEQ : 07

TAG : 67 - PTAG\_ADMIN Container

TAGlen : DLE 03

TAG : 88 - PTAG\_COMMAND

TAGlen : 01

TAGvalue : 30 - ADMIN\_GETTELEDONE\_SETTINGS '0'

ETX : 03

CRC : 3fd1

TimeStamp: 11:57:34

ACK : 06

SEQ : 07

TimeStamp: 11:57:34  
 STX : 02  
 SEQ : 0a  
   TAG : 60 - PTAG\_INFO Container  
   TAGlen : 25  
     TAG : 00 - PTAG\_TSI  
     TAGlen : 14  
     TAGvalue : 48 45 4e 54 20 54 45 4c 45 44 4f 4e 45 20 4f 50 'HENT TELEDONE OP'  
     TAGvalue : 53 c6 54 2e 'SAET.'  
     TAG : 80 - PTAG\_BINARY  
     TAGlen : 01  
     TAGvalue : 01 '.'  
     TAG : 89 - PTAG\_SEQNO  
     TAGlen : 01  
     TAGvalue : 07 '.'  
     TAG : 85 - PTAG\_FROMSTATE  
     TAGlen : 01  
     TAGvalue : 2b - PSTATE\_ADMIN '+'  
     TAG : 87 - PTAG\_EVENT  
     TAGlen : 01  
     TAGvalue : 28 - PEVENT\_ECR\_INFO '('  
     TAG : 86 - PTAG\_STATE  
     TAGlen : 01  
     TAGvalue : 2b - PSTATE\_ADMIN '+'  
 ETX : 03  
 CRC : 2a10

TimeStamp: 11:57:34  
 ACK : 06  
 SEQ : 0a

TimeStamp: 11:57:34  
 STX : 02  
 SEQ : 0b  
   TAG : 6b - PTAG\_RECEIPT Container  
   TAGlen : 81 ef  
     TAG : 82 - PTAG\_TEXT  
     TAGlen : 81 d7  
     TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
     TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
     TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 20 '\*\*\*\*\*.'  
     TAGvalue : 20 20 20 20 20 20 20 20 20 20 54 45 4c 45 44 4f ' TELEDON'  
     TAGvalue : 4e 45 20 53 65 74 74 69 6e 67 73 20 52 61 70 70 'NE Settings Rapp'  
     TAGvalue : 6f 72 74 20 20 20 20 20 20 20 20 20 0a 2a 2a 2a 'ort .\*\*\*'  
     TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'

TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
 TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 49 50 3b 31 '\*\*\*\*\*.IP;1'  
 TAGvalue : 30 2e 30 2e 30 2e 31 32 33 0a 50 4f 52 54 3b 32 '0.0.0.123.PORT;2'  
 TAGvalue : 30 30 31 0a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '001.\*\*\*\*\*'  
 TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
 TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'  
 TAGvalue : 2a 2a 0a 0a 0a 0a 0a '\*\*\*\*\*'

TAG : 4a - PTAG\_REF\_NO  
 TAGlen : 04  
 TAGvalue : 00 00 00 00 '....'

TAG : 80 - PTAG\_BINARY  
 TAGlen : 01  
 TAGvalue : 00 ' .'

TAG : 89 - PTAG\_SEQNO  
 TAGlen : 01  
 TAGvalue : 07 ' .'

TAG : 85 - PTAG\_FROMSTATE  
 TAGlen : 01  
 TAGvalue : 2b - PSTATE\_ADMIN '+'

TAG : 87 - PTAG\_EVENT  
 TAGlen : 01  
 TAGvalue : 22 - PEVENT\_ECR\_LAST\_RECEIPT ''

TAG : 86 - PTAG\_STATE  
 TAGlen : 01  
 TAGvalue : 2d - PSTATE\_ADMIN\_RECEIPT\_PRINTED '- '

ETX : 03  
 CRC : 41d3

\*\*\*\*\*  
 TELEDONE Settings Rapport  
 \*\*\*\*\*  
 IP;10.0.0.123  
 PORT;2001  
 \*\*\*\*\*

TimeStamp: 11:57:34  
 ACK : 06  
 SEQ : 0b

TimeStamp: 11:57:34  
 STX : 02  
 SEQ : 08  
 TAG : 6b - PTAG\_RECEIPT Container  
 TAGlen : DLE 03  
 TAG : 84 - PTAG\_RESULT

TAGlen : 01  
    TAGvalue : 00 - OK '.'

ETX : 03  
CRC : a4c1

TimeStamp: 11:57:34  
ACK : 06  
SEQ : 08

TimeStamp: 11:57:34  
STX : 02  
SEQ : 0c

    TAG : 67 - PTAG\_ADMIN Container  
    TAGlen : 0f

        TAG : 84 - PTAG\_RESULT  
        TAGlen : 01  
        TAGvalue : 00 - OK '.'

        TAG : 89 - PTAG\_SEQNO  
        TAGlen : 01  
        TAGvalue : 08 '.'

        TAG : 85 - PTAG\_FROMSTATE  
        TAGlen : 01  
        TAGvalue : 2b - PSTATE\_ADMIN '+'

        TAG : 87 - PTAG\_EVENT  
        TAGlen : 01  
        TAGvalue : 1e - PEVENT\_ECR\_TRANSACTION\_COMPLETED '.'

        TAG : 86 - PTAG\_STATE  
        TAGlen : 01  
        TAGvalue : 07 - PSTATE\_OPEN '.'

ETX : 03  
CRC : 23a0

TimeStamp: 11:57:34  
ACK : 06  
SEQ : 0c

Example: ADMIN\_GETTELEDONE\_SETTINGS No entry in terminal

TimeStamp: 12:00:35  
STX : 02  
SEQ : 0b

    TAG : 67 - PTAG\_ADMIN Container  
    TAGlen : DLE 03

        TAG : 88 - PTAG\_COMMAND  
        TAGlen : 01  
        TAGvalue : 30 - ADMIN\_GETTELEDONE\_SETTINGS '0'

ETX : 03  
CRC : 3f84

TimeStamp: 12:00:35  
ACK : 06  
SEQ : 0b

TimeStamp: 12:00:35  
STX : 02  
SEQ : 10

TAG : 60 - PTAG\_INFO Container

TAGlen : 25

TAG : 00 - PTAG\_TSI

TAGlen : 14

TAGvalue : 48 45 4e 54 20 54 45 4c 45 44 4f 4e 45 20 4f 50 'HENT TELEDONE OP'

TAGvalue : 53 c6 54 2e 'SAET.'

TAG : 80 - PTAG\_BINARY

TAGlen : 01

TAGvalue : 01 '.'

TAG : 89 - PTAG\_SEQNO

TAGlen : 01

TAGvalue : 0b '.'

TAG : 85 - PTAG\_FROMSTATE

TAGlen : 01

TAGvalue : 2b - PSTATE\_ADMIN '+'

TAG : 87 - PTAG\_EVENT

TAGlen : 01

TAGvalue : 28 - PEVENT\_ECR\_INFO '('

TAG : 86 - PTAG\_STATE

TAGlen : 01

TAGvalue : 2b - PSTATE\_ADMIN '+'

ETX : 03  
CRC : 4702

TimeStamp: 12:00:35  
ACK : 06  
SEQ : 10

TimeStamp: 12:00:35  
STX : 02  
SEQ : 11

TAG : 6b - PTAG\_RECEIPT Container

TAGlen : 81 e1

TAG : 82 - PTAG\_TEXT

TAGlen : 81 c9

TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '\*\*\*\*\*'

```

TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 20 '*****. '
TAGvalue : 20 20 20 20 20 20 20 20 20 20 54 45 4c 45 44 4f ' TELEDON'
TAGvalue : 4e 45 20 53 65 74 74 69 6e 67 73 20 52 61 70 70 'NE Settings Rapp'
TAGvalue : 6f 72 74 20 20 20 20 20 20 20 20 20 20 0a 2a 2a 2a 'ort .***'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 0a 49 50 3b 0a '*****.IP;.'
TAGvalue : 50 4f 52 54 3b 0a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 'PORT;*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a 2a '*****'
TAGvalue : 2a 2a 2a 2a 0a 0a 0a 0a 0a 0a '****....'

```

```

TAG : 4a - PTAG_REF_NO
TAGlen : 04
TAGvalue : 00 00 00 00 '....'

```

```

TAG : 80 - PTAG_BINARY
TAGlen : 01
TAGvalue : 00 '.,'

```

```

TAG : 89 - PTAG_SEQNO
TAGlen : 01
TAGvalue : 0b '.,'

```

```

TAG : 85 - PTAG_FROMSTATE
TAGlen : 01
TAGvalue : 2b - PSTATE_ADMIN '+'

```

```

TAG : 87 - PTAG_EVENT
TAGlen : 01
TAGvalue : 22 - PEVENT_ECR_LAST_RECEIPT ''

```

```

TAG : 86 - PTAG_STATE
TAGlen : 01
TAGvalue : 2d - PSTATE_ADMIN_RECEIPT_PRINTED '- '

```

```

ETX : 03
CRC : 027c

```

```

*****
TELEDONE Settings Rapport
*****
IP;
PORT;
*****

```

```

TimeStamp: 12:00:35
ACK : 06
SEQ : 11

```

```

TimeStamp: 12:00:35

```

```

STX : 02
SEQ : 0c
  TAG : 6b - PTAG_RECEIPT Container
  TAGlen : DLE 03
    TAG : 84 - PTAG_RESULT
    TAGlen : 01
    TAGvalue : 00 - OK ','

```

```

ETX : 03
CRC : a532

```

```

TimeStamp: 12:00:35
ACK : 06
SEQ : 0c

```

## 2.A.6 Result data

### 2.A.6.1 Result data with Transaction

When receiving RESULT tags in conjunction with TRANSACTION tags (not Local Card), e.g. like this:

```

TRANSACTION
|
RESULT

```

The result value is a direct answer from the host (the transaction inquirer):

Transaction result	Value
Transaction approved	0x00
Transaction not approved	0x01
Fallback transaction	0x02

In cases where the transaction result is 0x02 - fallback transaction, the ECR must restart the transaction once more (send the same data you used to start the transaction leading to the Fallback transaction 0x02 result). This transaction will end up as either approved or not approved.

### 2.A.6.2 Result data with Admin

When receiving RESULT tags in conjunction with ADMIN tags, e.g. like this:

Admin result	Value
Successful	0x00
Unsuccessful	0x01

### 2.A.6.3 Result data with Receipt

When acknowledging receipts, the ECR must reply with a result tag like this:



ADMIN  
└  
RESULT

The result value is reflecting the following:

Receipt result	Value
Saved/Signature OK	0x00
Error/Signature not OK	0x01

## 2.A.7 Binary data

The binary tags are optional

### 2.A.7.1 Binary data with Info–Text

When receiving BINARY tags in conjunction with INFO and TEXT tags, e.g. like this:

INFO  
└  
TEXT ( "Købet er afbrudt" )  
BINARY ( status line 2 )

The binary data holds information on where to place the text:

Placement of text	Value
Status line 1	0x01
Status line 2	0x02
Status line 3	0x03
Status line 4	0x04

### 2.A.7.2 Binary data with Transaction–Result

When receiving BINARY tags in conjunction with TRANSACTION and RESULT tags, e.g. like this:

TRANSACTION  
└  
RESULT ( always 0x01 in this situation )  
BINARY

The binary data holds information on transaction failure.

### 2.A.7.3 Binary data with Admin–Result

When receiving BINARY tags in conjunction with ADMIN and RESULT tags, e.g. like this:

*ADMIN*

|  
*RESULT*  
*BINARY*

The binary data holds information on transaction failure.

## **2.A.8 Binary data with Receipt–Text**

When receiving BINARY tags in conjunction with RECEIPT and TEXT tags e.g. like this:

*RECEIPT*

|  
*TEXT*  
*BINARY*

The binary data holds information on that this segment is part of a larger receipt:

## **2.A.9 Acqmsg tags**

### **2.A.9.1 Acqmsg data with Transaction**

When receiving ACQMSG tags in conjunction TRANSACTION tags e.g. like this:

*TRANSACTION*

|  
*ACQMSG ( value )*

The binary data holds information on that this segment is part of a larger receipt:

<b>Acqmsg</b>	<b>Value</b>
If the acquirer requests an endOfDay routine	0x01

## **2.A.10 Error tags**

### **2.A.10.1 Receiving Error data**

When receiving ERROR tags, they will be constructed like this:

*ERROR*

|  
*BINARY*  
*STATE*  
*MSG - the unknown TLV or TLV not possible in current terminal state*

The BINARY tag can have the following values:

<b>BINARY</b>	<b>Value</b>
Not TLV	0x01
Unknown TLV	0x02
TLV not allowed/possible in the current terminal state	0x03

## 2.A.11 Abort data

### 2.A.11.1 Abort data with Info

When receiving BINARY tags in conjunction with INFO and ABORT tags, e.g. like this:

*INFO*

└  
*ABORT*

The abort data holds information on how the transaction was aborted:

<b>Transaction aborted origin</b>	<b>Value</b>
Operator/ECR aborts transaction	0x01
User	0x02
Timeout in terminal	0x03
MI disagreement (the user swipes the card and the operator sends a signature purchase)	0x04
Transaction type disagreement (the user swipes the card and the operator starts a refund transaction)	0x05

## 2.A.12 CAC9 data

### 2.A.12.1 CAC9 data with Transaction

When receiving CAC9 tags in conjunction TRANSACTION tags e.g. like this:

*TRANSACTION*

└  
*CAC9 ( value )*

The value data holds information:

<b>CAC9</b>	<b>Value</b>
Display Line for Host Message	0xCA
Advice request flag	0xC9

## 2.A.13 ExtendedECR data

### 2.A.13.1 ExtendedECR functions

If no value is send during connect the terminal uses the default setting from its parameter file. ExtendedECR is 4 hex digits.

**Example:**

[ECREXTENDEDFUNCTIONS]

“1af5”

#### **LPP GET terminals EcrExtendedFunctions:**

Via LPP it's possible to get the terminals current setting using an ADMIN command.

**0x85    ADMIN\_GET\_ECR\_EXTENDED\_FUNCTIONS**

The response is a PTAG\_INFO container with a PTAG\_ECREXTENDEDFUNCTIONS tag giving the value.

**0x90    PTAG\_ECREXTENDEDFUNCTIONS**

Tag with the value of terminals EcrExtendedFunctions variable. Observe only the 2 last bytes of 4 bytes is used.

#### **LPP SET terminals EcrExtendedFunctions:**

Via LPP it's possible to set the terminals current setting using an ADMIN command, this value is not saved, so it must be reset every time the terminal is reboot.

**0x84    ADMIN\_SET\_ECR\_EXTENDED\_FUNCTIONS**

With a PTAG\_ECREXTENDEDFUNCTIONS tag giving the value.

**0x90    PTAG\_ECREXTENDEDFUNCTIONS**

Tag with the value to give the terminals EcrExtendedFunctions variable. Observe only the 2 last bytes of 4 bytes long value is used.

## 2.A.14 Terminal states and number

"	// 1
,"	
"INIT",	
"IDLE",	
"IDLE_NP",	
"CONNECTED",	
"CONNECTED_NP",	
"OPEN",	
"OPEN_NP",	
"TRANS",	
"TRANS_ABORT",	// 10
"CARD",	
"CARD_ABORT",	
"TRANS_CARD",	
"CARD_REQ_AMOUNT",	
"TRANS_REQ_AMOUNT",	
"SEND_CARDDATA",	
"AMOUNT_READY",	
"ANS_AMOUNT",	
"TRANS_COMMIT",	
"TRANS_PRINT",	// 20
"PRINT_WAIT_ACK",	
"PRINT_WAIT_ACK_VERIFY",	
"TRANS_WAIT_ACK",	
"TRANS_PRINTED",	
"TRANS_SIGNATURE_CONFIRMED",	
"TRANS_LASTPRINT",	

"TRANS_END",	
"TRANS_END_CARD",	
"TRANS_END_TRANS",	
"TRANS_END_ADMIN",	// 30
"TRANS_END_ADMIN_LOCK",	
"EXT_CARD",	
"EXT_CARD_TRANS",	
"EXT_CARD_WAIT_ACK",	
"EXT_WAIT_HOSTDATA",	
"EXT_CARD_COMMIT",	
"EXT_TRANS_END",	
"EXT_CARD_COMMIT_NOECR",	
"EXT_CARD_COMMIT_CONFIRM",	
"EXTTRANS_END_CARD",	// 40
"EXTTRANS_END_CARD_ADMIN",	
"EXTTRANS_END_CARD_ADMINLOCK",	
"ADMIN",	
"ADMIN_PRINT",	
"ADMIN_RECEIPT_PRINTED",	
"ADMIN_PRINT_FIRST_RECEIPT",	
"ADMIN_PRINT_SECOND_RECEIPT",	
"TRANS_ECRCOMERR",	
"TRANS_ECRCOMERR_AMOUNT",	
"ECRCOMERR",	// 50
"ADMIN_ERROR",	
"ADMIN_PRINT_PREV",	
"ADMIN_PREV_PRINTED",	
"ADMIN_NP",	
"ADMIN_PRINT_NP",	
"ADMIN_RECEIPT_PRINTED_NP",	
"ADMIN_PRINT_PREV_NP",	
"ADMIN_PREV_PRINTED_NP",	
"ADMIN_REBOOT",	
"OPEN_MENU",	// 60
"TRANS_MENU",	
"TRANS_OPEN",	
"SYSTEM_ERROR",	
"TRANS_ABORT_TIMEOUT",	
"STOPLIST",	
"TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK",	
"TRANS_KEY",	
"TRANS_SCAN",	

## 2.A.15 Description of locked states

State name	State	Description
IDLE_NP	4	The terminal is not connected and a receipt is remaining in the terminal
CONNECTED_NP	6	The terminal is connected and a receipt is remaining in the terminal
OPEN_NP	8	The terminal is ready but a receipt is remaining in the terminal
ADMIN_NP	54	The terminal is running an administrative function while a receipt is remaining in the terminal
ADMIN_PRINT_NP	55	The terminal is running an administrative function which prints a receipt while another receipt is remaining in the terminal
ADMIN_RECEIPT_PRINTED_NP	56	The terminal is running an administrative function, which prints the last receipt in a series, but another receipt is still remaining in the terminal
ADMIN_PRINT_PREV_NP	57	The terminal is running an administrative function which prints a receipt while another receipt is remaining in the terminal
ADMIN_PREV_PRINTED_NP	58	The terminal is running an administrative function, which prints a receipt. The receipt is acknowledged, but another receipt is still remaining in the terminal

The terminal will not be locked in “no receipt state” from SW version 3.3.

## 2.A.16 The Merchant events and number

INT_DEFAULT	// 2
INT_STATUS	
INT_TRANSACTION_STATE_INFO	
INT_NORECEIPT	
INT_TIMEOUT	
INT_ADMIN_PRINT_END	
INT_CONFIRM	
INT_NOCONFIRM	
INT_DISCONNECT	// 10
INT_TIMER_ABORT	
INT_OPERATOR_ABORT	
INT_USER_ABORT	
INT_ABORT	
INT_TIMEOUT_RECEIPT	
INT_TIMEOUT_COMPLETE	
INT_TIMEOUT_AMOUNT	
INT_TIMEOUT_LOCALCARD	
INT_FIRST_RECEIPT	
INT_SECOND_RECEIPT	// 20
INT_ADMIN_ANOTHER_RECEIPT	
INT_REBOOT	
INT_ECR_COM_ERR	



ECR events	
ECR_CONNECT	
ECR_CONNECT_OK	
ECR_OPEN	
ECR_DISCONNECT	
ECR_CLOSE	
ECR_TRANSACTION	
ECR_TRANSACTION_COMPLETED	// 30
ECR_ADMIN	
ECR_IDLE	
ECR_RECEIPT	
ECR_LAST_RECEIPT	
ECR_RECEIPT_VERIFY	
ECR_RECEIPT_ACKNOWLEDGED	
ECR_AMOUNT	
ECR_VERSIG_ACKNOWLEDGED	
ECR_ABORT	
ECR_INFO	// 40
ECR_ERROR	
TAPA events	
TAPA_TRANSACTION_STATE_INFO	
TAPA_TRANS_VER_SIGNATURE	
TAPA_OPEN_HANDLER	
TAPA_CLOSE_HANDLER	
TAPA_READ_HANDLER_STRING	
TAPA_WRITE_HANDLER_STRING	
TAPA_DISPLAY_MESSAGE	
TAPA_TRANSACTION_COMPLETE	
TAPA_GET_AMOUNT	// 50

TAPA_UNKNOWN	
TAPA_CHECK_STOP_LIST	
Signal events	
SIG_GET_AMOUNT	
SIG_PRINT	
SIG_GET_STATE	
SIG_PRINTER_STATUS	
SIG_USER_STOP	
SIG_TRANSACTION_STATE	
SIG_TRANS_COMMIT	
SIG_ADVICE_FLAG	// 60
SIG_CARDDATA	
SIG_ASW_CODE	
SIG_EXTERNAL_CARDDATA	
SIG_EXTERNAL_CARDNAME	
SIG_EXTERNAL_HOSTDATA	
SIG_IDLE	
SIG_EJECTCARD	
SIG_TERM_INITIALISED	
SIG_PRERESULT	
Other events	
ECR_PREV_RECEIPT	// 70
ECR_PREV_RECEIPT_LAST	
ECR_UNLOCK_RECEIPT	
ECR_LOCK_RECEIPT	
ECR_CLOSE_RECEIPT	
ECR_EXTCARD	
ECR_EXTCARD_CONFIRM	
ECR_EXTCARD_HOSTRSP	

SIG_LOCKED	
INT_CARD_AMOUNT2	
INT_RECEIPT_ACK	// 80
INT_RECEIPT_NAK	
INT_TAPA_WRITESTR	
TAPA_GET_AMOUNT_NOPAN	
ECR_MENU	
ECR_MENU_REPLY	
ECR_FEE	
ECR_DUMMY1	
SIG_INIT	
SYSTEM_ERROR	
SIG_ENAI_SEND	// 90
ECR_ENAI_RECEIVE	
SIG_ENAI_CONNECT	
SIG_ENAI_DISCONNECT	
TAPA_GET_MERCHANT_DATA	
ECR_CHECK_STOP_LIST	
ECR_CHECK_STOP_LIST_OK	
INT_TIMEOUT_STOPLIST	
ECR_PRERESULT	
ECR_AMOUNT_DCC	
SIG_PRINT_ACK	// 100
ECR_AMOUNT_USER	
ECR_ABORT_APE_DAPE_ERROR	

### 2.A.17 The Merchant state–event diagram

A pdf file of the current state–event diagram is available upon request.

The state diagram shows all the states available the merchant module in the terminal.

Event colour codes:

- BLACK    Static events – these events are only handled in that particular state.
- GREEN    Dynamic events – these events can be handled in the current state or in the hierarchical 'parent' state if the event isn't available in the current state.
- RED       Hierarchy – these red lines shows the hierarchical structure of the state machine.

### 2.A.18 The Merchant state–events

From state	Event	To state	
INIT,2	SIG_TERM_INITIALISED	IDLE,2	STATIC
IDLE,2	ECR_CONNECT_OK	CONNECTED,5	STATIC
CONNECTED,5	ECR_DISCONNECT	IDLE,2	STATIC
CONNECTED,5	ECR_OPEN	OPEN,7	STATIC
OPEN,7	ECR_ADMIN	ADMIN,43	STATIC
OPEN,7	ECR_CLOSE	CONNECTED,5	STATIC
OPEN,7	ECR_MENU	OPEN_MENU,60	STATIC
OPEN,7	SIG_CARDDATA	CARD,11	STATIC
OPEN,7	SIG_INIT	TRANS_OPEN,62	STATIC
OPEN,7	TAPA_CHECK_STOP_LIST	STOPLIST,65	DYNAMIC
OPEN,7	TAPA_GET_AMOUNT	CARD_REQ_AMOUNT,14	STATIC
OPEN,7	TRANS_START_KEY	TRANS_KEY,67	DYNAMIC
TRANS,9	ECR_AMOUNT	AMOUNT_READY,17	STATIC
TRANS,9	ECR_MENU	TRANS_MENU,61	STATIC
TRANS,9	ECR_RECEIPT_ACKNOWLEDGED	TRANS_WAIT_ACK,21	STATIC
TRANS,9	INT_ECR_COM_ERR	TRANS_ECRCOMERR,48	DYNAMIC
TRANS,9	SIG_EXTERNAL_CARDDATA	EXT_CARD_TRANS,33	STATIC
TRANS,9	SIG_TRANS_COMMIT	TRANS_COMMIT,19	DYNAMIC
TRANS,9	TAPA_GET_AMOUNT	TRANS_REQ_AMOUNT,15	STATIC
TRANS,9	TAPA_TRANSACTION_COMPLETE	TRANS_WAIT_ACK,21	DYNAMIC
CARD,11	ECR_TRANSACTION	TRANS,9	STATIC
CARD,11	TAPA_GET_AMOUNT	CARD_REQ_AMOUNT,15	STATIC
CARD,11	TAPA_TRANSACTION_COMPLETE	OPEN,7	DYNAMIC
CARD_REQ_AMOUNT,14	ECR_CONNECT	TRANS_REQ_AMOUNT,15	STATIC
CARD_REQ_AMOUNT,14	ECR_TRANSACTION	TRANS_REQ_AMOUNT,15	STATIC
CARD_REQ_AMOUNT,14	INT_ABORT	OPEN,7	STATIC
TRANS_REQ_AMOUNT,15	ECR_AMOUNT_DCC	TRANS_MENU,61	STATIC
TRANS_REQ_AMOUNT,15	ECR_AMOUNT_USER	TRANS_MENU,61	STATIC
TRANS_REQ_AMOUNT,15	INT_ECR_COM_ERR	TRANS_ECRCOMERR_AMOUNT,49	DYNAMIC
TRANS_REQ_AMOUNT,15	TAPA_GET_AMOUNT	TRANS_REQ_AMOUNT,15	STATIC
TRANS_COMMIT,19	ECR_RECEIPT	PRINT_WAIT_ACK,21	STATIC
TRANS_COMMIT,19	ECR_RECEIPT_VERIFY	PRINT_WAIT_ACK,21	STATIC

TRANS_COMMIT,19	SIG_PRERESULT	TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK,66	STATIC
PRINT_WAIT_ACK,21	ECR_RECEIPT_ACKNOWLEDGED	TRANS_COMMIT,19	STATIC
PRINT_WAIT_ACK,21	ECR_RECEIPT_ACKNOWLEDGED	TRANS_COMMIT,19	STATIC
PRINT_WAIT_ACK,21	SIG_PRERESULT	TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK,66	STATIC
PRINT_WAIT_ACK,21	TAPA_TRANS_VER_SIGNATURE	PRINT_WAIT_ACK_VERIFY,22	STATIC
PRINT_WAIT_ACK_VERIFY,22	ECR_RECEIPT_ACKNOWLEDGED	TRANS_COMMIT,19	STATIC
PRINT_WAIT_ACK_VERIFY,22	INT_TIMEOUT_RECEIPT	TRANS_COMMIT,19	STATIC
TRANS_WAIT_ACK,21	INT_RECEIPT_ACK	TRANS_END,27	STATIC
TRANS_END,27	ECR_TRANSACTION	TRANS_END_TRANS,29	STATIC
TRANS_END,27	SIG_IDLE	OPEN,7	STATIC
TRANS_END_TRANS,29	SIG_IDLE	TRANS,9	STATIC
EXT_CARD,32	ECR_EXTCARD_CONFIRM	EXT_CARD_COMMIT_CONFIRM,39	DYNAMIC
EXT_CARD,32	ECR_EXTCARD	EXT_CARD_COMMIT,36	DYNAMIC
EXT_CARD,32	ECR_TRANSACTION	EXT_CARD_TRANS,33	STATIC
EXT_CARD,32	TAPA_TRANSACTION_COMPLETE	OPEN,7	DYNAMIC
EXT_CARD_TRANS,33	TAPA_TRANSACTION_COMPLETE	EXT_CARD_WAIT_ACK,34	DYNAMIC
EXT_CARD_WAIT_ACK,34	ECR_RECEIPT_ACKNOWLEDGED	OPEN,7	STATIC
EXT_CARD_WAIT_ACK,34	INT_TIMEOUT_RECEIPT	OPEN,7	STATIC
EXT_WAIT_HOSTDATA,35	ECR_EXTCARD_HOSTRSP	EXT_TRANS_END,37	STATIC
EXT_WAIT_HOSTDATA,35	INT_ECR_COM_ERR	EXT_TRANS_END,37	STATIC
EXT_CARD_COMMIT,36	SIG_EXTERNAL_HOSTDATA	EXT_WAIT_HOSTDATA,35	STATIC
EXT_CARD_COMMIT_NOECR,38	TAPA_TRANSACTION_COMPLETE	OPEN,7	DYNAMIC
EXT_CARD_COMMIT_CONFIRM,39	SIG_EXTERNAL_HOSTDATA	EXT_WAIT_HOSTDATA,35	STATIC
ADMIN,43	ECR_LAST_RECEIPT	ADMIN_RECEIPT_PRINTED,46	STATIC
ADMIN,43	ECR_PREV_RECEIPT	ADMIN_PRINT_PREV,52	STATIC
ADMIN,43	ECR_PREV_RECEIPT_LAST	ADMIN_PREV_PRINTED,53	STATIC
ADMIN,43	ECR_PREV_RECEIPT_LAST	ADMIN_PREV_PRINTED,53	STATIC
ADMIN,43	ECR_RECEIPT	ADMIN_PRINT,44	STATIC
ADMIN,43	ECR_TRANSACTION_COMPLETED	OPEN,7	DYNAMIC
ADMIN,43	INT_REBOOT	ADMIN_REBOOT,59	STATIC
ADMIN_PRINT,44	ECR_LAST_RECEIPT	ADMIN_RECEIPT_PRINTED,45	STATIC
ADMIN_PRINT,44	INT_ECR_COM_ERR	ADMIN_ERROR,51	DYNAMIC
ADMIN_PRINT,44	INT_TIMEOUT_RECEIPT	ADMIN_ERROR,51	DYNAMIC
ADMIN_RECEIPT_PRINTED,45	ECR_RECEIPT_ACKNOWLEDGED	ADMIN,43	STATIC

TRANS_ECRCOMERR,48	TAPA_TRANSACTION_COMPLETE	ECRCOMERR,50	DYNAMIC
ECRCOMERR,50	ECR_CONNECT_OK	CONNECTED,5	STATIC
ADMIN_ERROR,51	ECR_TRANSACTION_COMPLETED	OPEN,7	STATIC
ADMIN_PRINT_PREV,52	ECR_RECEIPT_ACKNOWLEDGED	ADMIN_PREV_PRINTED,53	STATIC
ADMIN_PREV_PRINTED,53	ECR_RECEIPT_ACKNOWLEDGED	ADMIN,43	STATIC
OPEN_MENU,60	ECR_TRANSACTION	TRANS_MENU,61	STATIC
OPEN_MENU,60	INT_ABORT	OPEN,7	STATIC
OPEN_MENU,60	INT_TIMEOUT_LOCALCARD	OPEN,7	STATIC
OPEN_MENU,60	SIG_IDLE	OPEN,7	STATIC
TRANS_MENU,61	ECR_MENU_REPLY	TRANS,9	STATIC
TRANS_MENU,61	INT_TIMEOUT_LOCALCARD	TRANS,9	STATIC
TRANS_MENU,61	TAPA_GET_AMOUNT	TRANS_REQ_AMOUNT,15	STATIC
TRANS_OPEN,62	ECR_MENU	OPEN_MENU,60	STATIC
TRANS_OPEN,62	ECR_TRANSACTION	TRANS,9	STATIC
TRANS_OPEN,62	INT_ABORT	OPEN,7	STATIC
TRANS_OPEN,62	SIG_EXTERNAL_CARDDATA	EXT_CARD,32	STATIC
TRANS_OPEN,62	TAPA_GET_AMOUNT	CARD_REQ_AMOUNT,14	STATIC
TRANS_OPEN,62	TAPA_TRANSACTION_COMPLETE	OPEN,7	DYNAMIC
SYSTEM_ERROR,63	ECR_ADMIN	ADMIN,43	STATIC
STOPLIST,65	TAPA_CHECK_STOP_LIST	TRANS_COMMIT,19	STATIC
TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK,66	ECR_RECEIPT_ACKNOWLEDGED	TRANS_COMMIT,19	DYNAMIC
TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK,66	INT_TIMEOUT_RECEIPT	TRANS_COMMIT,19	STATIC
TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK,66	SIG_PRINT	PRINT_WAIT_ACK,21	DYNAMIC
TRANS_DELAY_PRERES_UNTIL_RECEIPT_ACK,66	TAPA_TRANSACTION_COMPLETE	TRANS_WAIT_ACK,21	DYNAMIC

Parent state	Child state	
INIT,2	IDLE,3	Hierarchy
IDLE,3	CONNECTED,5	Hierarchy
IDLE,3	SYSTEM_ERROR,63	Hierarchy
CONNECTED,5	OPEN,7	Hierarchy
OPEN,7	TRANS,9	Hierarchy
OPEN,7	TRANS_OPEN	Hierarchy
OPEN,7	CARD.9	Hierarchy
TRANS,9	TRANS_REQ_AMOUNT,15	Hierarchy
TRANS,9	AMOUNT_READY,17	Hierarchy
AMOUNT_READY,17	TRANS_COMMIT,19	Hierarchy
OPEN,7	OPEN_MENU,60	Hierarchy
TRANS,9	TRANS_MENU,61	Hierarchy
TRANS,9	TRANS_KEY	Hierarchy
TRANS_COMMIT,19	PRINT_WAIT_ACK,23	Hierarchy
PRINT_WAIT_ACK,23	PRINT_WAIT_ACK_VERIFY,22	Hierarchy
TRANS_COMMIT	TRANS_WAIT_ACK,21	Hierarchy
CARD,11	CARD_REQ_AMOUNT,14	Hierarchy
TRANS_ECRCOMERR,48	TRANS_ECRCOMERR_AMOUNT,49	Hierarchy
TRANS_ECRCOMERR,48	ECRCOMERR,50	Hierarchy
ADMIN,43	ADMIN_ERROR,51	Hierarchy
ADMIN,43	ADMIN_PRINT,44	Hierarchy
ADMIN_PRINT,44	ADMIN_RECEIPT_PRINTED,45	Hierarchy
ADMIN,43	ADMIN_PRINT_PREV,52	Hierarchy
ADMIN_PRINT_PREV,52	ADMIN_PREV_PRINTED,53	Hierarchy
EXT_CARD,32	EXT_CARD_TRANS,33	Hierarchy
EXT_CARD_TRANS,33	EXT_CARD_COMMIT_CONFIRM,39	Hierarchy
EXT_CARD,32	EXT_CARD_COMMIT_NOECR,38	Hierarchy
EXT_CARD_TRANS,33	EXT_CARD_COMMIT,36	Hierarchy
EXT_CARD_COMMIT,36	EXT_WAIT_HOSTDATA,35	Hierarchy
EXT_WAIT_HOSTDATA,35	EXT_TRANS_END,37	Hierarchy
EXT_TRANS_END,37	EXT_CARD_WAIT_ACK,34	Hierarchy

## 2.B Drop IP

ECR connected via IP can now drop the ip connection or reboot terminal on another port.

At boot the terminal listen for a new connection from the ECR, RS232/USB or IP. If an IP connection is made, terminal will begin listen on port 2001 for a command, to either drop the IP connection or to reboot.

The command can be send via telnet but it have to be completed within 10 sec, to be a success, and only if the right data has been sent.

Where XXXXXX is the terminal identification

To drop the IP connection send

DropIp:00XXXXXX

To reboot the terminal send

Reboot:00XXXXXX

If the command is accepted by the terminal, it will respond with:

Dropping IP port 2000 on terminal 00XXXXXX

or

Restarts terminal 00XXXXXX - Wait 1-2 min.

The flxdrv.dll has a new function to send the above commands

Function:

```
int flxDropIp(char *ip_addr, char *terminal_ident, int to_do);
```

Tell terminal to drop IP connection, making it ready for a new connect:

Input: ip\_addr Terminals IP address

ex. 10.0.0.156

terminal\_ident Terminal identifier

ex. 00990768 Must be 8 digits

to\_do Action terminal has to perform

0: DropIp Dropping terminals IP connection

1: Reset Restarts terminal

Output: 1 - SUCCESS - OK

0 - FAILURE - not OK

```
int flxDropIp(char *ip_addr, char *terminal_ident, int to_do)
{
    int mysocket;
    struct sockaddr_in dest;
    char wzRec[256];
    char searchResult[256];
    int nLeft = 256;
    int iPos = 0;
    int nData = 0;
    int res = 0;
#ifdef _WIN32
    uint NonBlock = 0;
#endif
    fd_set Reader ;
    int n ;
    struct timeval tv ;
```



```

int iLoopCnt = 0;

mysocket = socket(AF_INET, SOCK_STREAM, 0);

memset(&dest, 0, sizeof(dest)); /* zero the struct */
dest.sin_family = AF_INET;
dest.sin_addr.s_addr = inet_addr(ip_addr);
    /* set destination IP number */
dest.sin_port = htons(2001);
    /* set destination port number */

if (connect(mysocket, (struct sockaddr *)&dest,
    sizeof(struct sockaddr)) == SOCKET_ERROR)
{
    _trace2File("Error connect %s 2001 %d - %s",
        ip_addr, errno, strerror(errno)) ;
    return 0;
}
    _trace2File("Connect %s 2001 %d - %s", ip_addr,
        errno, strerror(errno));

/* Change the socket mode on the listening socket
    from blocking to non-block */
#ifdef _WIN32
    NonBlock = 1;
    if (ioctlsocket(mysocket, FIONBIO, &NonBlock) == SOCKET_ERROR)
    {
        _trace2File("ioctlsocket() failed - %d - %s", errno,
            strerror(errno));
        return 0;
    }
#else
    fcntl(mysocket, F_SETFL, fcntl(mysocket, F_GETFL, 0) | O_NONBLOCK);
#endif
    // process data

    if (to_do == 0)
    {
        sprintf(wzRec, "DropIp:%s\n", terminal_ident);
        sprintf(searchResult, "Dropping IP port 2000 on
            terminal %8.8s\n", terminal_ident);
    }
    else
    {
        sprintf(wzRec, "Reboot:%s\n", terminal_ident);
        sprintf(searchResult, "Restarts terminal %8.8s
            - Wait 1-2 min.\n", terminal_ident);
    }
    nLeft = strlen(wzRec);
    do
    {
        nData = send( mysocket, &wzRec[iPos], nLeft, 0 );
        if( nData == SOCKET_ERROR ) {
            _trace2File("Error sending data %d - %s", errno,

```

```

        strerror(errno)) ;
    break;
}
nLeft -= nData;
iPos += nData;
} while( nLeft > 0 );

// Set up the file descriptor set.
FD_ZERO(&Reader) ;
FD_SET(mysocket, &Reader) ;

// Set up the struct timeval for the timeout.
tv.tv_sec = 10 ;
tv.tv_usec = 0 ;

// Wait until timeout or data received.
_trace2File("Before select..");
n = select ( mysocket+1, &Reader, NULL, NULL, &tv ) ;
if ( n == 0 )
{
    _trace2File("Timeout..");
    return 0;
}
else if( n == -1 )
{
    _trace2File("Error..");
    return 0;
}
_trace2File("After select..n=%d FD_ISSET(mysocket, &Reader)=%d",
    n, FD_ISSET(mysocket, &Reader));
memset( &wzRec, 0, sizeof( wzRec ) );
nLeft = strlen(searchResult);
iPos = 0;
if (FD_ISSET(mysocket, &Reader))
{
    do
    {
        nData = recv( mysocket, &wzRec[iPos], nLeft, 0 );
        if( nData == SOCKET_ERROR ) {
            _trace2File("Error receiving data %d - %s",
                errno, strerror(errno)) ;
            break;
        }
        nLeft -= nData;
        iPos += nData;
    } while((nLeft > 0) && (iLoopCnt++ < 10));
}
_trace2File("Data Received len=%d", iPos);
if (iPos > 0)
{
    if (strncmp(wzRec, searchResult, strlen(searchResult)) == 0)
    {
        _trace2File("Received expected result from terminal");
        if (pKiss)

```

```

    {
        pSleep(CONNECT_SLEEP);
        PKISS_Delete(&pKiss);
    }
    res = 1;
}

}
shutdown(mysocket, 2); // SD_BOTH
closesocket(mysocket);
return res;
}

```

## 2.C How to interpret a Network Report

On terminals with software version 2.2.01 or newer a Network Report can be made. The report can be found in test menu – 7 – 6 or the title “NETVÆRK”.

The network type is written in the terminal, but if the type is Ethernet a report is also written on the printer.

### Network Report

2007-01-22

14:52

### Explanation

TERM:	00990067	The terminals id
DOMAIN:		If the terminal belongs to a domain, part of point.local network, the name is written here
DNS1:	192.168.0.200	Domain Name Server 1
DNS2:	192.168.0.204	Domain Name Server 2
IP:	192.168.0.56	The IP address of the terminal
SUBNET:	255.255.255.0	The terminal under a subnet
GATEWAY:	192.168.0.1	If IP address can't be found then ask here
DHCP:	192.168.0.204	Were shall the terminal ask to get its address

### PING TEST

DNS1

Ping 192.168.0.200 – SUCCESS

A DNS shall not reply on ping, but often does

DNS2

Ping 192.168.0.204 – SUCCESS

GATEWAY

Ping 192.168.0.1 – SUCCESS

A DNS shall not reply on ping, but often does

### DNS lookup TEST

Lookup on Domain Name Server to check if it recognizes the IP addresses we need to make:  
Transactions,  
Send logfiles,  
Download parameter,  
Download TLCMDB,  
Download programs etc.

DNS1: 192.168.0.200

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : param.point-ts.dk  
IP : 80.164.132.228

Used by sendlog, download param, TLCMDB

Host to find : time.point-ts.dk  
IP : 80.164.132.227

Verifone Denmarks time server

Host to find : rtl.point-ts.dk  
IP : 80.164.132.227

Used by download program

DNS2: 192.168.0.204

Same as above on Domain Name Server 2

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : test.point-ts.dk  
IP : 80.164.132.229

Host to find : param.point-ts.dk  
IP : 80.164.132.228

Host to find : time.point-ts.dk  
IP : 80.164.132.227

Host to find : rtl.point-ts.dk  
IP : 80.164.132.227

## **TLCMDB:**

[100]

NAME: test.point-ts.dk

PORT: 22000

RECV ERROR -3

Err Timeout

Connect [100] – FAILED

[101]

NAME: test.point-ts.dk

PORT: 22000

RECV ERROR -3

Err Timeout

Connect [101] – FAILED

[120]

NAME: test.point-ts.dk

PORT: 22000

Date: 20070122 14:53:03

Connect [120] – SUCCESS

[121]

NAME: test.point-ts.dk

PORT: 22000

RECV ERROR -3

Err Timeout

Connect [121] – FAILED

[181]

NAME: param.point-ts.dk

PORT: 24000

[182]

NAME: time.point-ts.dk

PORT: 13

2007-01-22 14:52

## **Contents of TLCMDB entries**

Who is contacted when TLCMDB entry 100 is chosen

Which port is chosen

Status on the get date/time that the terminal has made for entry 100

A timeout error has occurred

Date and time is shown when connection is made

It is not possible to make a test on entry 181 and 182

Verifone Denmarks time server

## 3 | Spin Connect

### 3.1 Preface

This document describes how to interface to the Flex Terminal Vx680/Vx690. The approach described here is the “Spin Connect”.

Compared to our ‘normal integrations’ the logic is turned around so now the Vx680/Vx690 is the client and the ECR is the server.

Before making integration to your ECR you must sign a development agreement with Verifone Denmark.

The certifying authority, Nets, must certify an ECR implementation. More information of the certifying process is found on Nets website.

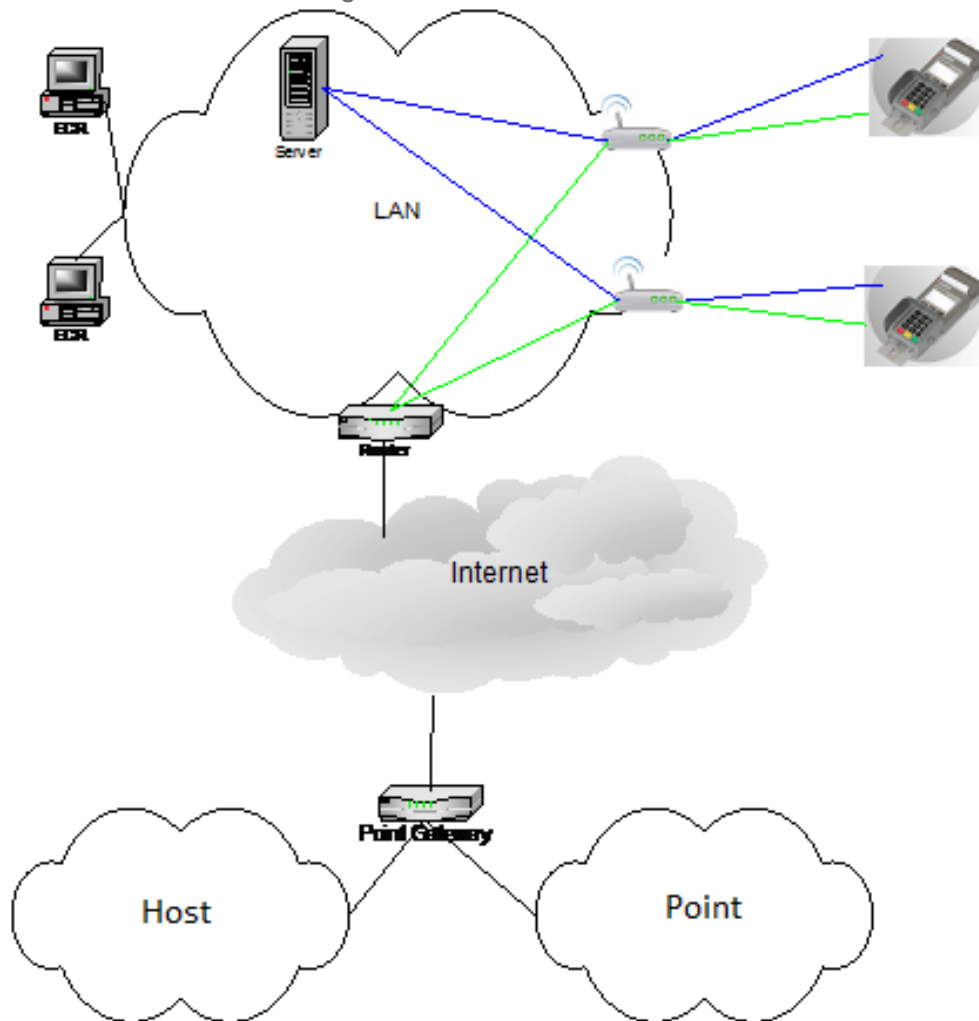
Verifone Denmark will not deliver any software for ECR/Server. Revision:

- Version: 1.0.00 First release.
- Version: 2.0.00 Second release.
- Version: 3.0.00 New layout. More functional descriptions.
- Version: 3.0.01 More functional descriptions
- Version: 3.0.02 Removal of clerk id in “Hent Regning”.
- Version: 3.0.03 Add support for VAT and cancellation.
- Version: 3.0.05 Add support for MSC Loyalty Card handling.
- Version: 3.0.05 Add support for Cancellation from the server.
- Version: 3.0.06 Added a response to “Check Result”. Translated message and field names to English. Added the possibility for the waiter/clerk to select a check/bill from a list returned by the server.
- Version: 3.0.07 Added new configuration options. Changed format of 102 message. Added possibility to respond with a 102 message to 001 message Added PCI masked PAN to 002 message.
- Version: 3.0.08 Added SSL support. Added support for Cash refund transactions.
- Version: 3.0.09 Added Message 004 and 401. Added Transaction Condition Code, PSAM ID and STAN to message 002.
- Version: 3.0.10 Added text which tells which terminal version is compatible with this specification.
- Version: 3.0.11 <CLERK\_ID> was added to message 002.
- Version: 3.2.00 Added support for RS232 and the “one-to-one” scenario. Added new admin function message 103.
- Version: 3.3.00 Clarified that it is the server that closes the connection.
- Version: 3.3.01 Listed configuration that is tested by Verifone.
- Version: 3.5.03 New Storebox (ekvittering) message 002 function added.
- Version: 3.6.00 Pay with cash in SpinConnect now uses the entered amount.
- Version: 3.7.xx Added support for VAS.

## 3.2 Hardware Overview

The terminal and ECR/Server can communicate using TCP/IP or RS232.

When communicating by TCP/IP, no RS232 cable can be used, hence the ECR and terminal communicate by the local network. Encryption can be used on the communication between the terminal and the server when using TCP/IP.



The illustration above shows an example of a normal setup using TCP/IP. Only 2 terminals and ECR's are shown, but in principle there is no limit to the number of terminals on the network. Commands and responses are blue on the drawing. The transaction to Nets is green in the illustration.

### 3.2.1 The Terminal

The terminal can be configured with 2 different communication modules. TCP/IP is only supported with the WLAN module.



<b>Communication module</b>	<b>RS232</b>	<b>TCP/IP</b>
WLAN	Supported	Supported
GSM/WLAN	Supported	Not supported

### 3.3 Software Overview

The software interface is based on simple commands and responses in ASCII format char 8 bit. This means if a field contains the number 0, the ASCII character '0' (decimal 48) is sent. '\0' is used to terminate a message (decimal 0).

This document describes all commands and responses. Finally, these commands and responses are wrapped for communication in the TCP/IP Protocol. The communication can be encrypted with SSL. Alternatively RS232 can be used. When using RS232, proper CRC checking and handling must be implemented on the server.

How the solution is implemented on the server/ECR is out of scope for this documentation.

## 3.4 Functionality

The functionality of the terminal is fulfilling the Nets OTRS specification. It is recommended for the ECR developer to read the merchant section in the OTRS specification. Please visit Nets website for more information.

Terminal functions and options are configurable through parameter download from Verifone. Observe that functions and options are configured to support the wanted ECR functions.

### 3.4.1 Initialization

All transactions start with entering waiter id (if enabled) and table or check id. A check can also be selected from a list of outstanding checks delivered from the server. During a transaction the waiter can change transaction conditions and amount (unless disabled).

### 3.4.2 Transaction Functionality

The functions below are used to make transactions.

#### 3.4.2.1 PIN Purchase Transactions

The terminal connects to the server and asks for 'Get Check' and receives 'Get Check Response'. The operator accepts the transaction conditions and amount and hands over the terminal to the user. The user inserts or swipes the card and enter the pin and hands over the terminal to the operator. Now the terminal connects to Nets and makes the transaction as a standalone terminal. After receiving the result of the transaction from Nets the terminal prints the receipt and connect to server and send 'Check Result'. The server responds with 'Check Result Response' which holds information about if the check is done, or if a new transaction on the same check should be started.

#### 3.4.2.2 Signature Purchase Transactions

The terminal connects to the server and asks for 'Get Check' and receives 'Get Check Response'. The operator accepts the transaction conditions and amount and hands over the terminal to the user. The user insert or swipe the card and hands over the terminal to the operator. Now the terminal connects to Nets and makes the transaction as a standalone terminal. After receiving the result of the transaction from Nets the terminal prints a receipt, the user must sign and the operator must verify the signature. The second receipt is then generated based on the operator's decision (signature accepted/rejected). Signature verification is controlled by the PSAM. The terminal connect to server and send 'Check Result'. The server responds with 'Check Result Response' which holds information about if the check is done, or if a new transaction on the same check should be started.

#### 3.4.2.3 Signature Refund Transactions

Refund transactions can be initiated after selecting/entering check/table ID. The terminal connects to the server and asks for 'Get Check' and receives 'Get Check Response'. The operator accepts

the transaction conditions and amount and hands over the terminal to the user. The user insert or swipe the card and hands over the terminal to the operator.

Now the terminal connects to Nets and makes the transaction as a standalone terminal. After receiving the result of the transaction from Nets the terminal prints a receipt, the operator must sign. The second receipt is then generated. The terminal connect to server and send 'Check Result'. The server responds with 'Check Result Response' which holds information about if the check is done, or if a new transaction on the same check should be started.

#### **3.4.2.4 Offline Refund Transactions**

The terminal connects to the server and asks for 'Get Check' and receives 'Get Check Response'. The operator enter the authorizations code accepts the transaction type and amount and hands over the terminal to the user. The transaction can now continue as a PIN or signature transaction.

#### **3.4.2.5 MSC Loyalty Card Transactions**

The terminal connects to the server and asks for 'Get Check' and receives 'Get Check Response'. The operator accepts the transaction conditions and amount and hands over the terminal to the user. The user swipe the card and hands over the terminal to the operator.

Now the terminal connects to server and sends 'MCS loyalty card'. The server process the transaction and send back 'MCS loyalty card response'. The terminal displays the result. The terminal connect to server and send 'Check Result'. The server responds with 'Check Result Response' which holds information about if the check is done, or if a new transaction on the same check should be started. With the 'Check Result Response' the server can send a receipt for the Loyalty card transaction, which will be printed on the terminal.

#### **3.4.2.6 Cancellation Transactions**

The terminal connects to the server and asks for 'Get Check' and receives 'Get Check Response'. The operator accepts the transaction conditions. This confirmation screen can be disabled, thus allowing for cancellation of last transaction without waiter intervention.

The terminal makes the transaction as a standalone terminal.

After receiving the result of the transaction from the PSAM the terminal prints the receipt and connect to server and send 'Check Result'. The server responds with 'Check Result Response' which holds information about if the check is done, or if a new transaction on the same check should be started. Cancellation cannot be used on cash transactions. Since the PSAM controls cancellation, it will always be done on the last card transaction, no matter if there has been a cash transaction in between.

#### **3.4.2.7 Cash Transactions**

The terminal connects to the server and asks for 'Get Check' and receives 'Get Check Response'. The operator accepts the transaction conditions and chooses cash. The operator is presented with a screen where it can be accepted or rejected that the money in cash is received. The terminal displays the result. The terminal connect to server and send 'Check Result'. The server responds

with 'Check Result Response' which holds information about if the check is done, or if a new transaction on the same check should be started. With the 'Check Result Response' the server can send a receipt for the cash transaction, which will be printed on the terminal.

#### **3.4.2.8 Cash Refund Transactions**

The terminal connects to the server and asks for 'Get Check' and receives 'Get Check Response'. The operator is presented with a screen where it can be accepted or rejected that the money in cash has been returned to the customer. The terminal displays the result. The terminal connect to server and send 'Check Result'. The server responds with 'Check Result Response' which holds information about if the check is done, or if a new transaction on the same check should be started. With the 'Check Result Response' the server can send a receipt for the cash refund transaction, which will be printed on the terminal. This transaction type can only be initiated from the server.

#### **3.4.2.9 Error Handling**

If the terminal cannot fetch the check from the server/ECR the clerk can start a transaction by selecting manual transaction and entering his clerk id. This functionality can be enabled or disabled. If the terminal cannot reach Nets the clerk can change the transaction type to offline. If the terminal cannot send the 'Check Result' to the server/ECR then the terminal will store it. The stored transaction results will send to server before fetching the next check. A list of stored transaction results can be printed, using "MENU – 5 – 11". Sending of stored transaction results can also be initiated by the operation by using "MENU – 4 – 13".

### **3.4.3 Administrative Functionality**

The administrative functions are used to all non transactions functionality or setup the terminal. This is only a part of all the administrative functions.

#### **3.4.3.1 The End of Day Functionality**

The end of day routine must be called at least one time every 24 hours. It is used to empty and balance the terminals. Data store against the transaction inquirer host system. It is also used for PSAM updates. A receipt is printed on the terminal. The end of day functionality can be initiated from the server by responding with message 'Admin Response (103)' or returning result code 23 in 'Check Result Response (201)'. The operation can also be initiated by the operator by using "MENU – 4 – 1".

#### **3.4.3.2 The Outstanding Result Functionality**

Sending of stored transaction results can be initiated by the operation by using "MENU – 4 – 13".

#### **3.4.3.3 The Terminal Report**

The terminal report provides vital information about the terminal configuration, e.g. SW and HW version, communication module etc. A receipt is printed on the terminal. The terminal report can be printed by using "MENU – 5 – 3". A terminal report will be sent to the server at startup in message 004.

#### **3.4.3.4 The Clock Synchronization**

The clock synchronization functionality is very useful when testing the terminal communication capabilities, testing the connection to Nets and , or testing the connection in case of communication errors during transactions. Synchronizing clock against Nets can be initiated by using "MENU – 4 – 7". Synchronizing clock against Verifone Denmark can be initiated by using "MENU – 4 – 2".

#### **3.4.4 Language Support**

The terminal supports different languages. Please contact Verifone Denmark for more specific information.

## 3.5 The Interface of Spin Connect

This section describes how to interface to the terminal.

There is no description of how the server/ECR shall implement this solution. 4 scenarios exist. 3 of the scenarios (fetch by check, fetch by table and fetch by list) can be used together. Which of the 3 scenarios is used is controlled by using the function buttons. The default prompt for the screen after waiter ID can be configured on the terminal through menus. The last scenario can only be used alone, and is used when there is a one-to-one relationship between the terminal and ECR and only one open check exists in one point of time (like a regular integration). Disabling one or more of the scenarios can be done through our terminal management system. This specification is implemented from version 3.2.04 of our terminal software.

### 3.5.1 Spin Connect Protocol Version

The current specification is using protocol version 1.03. When the terminal is booted it will try to send a message 004 to the server. This message informs the server of the current version, and the server can respond with a message 401 which can then indicate if the server is compatible with the terminal protocol version. If the 004 message cannot be delivered on boot, the terminal will make sure to deliver it before the first 001 message is sent to the server. If the server is written to handle new fields in the end of existing messages and new unknown messages by closing the connection with no response, it will be compatible with all versions sharing the same major version number. If the major number is different than in the protocol version that the server is written for, they are not compatible. Please note that field <CRC> field will always be the last, so this must be handled correctly when using RS232. All messages are terminated with a '\0'(decimal 0) character, so when reading a message data should be read until '\0' is received. After the server has sent the response it must close the connection.

### 3.5.2 Printing Receipts on the Terminal

Spin Connect gives the possibility for the server/ECR to print extra receipts on the terminal printer. If this functionality is used, it should be noted that the battery lifetime will be reduced drastically. The terminal can choose from 2 different fonts, one which can have 24 characters per line and one which can have 48 characters per line. If any of the lines in the receipt are over 24 characters the small font (48 characters per line) is selected, otherwise the normal font (24 characters per line) is selected.

### 3.5.3 Encrypting the Communication

The communication between the terminal and the server can be secured using SSL encryption. If this shall be enabled, the integrator must send the CA certificate to Verifone. The terminal does not use client certificate. The terminal will validate the server certificate and ensure that the server certificate is signed by the CA and that the Common Name (CN) is the same as the IP address of the server.

The terminal supports the cipher "AES256-SHA". It is up to the integrator to make sure that the latest requirements from PAN Nordic and PCI regarding key sizes are met.

Please note that you should create a new certificate for each server installation. All of these certificates should be signed by the same CA certificate, which is the certificate that you sent to Verifone.

### **3.5.3.1 Using RS232**

When using RS232 the following configuration for the serial port should be used:

- Baud rate: 4800, 9600, 19200, 38400, 57600 or 115200 (Can be configured on terminal).
- Data bits: 8
- Parity: None
- Stop bits: 1
- Flow Control: None

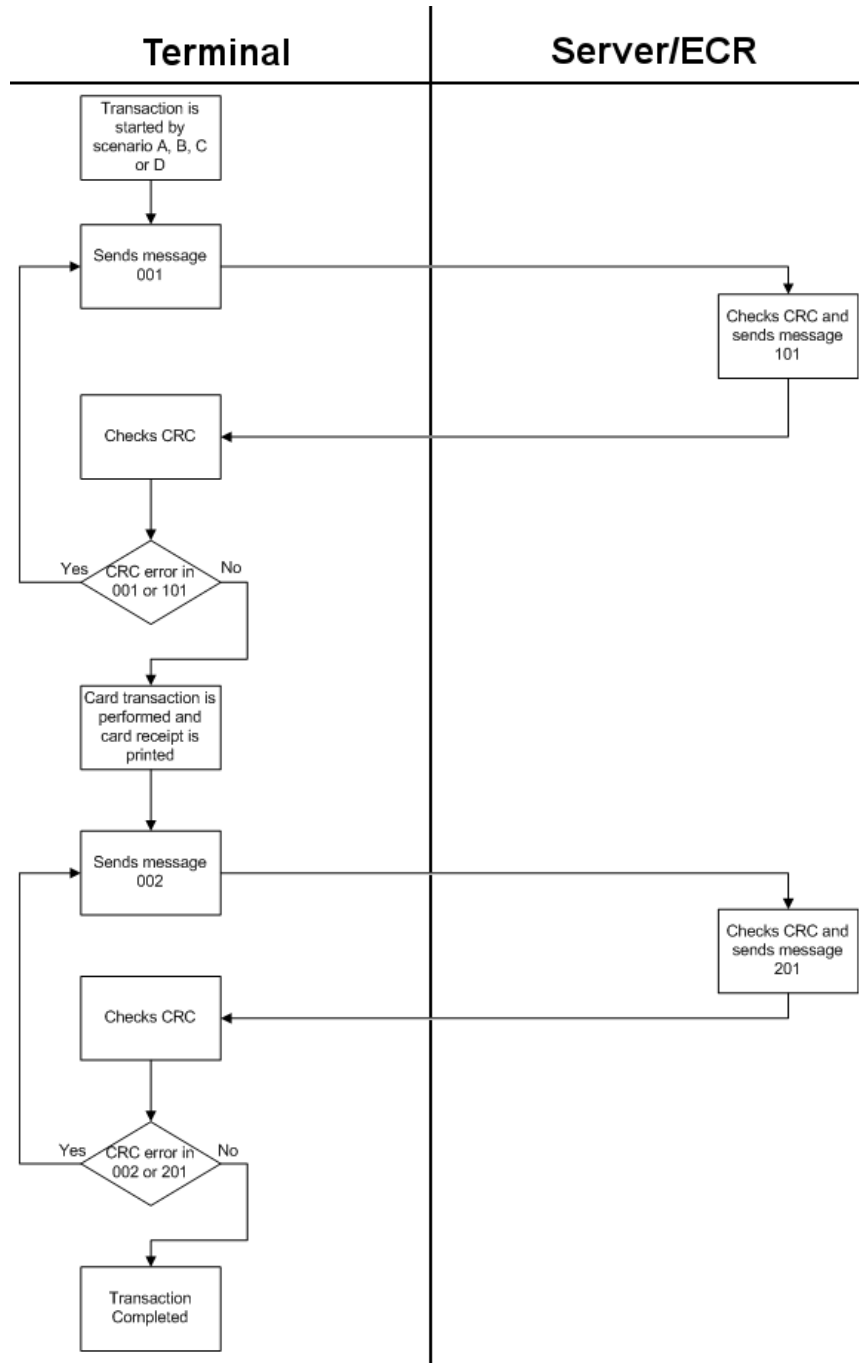
CRC should be calculated and checked for every message (see section below). When the server discovers a CRC error in a message received from the terminal the server should respond with a message with the result code set to '30'. For simplicity message 401 can always be used for this. The 401 message would be "401;0;30;". With an added CRC Checksum the message is "401;0;30;25001;".

On CRC error each message will be repeated up to 4 times. The time the terminal waits for a response on a message can be configured in "MENU – 6 – 6 – 2" on the terminal.

There is no inter-char timeout.

When using RS232 the flow is as illustrated here:





Even though not displayed in the above diagram, the same flow is used for the message pairs 003/301 and 004/401.

### 3.5.3.2 Calculating CRC

This is only relevant if RS232 is used for communication. If TCP/IP is used, just set <CRC> fields to '0'.

The polynomial  $x^{15} + x^{13} + 1$  is used for calculating CRC values. The CRC is calculated for the complete message except the last <CRC><DL>.

The following C code can be used to calculate the CRC value:

```
static const unsigned int crctp[] = {
    0x0, 0xc0c1, 0xc181, 0x140, 0xc301, 0x3c0, 0x280, 0xc241,
    0xc601, 0x6c0, 0x780, 0xc741, 0x500, 0xc5c1, 0xc481, 0x440,
    0xcc01, 0xcc0, 0xd80, 0xcd41, 0xf00, 0xcfc1, 0xce81, 0xe40,
    0xa00, 0xcac1, 0xcb81, 0xb40, 0xc901, 0x9c0, 0x880, 0xc841,
    0xd801, 0x18c0, 0x1980, 0xd941, 0x1b00, 0xdbc1, 0xda81, 0x1a40,
    0x1e00, 0xdec1, 0xdf81, 0x1f40, 0xdd01, 0x1dc0, 0x1c80, 0xdc41,
    0x1400, 0xd4c1, 0xd581, 0x1540, 0xd701, 0x17c0, 0x1680, 0xd641,
    0xd201, 0x12c0, 0x1380, 0xd341, 0x1100, 0xd1c1, 0xd081, 0x1040,
    0xf001, 0x30c0, 0x3180, 0xf141, 0x3300, 0xf3c1, 0xf281, 0x3240,
    0x3600, 0xf6c1, 0xf781, 0x3740, 0xf501, 0x35c0, 0x3480, 0xf441,
    0x3c00, 0xfcc1, 0xfd81, 0x3d40, 0xff01, 0x3fc0, 0x3e80, 0xfe41,
    0xfa01, 0x3ac0, 0x3b80, 0xfb41, 0x3900, 0xf9c1, 0xf881, 0x3840,
    0x2800, 0xe8c1, 0xe981, 0x2940, 0xeb01, 0x2bc0, 0x2a80, 0xea41,
    0xee01, 0x2ec0, 0x2f80, 0xef41, 0x2d00, 0xedc1, 0xec81, 0x2c40,
    0xe401, 0x24c0, 0x2580, 0xe541, 0x2700, 0xe7c1, 0xe681, 0x2640,
    0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0, 0x2080, 0xe041,
    0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281, 0x6240,
    0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441,
    0x6c00, 0xacc1, 0xad81, 0x6d40, 0xaf01, 0x6fc0, 0x6e80, 0xae41,
    0xaa01, 0x6ac0, 0x6b80, 0xab41, 0x6900, 0xa9c1, 0xa881, 0x6840,
    0x7800, 0xb8c1, 0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41,
    0xbe01, 0x7ec0, 0x7f80, 0xbf41, 0x7d00, 0xbdc1, 0xbc81, 0x7c40,
    0xb401, 0x74c0, 0x7580, 0xb541, 0x7700, 0xb7c1, 0xb681, 0x7640,
    0x7200, 0xb2c1, 0xb381, 0x7340, 0xb101, 0x71c0, 0x7080, 0xb041,
    0x5000, 0x90c1, 0x9181, 0x5140, 0x9301, 0x53c0, 0x5280, 0x9241,
    0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x9481, 0x5440,
    0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5f00, 0x9fc1, 0x9e81, 0x5e40,
    0x5a00, 0x9ac1, 0x9b81, 0x5b40, 0x9901, 0x99c1, 0x9880, 0x9841,
    0x8801, 0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x8a81, 0x4a40,
    0x4e00, 0x8ec1, 0x8f81, 0x4f40, 0x8d01, 0x4dc0, 0x4c80, 0x8c41,
    0x4400, 0x84c1, 0x8581, 0x4540, 0x8701, 0x47c0, 0x4680, 0x8641,
    0x8201, 0x42c0, 0x4380, 0x8341, 0x4100, 0x81c1, 0x8081, 0x4040
};

unsigned int CalcCRC(const unsigned char *buffer, size_t size)
{
    unsigned int i, cval, crc = 0;

    for (i = 0; i < size; i++)
    {
        cval = crc ^ (unsigned int)buffer[i];
        crc = (crc >> 8) ^ crctp[cval & 0xff];
    }
}
```

```
    return crc;  
}
```

## 3.6 Configuration Options

Through our terminal management system it is possible to configure the following:

- Enabling and disabling of the two different scenarios. At least one scenario should be enabled. Scenario A, B and C can be enabled at the same time.
- Enabling and disabling of clerk id prompt. If clerk id prompt is disabled, the server has the possibility to return a clerk id, which will be printed on the receipt.
- Enabling of the possibility to run transactions without a check. If enabled and used, result will still be sent to the server.
- Enabling of a "transaction condition lock". If this is enabled the merchant will not be able to change transaction conditions (e.g. amount).
- Disabling of confirmation on cancellation. This allows for cancellation of last transaction without user intervention.
- Enabling of SSL on the server connection.

In various menus on the terminal it is possible to configure the following:

- Server IP address
- Server IP port
- Server baud rate
- Server response timeout
- If both scenario A and scenario C is enabled, you can configure which scenario should be the default.

### 3.6.1 Tested configuration

Some combination of the configuration options that can be set in our terminal management system are tested by our QA department. These are the configurations that should be used. If you want to use another configuration, you should contact Verifone Denmark for an agreement about this.

The tested configurations are listed here:

0x01	Scenario A – Fetch by table
0x4F	Scenario A, B & C – Ask for clerk ID – No confirmation on cancellation

### 3.6.2 Specific Settings for Spin Connect

- **Server IP Setting:**

With this setting the merchant can setup the IP address that the server/ECR is listening on. Setting is accessed through "MENU – 6 – 6 – 1 – 1"

- **Server Port Setting:**

With this setting the merchant can setup the port on which the server/ECR is listening on. Setting is accessed through "MENU – 6 – 6 – 1 – 1"

- **Server Baud rate:**

With this setting the merchant can setup the baud rate at which the server/ECR is using. Setting is accessed through MENU – 6 – 6 – 1 – 2"

- **Timeout Setting:**

With this setting the merchant can setup how long the terminal should wait for a response from the server, before aborting or continuing.

Setting is accessed through “MENU – 6 – 6 – 2”

- **Default Initial Screen:**

With this setting the merchant get the possibility to choose if the first screen (after waiter id, if enabled), should be a prompt for check id or table id.

Setting is accessed through “MENU – 6 – 6 – 3”

- **Test Connection:**

This is a function which can test if it possible to connect server using the entered IP address and port settings.

Function is accessed through “MENU – 6 – 6 – 4”

## 3.7 Transaction Scenarios

In the following sections the flow of the 4 scenarios are showed. Loyalty card transactions are not showed in the diagrams. If loyalty card transactions are enabled and a loyalty card is used “MSC Loyalty Card” and “MSC Loyalty Card Response” messages are exchanged with the server instead of authorising with Nets. The rest of the flow is the same.

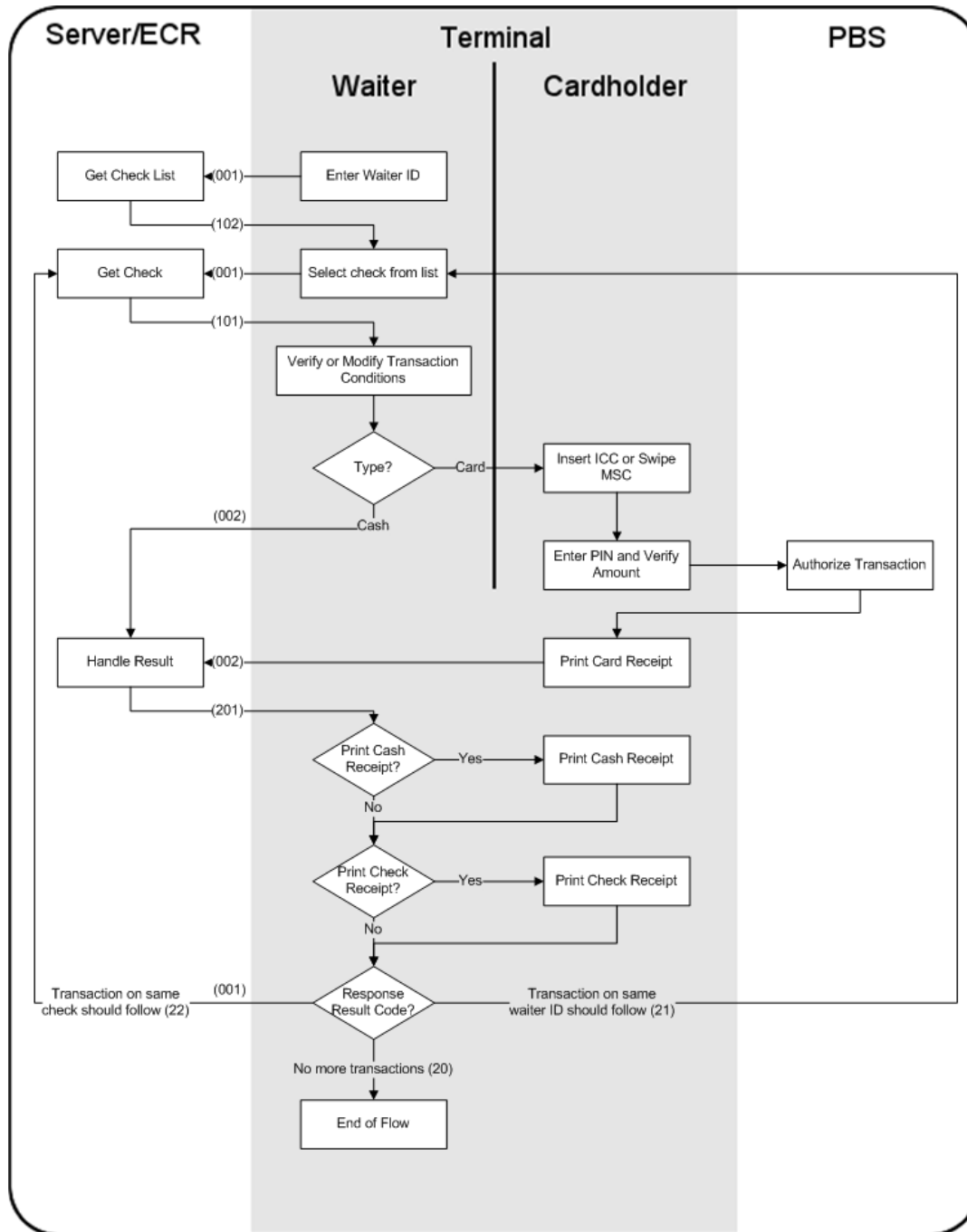
### 3.7.1 Scenario A (Fetch by Table)

1. A transaction starts with entry of the waiter ID (if enabled).
2. The waiter enters table ID.
3. Terminal fetches the Check from the server/ECR.
4. Terminal makes the transaction like a standalone, or nothing in case of cash transaction.
5. Terminal sends the result to the server/ECR.
6. The Server Responds with receipts which should be printed on the terminal and a result code which reflects if a new transaction should be started. The result code controls if a new transaction is started from step 2 or 3.



4. The waiter selects a check.
5. The terminal fetches the check from the server/ECR.
6. Terminal makes the transaction like a standalone, or nothing in case of cash transaction.
7. Terminal sends the result to the server/ECR.
8. The server responds with receipts which should be printed on the terminal and a result code which reflects if a new transaction should be started. The result code controls if a new transaction is started from step 2 or 5.

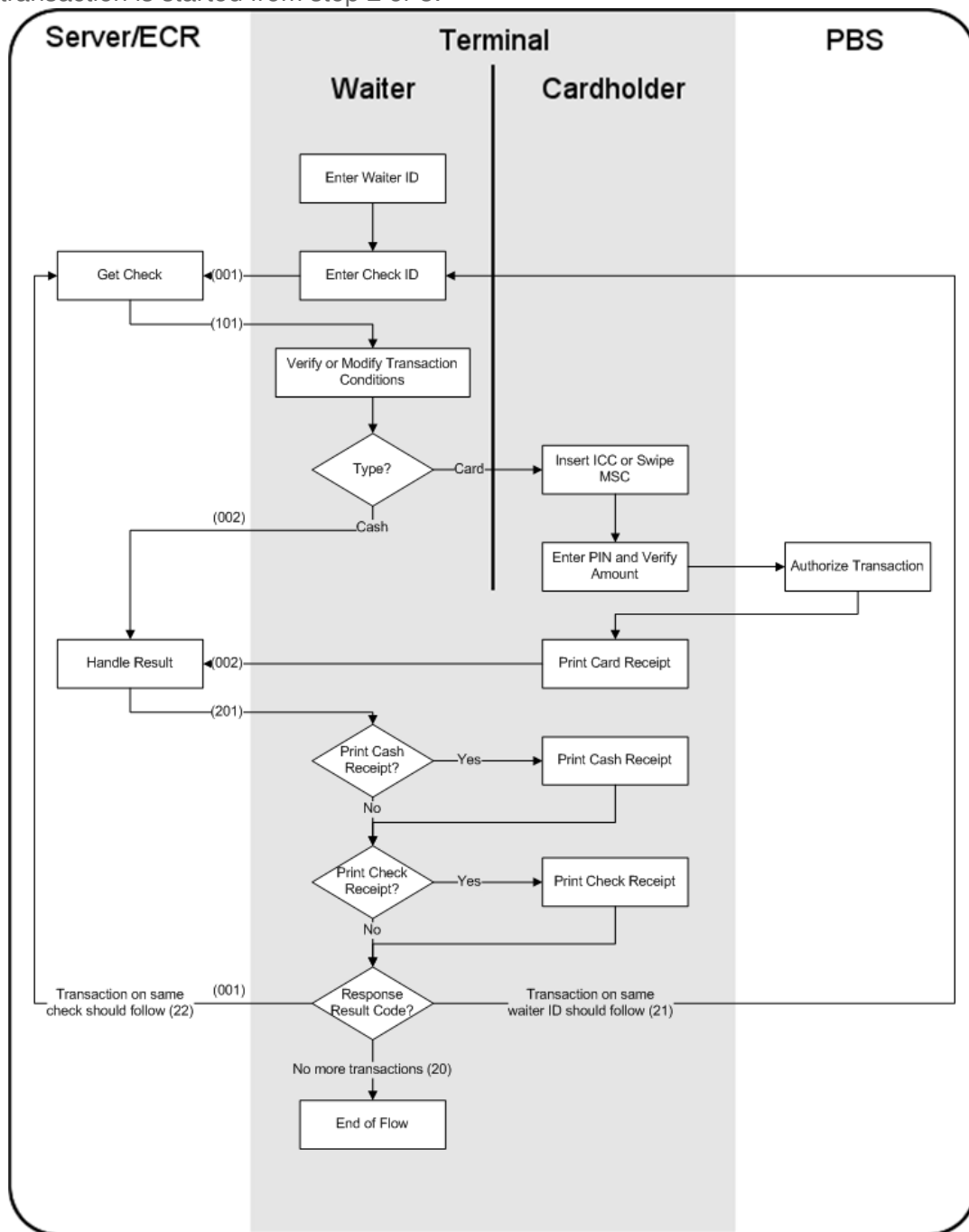




### 3.7.3 Scenario C (Fetch by Check ID)

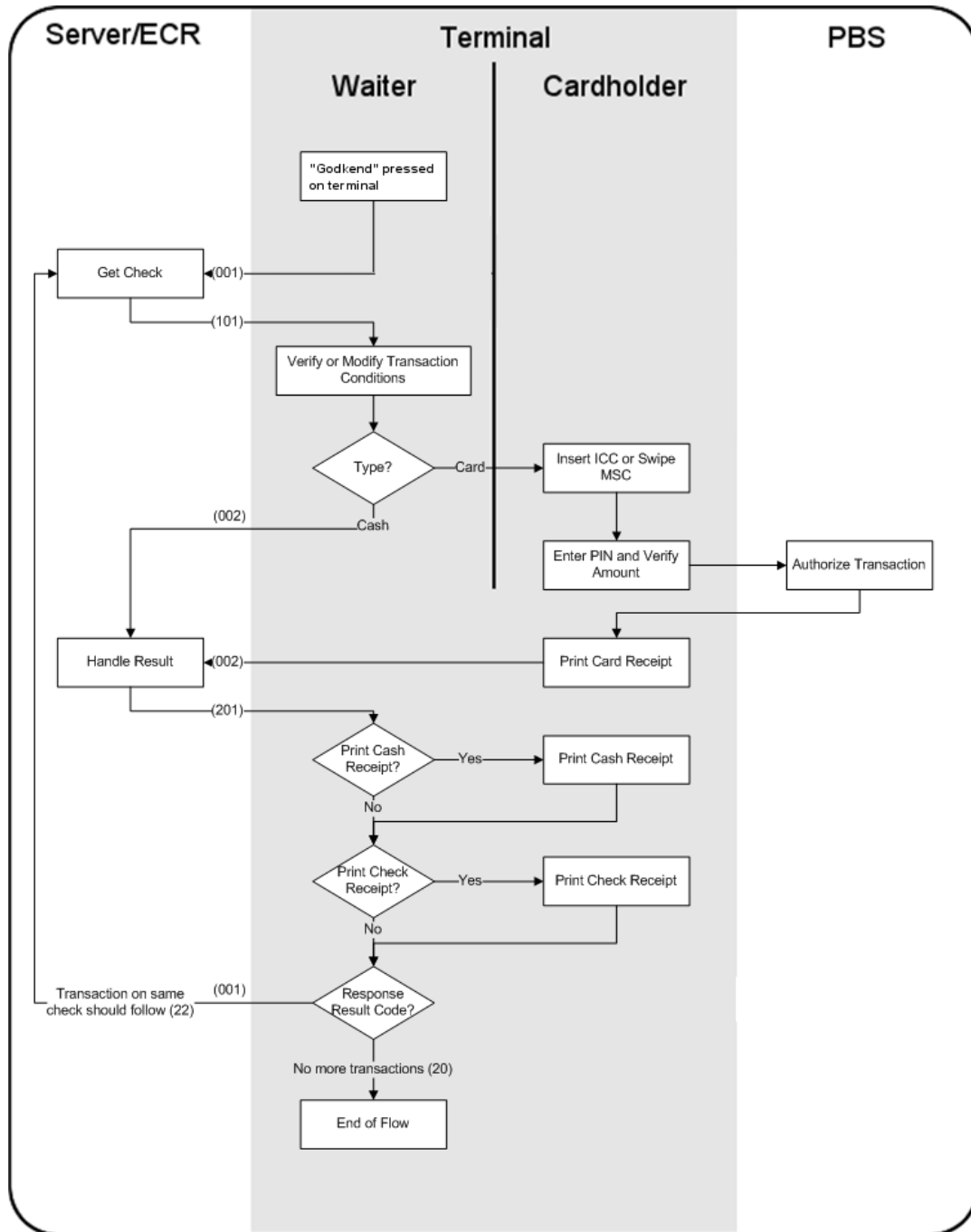
1. A transaction starts with entry of the waiter ID (if enabled).
2. The waiter enter check ID.
3. Terminal fetches the check from the server/ECR.

4. Terminal makes the transaction like a standalone, or nothing in case of cash transaction.
5. Terminal sends the result to the server/ECR
6. The server responds with receipts which should be printed on the terminal and a result code which reflects if a new transaction should be started. The result code controls if a new transaction is started from step 2 or 3.



#### **3.7.4 Scenario D (one-to-one)**

1. A transaction starts with a press on “Godkend”.
2. Terminal fetches the check from the server/ECR.
3. Terminal makes the transaction like a standalone, or nothing in case of cash transaction.
4. Terminal sends the result to the server/ECR.
5. The server responds with receipts which should be printed on the terminal. and a result code which reflects if a new transaction should be started from step 2.



## 3.8 The Spin Connect Functions

### 3.8.1 Get Check (001)

This function is used to get a check from the server/ECR.

Synopsis:

```
<001><DL><TERMINAL_ID><DL><CLERK_ID><DL><CHECK_ID><DL><TABLE_ID>  
<DL>[<CRC ><DL >]
```

[<CRC ><DL >] is only used when using RS232.

When fetching a check by check id, <TABLE\_ID>will be '0'.

When fetching by table number, <CHECK\_ID>will be '0'.

If both <CHECK\_ID>and <TABLE\_ID>is '0', the server should return "Check List (102)" instead of "Get Check Response (101)".

It is also allowed to return "Check List (102)" or "Admin Response (103)" when a request is made for a table. If using Scenario D, lists is not supported and a message 101 or message 103 should be returned no matter the value of <CHECK\_ID>and <TABLE\_ID>.

### 3.8.2 Get Check Response (101)

This function is the response to "Get Check (001)" from the server/ECR to the terminal.

Synopsis:

```
<101><DL><TERMINAL_ID><DL><CLERK_ID><DL><CHECK_ID>  
<DL><RESULT><DL><TYPE><DL><AMOUNT><DL><VAT><DL>  
<CURRENCY><DL>[<CRC ><DL >]
```

[<CRC ><DL >] is only used when using RS232.

<CURRENCY> can be omitted. In this case the terminals home currency is used (i.e. usually DKK in Denmark).

### 3.8.3 Check List (102)

This function is used to send a list of outstanding bills for specific clerk ID or table ID. This is send as a response to a "Get Check (001)" with '0' in <CHECK\_ID>.

Synopsis:

```
<102><DL><TERMINAL_ID><DL><RESULT><DL><CHECK_COUNT><DL>  
<CHECK_ID><DL><MENUITEM><DL>[<CRC ><DL >]
```

[<CRC ><DL >] is only used when using RS232.

Everything enclosed in curly brackets ( { and } ) should be repeated <CHECK\_COUNT>times, representing each outstanding check. The maximum number of checks is 20. If more than 20

check are returned, only the first 20 will be displayed. The field <MENUITEM> represents a text string which represent a check. It's up to the server to format this string. This message can only be returned if Scenario B is supported on the terminal.

### 3.8.4 Admin Response (103)

This message can be responded to any "Get Check (001)" message. It will start an admin function on the terminal. The list of admin commands can be found in Appendix A under <RESULT>.

Synopsis:

<103><DL><TERMINAL\_ID><DL><RESULT><DL>[<CRC ><DL >]

[<CRC ><DL >] is only used when using RS232.

### 3.8.5 Check Result (002)

This function is used to send the transaction result to the server/ECR.

Synopsis:

<002><DL><TERMINAL\_ID><DL><CHECK\_ID><DL><TYPE><DL><RESULT>  
<DL><ASW1ASW2><DL><CRC\_ID><DL><MASKED\_PAN><DL><CURRENCY>  
<DL><AMOUNT><DL><BATCH\_ID><DL><FEE><DL><VAT><DL><EXTRA>  
<DL><CASH\_BACK><DL><CURRENCY\_DCC><DL><AMOUNT\_DCC><DL><FEE\_DCC>  
<DL><EXTRA\_DCC><DL><PSAM\_ID><DL><STAN><DL><CONDITION\_CODE>  
<DL><CLERK\_ID><DL>[<STOREBOXHASH><DL><EKVITTERINGHASH><DL>  
<NETSHASH><DL>][<JSON ><DL >] [<CRC ><DL >]

[<STOREBOXHASH><DL><EKVITTERINGHASH><DL><NETSHASH><DL>] only when the terminal has been setup for e-receipt.

[<VasResponse><DL>] only when the terminal has been setup with VAS.

[<CRC ><DL >] only used when using RS232.

### 3.8.6 Check Result Response (201)

This function is the response to "Check Result (002)".

Synopsis:

<201><DL><TERMINAL\_ID><DL>  
<CHECK\_ID><DL><RESULT><DL>  
<CASH\_RECEIPT><DL><CHECK\_RECEIPT><DL>[<CRC ><DL >]

[<CRC ><DL >] is only used when using RS232.

### 3.8.7 MSC Loyalty Card (003)

This function is used to send the loyalty card to the server/ECR for processing.

Synopsis:

<301><DL><TERMINAL\_ID><DL><CHECK\_ID><DL><RESULT><DL>[<CRC ><DL >]

[<CRC ><DL >] is only used when using RS232.

### **3.8.8 Version Information (004)**

This message is sent once to the server for every boot of the terminal. The message will inform the server of the terminal ID and a version number of spin connect protocol which is used. The message will additionally contain a terminal report.

Synopsis:

<004><DL><TERMINAL\_ID><DL><PROTOCOL\_VERSION><DL>  
<TERMINAL\_REPORT><DL>[<CRC ><DL >]

[<CRC ><DL >] is only used when using RS232.

### **3.8.9 Version Information Response (401)**

This message can be used as a response to message 004. The field <COMPATIBLE\_STATE>will indicate if the server is compatible with the Spin Connect protocol used by the terminal. In case of CRC error in 004 message set <COMPATIBLE\_STATE>to '30'.

Synopsis:

<401><DL><TERMINAL\_ID><DL><COMPATIBLE\_STATE><DL>[<CRC ><DL >]

[<CRC ><DL >] is only used when using RS232.

### 3.A Parameters

ASCII format char 8 bit is used.

Field	Value	Size	Notes
<001>	001	3	Command 'Get Check'
<101>	101	3	Response 'Get Check Response'
<102>	102	3	Response 'Check List'
<002>	002	3	Command 'Check Result'
<201>	201	3	Response 'Check Result Response'
<003>	003	3	Command 'MSC Loyalty Card'
<301>	301	3	Response 'MSC Loyalty Card Response'
<DL>	;	1	Semicolon
<TERMINAL_ID>		≤ 8	PIN pad ID
<CLERK_ID>		≤ 4	Clerk or waiter ID
<TABLE_ID>		≤ 6	Table number.
<CHECK_ID>		≤ 12	Check/Bill identification number.
<TYPE>		1	Type of the transaction. There are the following possible values: P = Card purchase R = Card refund S = Signature card purchase O = Offline card purchase C = Cancellation of card purchase/refund M = Cash purchase B = Cash refund
<CURRENCY>		3	Currency of the transaction. There are the following possible values: 'DKK' = Danish kroner 'EUR' = Euro 'GBP' = Sterling pound 'USD' = US dollar 'SEK' = Swedish kroner 'JPY' = Japanese yen
<AMOUNT>		≤ 8	One of the following: Amount due on check/bill Amount which has been paid in the transaction. This does not include extra and fee.
<BATCH_ID>		12	In case of a card transaction this will hold the batch id assigned to the card transaction. See Nets OTRS specification.



Field	Value	Size	Notes
<ASW1ASW2>		4	Nets Response code. In case of cash transaction this will be '0000'. In case of transaction abort by failure or user request, this will hold 'FFFF'. See Nets OTRS specification.
<CRC_ID>		3	In case of a card transaction this will hold the Card Reconciliation Counter ID. This can be used to identify the card which was used. See Nets OTRS specification.
<MASKED_PAN>		4 – 20	PCI masked PAN.
<FEE>		≤ 8	The amount of surcharge fee that was added to the card transaction.
<VAT>		≤ 8	The VAT amount for the check/bill.
<EXTRA>		≤ 8	The amount of gratuity given.
<CASH_BACK>		≤ 8	Amount of cash back.
<MSC_TRACK2>		≤ 37	Track 2 of MSC loyalty card.
<CURRENCY_DCC>		≤ 8	If a DCC card transaction, this will hold the card holder currency.
<AMOUNT_DCC>		≤ 8	If a DCC card transaction, this will hold the amount in card holder currency.
<FEE_DCC>		≤ 8	If a DCC card transaction, this will hold the fee amount in card holder currency.
<EXTRA_DCC>		≤ 8	If a DCC card transaction, this will hold the gratuity amount in card holder currency.
<PSAM_ID>		≤ 9	The ID of the PSAM used in the terminal.
<STAN>		≤ 9	Identification of the transaction. Together with PSAM ID this gives a unique transaction reference.
<CONDITION_CODE>		≤ 3	Transaction condition code. This code reflects the transaction conditions (which card and which Card Verification Method was used). See OTRS from Nets.
<CHECK_COUNT>		≤ 2	The number of checks in response 102.
<MENUITEM>		≤ 16	A string which represents a check on the "Check List(102)". The delimiter (';') cannot be present in the string.
<CASH_RECEIPT>		≤ 1250	The receipt of a cash transaction. If this is empty no receipt will be printed. This receipt will only be printed in case of a cash transaction, but the field should always be present. The receipt should be formatted to lines of maximum 24 character width. LF should be used as line separator. The character encoding used is ISO 8859-15. The delimiter (';') cannot be present in the receipt.

Field	Value	Size	Notes
<CHECK_RECEIPT>		≤ 1250	The check receipt. If this field is empty, no receipt will be printed. Rules for formatting is as in <CASH_RECEIPT>
<PROTOCOL_VERSION>	1.00	≤ 5	Version of the SpinConnect protocol. The current version described in this document is version 1.03
<COMPATIBLE_STATE>		= 1	This field is used a response the protocol version sent to server. The field can have the following values: '0' = Server and terminal is compatible '1' = Server and terminal is not compatible
<TERMINAL_REPORT>		≤ 3000	Terminal report generated on startup of the terminal.
<CRC>		≤ 10	CRC value for the message.
<RESULT>		≤ 2	This field holds result codes. The following values are possible: '0' = approved (002, 102) '1' = declined (002, 102) '2' = syntax error (101, 102, 002, 201, 301) '10' = Bill found (101) '11' = locked by another terminal (101) '12' = Bill isn't ready (101) '13' = Unknowing Terminal id (101, 102, 201, 301) '14' = Unknown Bill (101) '16' = approved Loyalty card (301) '17' = declined Loyalty card (301) '20' = No more transactions (201) '21' = New transaction on same waiter should follow (201) '22' = New transaction on same check should follow (201) '23' = "End of Day" should follow (201) '30' = CRC error(all messages) '40' = Start "End of Day" (103) '41' = Start "Clean Terminal" (103)
<STOREBOXHASH>			Storebox hash
<EKVITTERINGHASH>			eKvitterings hash
<NETSHASH>			Nets Secure Hash
<VasResponse>			Data from VAS

## 4 | Gavekort Scan Bar Code

### 4.1 Flex driver (DLL)

To support prepaid scanned bar code in the terminal, Flexdriver (flxdrv.dll) now have a new callback to put the scanned bar code to the terminal.

#### 4.1.1 Callback pcbPutScanBarCode

The callback must be enabled by calling:

```
flxInitCallback(FLX_CALLBACK.FLX_CALLBACK_PUT_SCAN_BAR_CODE, pcbPutScanBarCodeDelegate);
```

The callback is called as a result of performing a flxCardTransaction with the

**FLX\_TRANSTYPE** set to one of:

```
FLX_TRANS_PREPAID_SCAN_PURCHASE, // Debit prepaid scan transaction
FLX_TRANS_PREPAID_SCAN_REFUND,   // Credit prepaid scan transaction
FLX_TRANS_PREPAID_SCAN_ORIGINAL_AUTH, // Orig auth prepaid scan transaction,
no token output
```

Remember to scan the bar code before starting the transaction.

Using the FLX\_TRANS\_PREPAID\_SCAN\_PURCHASE transaction type, it's now possible to start a Swipp transaction to a specific phone number given as a cardnumber like:

**sw:45PPPPPPPP** where sw tells it's a Swipp transaction, the 45 is the country code without the normal plus, and the P's are the phone number digits going to be used for the transaction. Other country codes - shorter/longer phone numbers are allowed as long the maximum length of the total string passed is 19 char or below. White space in the end is not allowed.

The terminal must be configured to handle scanned bar code transactions, this is done by a param download, after Verifone has set the terminals configuration in Verifone's terminal database.

```
static int (*pcbPutScanBarCode)(int *ScanBarCodeLength, char *ScanBarCodeData);
```

or the Visual Basic interface

```
static void (WINAPI *pcbPutScanBarCodeVB)(int *ScanBarCodeLength,
    BSTR*ScanBarCodeData, int *res);
```

#### 4.1.2 Saldo and expire data in callback pcbGetReceipt/pcbGetReceiptVB

Two fields added to the extra receipt information, with prepaid info  
Saldo;ExpireDateYYMM

expire date as digits (first: year (bcd) 00-99, second: month (bcd) 01-12)

The terminal must be configured to convey this information ReceiptType with bit 5 set. This is done by a param download, after Verifone has set the terminals configuration in Verifone's terminal database.

### 4.1.3 Example: binary receipt of prepaid scanned bar code purchase

BINARY RECEIPT:

```
2015-12-10 10:07;9208607599999904780;3600;0;0;0;208;471;2129592318;105723;0000;0000;8;D53398;
NETS PP Test 999;00990798;2122227; Hotel DCC Test; Ballerup; Lautrupbjerg 10;
2750; 46352966; ;0;208;1 ;0;980073
;1;0;8B71D5972C1C6998011C07862D740B869505DF13857EADA1F9712EBA0E5EEF52;0;0;0;0;0;0;0;000000;
;0000;38A447E663329E6A1FBA7FAAB6B6ED35A9CA1EBA24D41051C1E9E77C88B64CE6;
8B71D5972C1C6998011C07862D740B869505DF13857EADA1F9712EBA0E5EEF52;;;DC1
```

1. TID	: 2015-12-10 10:07
2. PAN	: 9208607599999904780
3. TOTAL	: 3600
4. EXTRA	: 0
5. GEBYR	: 0
6. DRIKKEPENGE	: 0
7. VALUTAKODE	: 208
8. STAN	: 471
9. PSAMCREATOR	: 2129592318
10. PSAMID	: 105723
11. STATUS	: 0000
12. ASW1ASW2	: 0000
13. CVMSTATUS	: 8
14. AUTORISATIONSKODE	: D53398
15. KORTNAVN	: NETS PP Test 999
16. TERMINALID	: 00990798
17. AFTALENUMMER	: 2122227
18. NAVN	: Hotel DCC Test
19. BY	: Ballerup
20. ADRESSE	: Lautrupbjerg 10
21. POSTNUMMER	: 2750
22. TELEFON	: 46352966
23. CVR	:
24. REF.NUMMER.	: 0
25. DCC VALUTAKODE	: 208
26. DCC KURS	: 1
27. CRC	: 0
28. BATCH	: 980073
29. ANNULLERINGAKTIV	: 1
30. MOMS	: 0
31. E-KVITTERING	: 8B71D5972C1C6998011C07862D740B869505DF13857EADA 1F9712EBA0E5EEF52
32. SALDO	: 0

33.	BACK	:	0
34.	DCCTOTAL	:	0
35.	DCCGEBYR	:	0
36.	DCCDRIKKEPENGE	:	0
37.	CARDDATASOURCE	:	0
38.	AID	:	
39.	ATC	:	0
40.	AED	:	000000
41.	ARC	:	
42.	EXPDATE	:	0000
43.	STOREBOXDK	:	38A447E663329E6A1FBA7FAAB6B6ED35A9CA1EBA24D4105 1C1E9E77C88B64CE6
44.	EKVITTERING	:	8B71D5972C1C6998011C07862D740B869505DF13857EADA 1F9712EBA0E5EEF52
45.	NETSDANMARK	:	
46.	STOREBOX	:	
47.	vasJsonReceipt	:	
48.	TCC	:	DC1

#### 4.1.4 Example: binary receipt of completed MobilePay transaction

BINARY RECEIPT:

```
2017-03-02 11:24;920861615641535085;35;0;0;0;208;12345;0;0;0000;0000;0;
MobilePay; ;0; Verifone QA; Herlev; Knapholm 7; 2730; ;
POSDKVF418;0;0;1 ;508;;0;0; 991187170302112423 ;0;0;35;0;0;0;;0;000000;;0000;;;
5e9e967ee5d7464993b2b9e831b081a5;
{"time":"11:24","date":"2017-03-02","currency":"DKK","amount":35,
"orderId":"991187170302112423","customerToken":"","
"customerReceiptToken":"5e9e967ee5d7464993b2b9e831b081a5",
"transactionId":"61258283","tr":0,"status":"DONE"
};VF1
```

1.	TID	:	2017-03-02 11:24
2.	PAN	:	920861615641535085
3.	TOTAL	:	35
4.	EXTRA	:	0
5.	GEBYR	:	0
6.	DRIKKEPENGE	:	0
7.	VALUTAKODE	:	208
8.	STAN	:	12345 (locationId - "xxxxx") prefixed 0's are removed
9.	PSAMCREATOR	:	0
10.	PSAMID	:	0
11.	STATUS	:	0000
12.	ASW1ASW2	:	0000
13.	CVMSTATUS	:	0
14.	AUTORISATIONSKODE	:	

15. KORTNAVN	: MobilePay	
16. TERMINALID	:	
17. AFTALENUMMER	: 0	
18. NAVN	: Verifone QA	(merchantName)
19. BY	: Herlev	(locationCity[5])
20. ADRESSE	: Knapholm 7	(locationStreet)
21. POSTNUMMER	: 2730	(locationCity)
22. TELEFON	:	
23. CVR	: POSDKVF418	(merchantId)
24. REF.NUMMER.	: 0	
25. DCC VALUTAKODE	: 0	
26. DCC KURS	: 1	
27. CRC	: 508	
28. BATCH	:	
29. ANNULLERINGAKTIV	: 0	
30. MOMS	: 0	
31. E-KVITTERING	: 991187170302112423	(orderId)
32. SALDO	: 0	
33. BACK	: 0	
34. DCCTOTAL	: 35	
35. DCCGEBYR	: 0	
36. DCCDRIKKEPENGE	: 0	
37. CARDDATASOURCE	: 0	
38. AID	:	
39. ATC	: 0	
40. AED	: 000000	
41. ARC	:	
42. EXPDATE	: 0000	
43. STOREBOXDK	:	
44. EKVITTERING	:	
45. NETSDANMARK	:	
46. STOREBOX	: 5e9e967ee5d7464993b2b9e831b081a5	
47. vasJsonReceipt	: {"time":"11:24", "date":"2017-03-02" "currency":"DKK", "amount":35, "orderId":"991187170302112423", "customerToken":""," "customerReceiptToken":"5e9e967ee5d7464993b2b9e831b081a5", "transactionId":"61258283", "tr":0,, "status":"DONE" }	
48. TCC	: VF1	

For readability newlines are inserted in the vasJsonReceipt - real data are without newlines.

#### 4.1.5 Example: binary receipt of failed Swipp prepaid scanned bar code purchase

As Swipp is no longer available as a feature, this example should serve as a guideline for implementing MobilePay.

##### BINARY RECEIPT:

```
2015-12-10 09:51;4561626605;3600;0;0;0;208;0;0;0;0000;120B;0;      ;      VAS TYPE 507;
;0;;      ;      ;
;      ;0;0;1      ;507;;0;0;;0;0;0;0;0;0;0;000000;
;0000;;;;{"date":"2015-12-10","time":"09:51","currency":"DKK","amount":"3600",
"merch_id":"Verifone","pos_id":"991027","key_id":"3","trans_id":"","
"ref_id":"8c4a53049f2311e58a3100081924a19b","status":"FAILED","fail_code":"404001"};VF1
```

1. TID	: 2015-12-10 09:51
2. PAN	: 4561626605
3. TOTAL	: 3600
4. EXTRA	: 0
5. GEBYR	: 0
6. DRIKKEPENGE	: 0
7. VALUTAKODE	: 208
8. STAN	: 0
9. PSAMCREATOR	: 0
10. PSAMID	: 0
11. STATUS	: 0000
12. ASW1ASW2	: 120B
13. CVMSTATUS	: 0
14. AUTORISATIONSKODE	:
15. KORTNAVN	: VAS TYPE 507
16. TERMINALID	:
17. AFTALENUMMER	: 0
18. NAVN	:
19. BY	:
20. ADRESSE	:
21. POSTNUMMER	:
22. TELEFON	:
23. CVR	:
24. REF.NUMMER.	: 0
25. DCC VALUTAKODE	: 0
26. DCC KURS	: 1
27. CRC	: 507
28. BATCH	:
29. ANNULLERINGAKTIV	: 0
30. MOMS	: 0
31. E-KVITTERING	:
32. SALDO	: 0
33. BACK	: 0
34. DCCTOTAL	: 0
35. DCCGEBYR	: 0
36. DCCDRIKKEPENGE	: 0

37. CARDDATASOURCE	: 0
38. AID	:
39. ATC	: 0
40. AED	: 000000
41. ARC	:
42. EXPDATE	: 0000
43. STOREBOXDK	:
44. EKVITTERING	:
45. NETSDANMARK	:
46. STOREBOX	:
47. vasJsonReceipt	: {"date":"2015-12-10","time":"09:51", "currency":"DKK", "amount":"3600", "merch_id":"Verifone", "pos_id":"991027", "key_id":"3", "trans_id":"", "ref_id":"8c4a53049f2311e58a3100081924a19b", "status":"FAILED", "fail_code":"404001" }
48. TCC	: VF1

## 4.2 Local Payment Protocol (LPP)

To support prepaid scanned bar code in the terminal LPP now have to put the scanned bar code to the terminal.

### 4.2.1 LPP start scanned bar code transaction

In the PTAG\_TRANSACTION the PTAG\_CARD\_SOURCE must be set to 0x04 CARD\_SCAN, this will tell the terminal to wait for the scanned bar code from the ECR.

PTAG\_PREPAID is set based on the transaction started:

Purchase	TR=0x00	PTAG_PREPAID=0x03
Refund	TR=0x01	PTAG_PREPAID=0x02
Original Auth.	TR=0x02	PTAG_PREPAID=0x01

The terminal must be configured to handle scanned bar code transactions, this is done by a param download, after Verifone has set the terminals configuration in Verifone's terminal database.

### 4.2.2 LPP prepaid scanned bar code to ECR

Data is put into a PTAG\_DATA container, this must have a PTAG\_CARDNUMBER containing the scanned bar code.



### 4.2.3 LPP PTAG\_RECEIPT now contains PTAG\_SALDO and PTAG\_EXP\_DATE

You can obtain the cards saldo and expire date from the receipt container by looking at PTAG\_SALDO 0xC3 it contains a 4 byte unsigned amount and PTAG\_EXP\_DATE 0xC5 with expire date as 2 byte (first: year (bcd) 00-99, second: month (bcd) 01-12).

### 4.2.4 Example: LPP Start of scanned bar code purchase from terminal to ECR

```
TimeStamp: 08:54:37
STX       : 02
SEQ       : 02
TAG       : 65 - PTAG_TRANSACTION Container
TAGlen    : 2e
  TAG     : 4e - PTAG_MI
  TAGlen  : 01
  TAGvalue: 00      ',.'
  TAG     : 4c - PTAG_CU
  TAGlen  : DLE 02
  TAGvalue: 00 d0   ',.D'
  TAG     : 56 - PTAG_TT
  TAGlen  : 01
  TAGvalue: 00      ',.'
  TAG     : 50 - PTAG_TR
  TAGlen  : 01
  TAGvalue: 00      ',.'
  TAG     : 4a - PTAG_REF_NO
  TAGlen  : 04
  TAGvalue: 00 01 e2 40  '..â@'
  TAG     : 48 - PTAG_AMOUNT
  TAGlen  : 04
  TAGvalue: 00 00 00 16  '....'
  TAG     : 59 - PTAG_GRATUITY
  TAGlen  : 04
  TAGvalue: 00 00 00 00  '....'
  TAG     : 57 - PTAG_VAT
  TAGlen  : 04
  TAGvalue: 00 00 00 00  '....'
  TAG     : 94 - PTAG_TERM_ENV
  TAGlen  : 01
  TAGvalue: 00      ',.'
```

```

TAG      : 92 - PTAG_CARD_SOURCE
TAGlen   : 01
TAGvalue : 04                                     ', '

TAG      : 93 - PTAG_PREPAID
TAGlen   : 01
TAGvalue : DLE 03                                ', , '

ETX      : 03
CRC      : 38c7

TimeStamp: 08:54:37
ACK      : 06
SEQ      : 02

TimeStamp: 08:54:37
STX      : 02
SEQ      : 03
TAG      : 63 - PTAG_DATA Container
TAGlen   : 15
TAG      : 52 - PTAG_CARDNUMBER
TAGlen   : 13
TAGvalue : 36 30 37 35 31 30 30 30 30 30 30 30
          30 31 31 34 '6075100000000114'
TAGvalue : 37 31 31                                     '711'

ETX      : 03
CRC      : 8d6d

```

#### 4.2.5 Example: LPP Start of Swipp scanned bar code purchase from terminal to ECR

```

TimeStamp: 09:50:56
STX      : 02
SEQ      : 07
TAG      : 65 - PTAG_TRANSACTION Container
TAGlen   : 2e
TAG      : 4e - PTAG_MI
TAGlen   : 01
TAGvalue : 00                                     ', '

TAG      : 4c - PTAG_CU
TAGlen   : DLE 02
TAGvalue : 00 d0                                ', D '

TAG      : 56 - PTAG_TT
TAGlen   : 01
TAGvalue : 00                                     ', '

```

TAG	:	50 - PTAG_TR	
TAGlen	:	01	
TAGvalue	:	00	'.'
TAG	:	4a - PTAG_REF_NO	
TAGlen	:	04	
TAGvalue	:	00 00 00 00	'....'
TAG	:	59 - PTAG_GRATUITY	
TAGlen	:	04	
TAGvalue	:	00 00 00 00	'....'
TAG	:	57 - PTAG_VAT	
TAGlen	:	04	
TAGvalue	:	00 00 00 00	'....'
TAG	:	c4 - PTAG_BACK	
TAGlen	:	04	
TAGvalue	:	00 00 00 00	'....'
TAG	:	94 - PTAG_TERM_ENV	
TAGlen	:	01	
TAGvalue	:	00	'.'
TAG	:	92 - PTAG_CARD_SOURCE	
TAGlen	:	01	
TAGvalue	:	04	'.'
TAG	:	93 - PTAG_PREPAID	
TAGlen	:	01	
TAGvalue	:	DLE 03	'..'
ETX	:	03	
CRC	:	45ad	
TimeStamp:		09:50:56	
ACK	:	06	
SEQ	:	07	
TimeStamp:		09:50:56	
STX	:	02	
SEQ	:	08	
TAG	:	63 - PTAG_DATA Container	
TAGlen	:	0f	
TAG	:	52 - PTAG_CARDNUMBER	
TAGlen	:	0d	
TAGvalue	:	73 77 3a 34 35 36 31 36 32 36 36 30 35	'sw:4561626605'

ETX : 03  
CRC : 5269

## 4.3 Configuration

```
[SCAN_ENTRY]
"01"
[RECEIPT_TYPE]
"45"          // bit 5 - 32 must be set for
               // SALDO / EXP_DATE in tagged receipt
               // 13 + 32 = 45 or 141 + 32 = 173
```

# 5 | Extra App Protocols

## 5.1 Preface

### 5.1.1 Format of the header

The header is 4 bytes long and contains the following fields:

char app_no;	Application number in terminal
char version;	Version of the commands in the data field
char error;	0x00 No error
char last_block;	0xFF Last block of data, 0x00 more blocks

### 5.1.2 Application number

The terminal can contain up to 8 extra applications the first have an application id of 12 in the terminal. So app\_no has to be between 12 and 19. Verifone Denmark may use app\_no above 21 for special tasks.

### 5.1.3 Version

The version is used to tell how the data field after the header has to be decoded.

0x00	Default the data field is plain text/XML with newlines
0xFD	Status for all extra application from ECR, only to be passed on if needed.
0xFE	If the extra application sends this, the terminal will relay the packet back to the extra application. Used for testing new function in extra application where the ECR not ready to respond.
0xFF	If the terminal gets extra data from the ECR with this version, terminal relay the packet back to the ECR. Used for testing new function in ECR where the extra app in the terminal not yet ready to respond.

### 5.1.4 Error

The error is used to indicate problems found in the data communicated. The version is always set to 0x00 on errors and the data field following the header contains a text with detail about the error.

0xE0	AppNo %d not allowed - must be above %d
0xE1	Version %d not supported by this %d terminal
0xE2	Length max is 1000 bytes data + 4 byte header - you send %d bytes total
0xE2	Length must be minium 4 the size of header - you send only %d
0xE2	Length overflow in extra app max is %d
0xE3	Host not responding - make sure host listen for extra app data
0xE3	Terminal unable to send to host - current state is %d - %s
0xE3	Host not ready - wait card swipe must be enabled - currentEcrExtendedFunctions 0x%04x need 0x0100 set
0xE3	Terminal unable to send to host – other menu %d running state is %d - %s
0xE3	Host not ready - DLL/OCX not in IP Idle (wait card swipe) or Transaction mode
0xE4	flxIdleIPforwarding no connection to terminal
0xE4	flxCARDTransaction no connection to terminal

### 5.1.5 Last block

If the data block you want to send is longer than 1000 bytes, its send in blocks of 1000 bytes + one block with remainder.

0x00	More blocks of data to come
0xFF	Last block of data

## 5.2 The Data Block

### 5.2.1 The Data Block

Up to 1000 bytes of data, current version 0x00 uses plaintext/XML. Data block comes right after the header.

If the first byte is ascii 0x3c - '<' XML data may be expected, else plain text is used.

### 5.2.2 Displayline

Text:	Max length is 20 char total.
format:	Values
left	center
	right
No:	Lineno values
-2	Show Idle Screen
-1	Append to next line
1-6	Display text on specific display line. Not tested.

Xpos is number of spaces put in front of Text. The terminal uses a proportional font - use with care.

An example on some displayline data:

```
<ExtraAppData>
  <DataItem type="0" format="displayline" typehelp="Display line
with item to buy" clearScreen="false"/>
  <Line>
    <Text format="left">Cola 2L</Text>
    <Text format="right">25,00</Text>
    <No>-01</No>
    <Xpos>00</Xpos>
  </Line>
</ExtraAppData>
```

### 5.2.3 Example plaintext data block

Here app\_no 12, version 0x00, error 0x00, last\_block 0xff

Extra app has received 44 data bytes and performed some action on it

```
HexDump Size=72
Hex   0  1  2  3  4  5  6  7  -  8  9  A  B  C  D  E  F
0000  0c 00 00 ff 45 78 74 72 - 61 20 61 70 70 20 68 61 ....Extra app ha
0010  73 20 72 65 63 65 69 76 - 65 64 20 34 34 20 64 61 s received 44 da
0020  74 61 20 62 79 74 65 73 - 20 61 6e 64 20 70 65 72 ta bytes and per
0030  66 6f 72 6d 65 64 20 73 - 6f 6d 65 20 61 63 74 69 formed some acti
0040  6f 6e 20 6f 6e 20 69 74                               on on it
```

### 5.2.4 Example XML data block

Here app\_no 12, version 0x00, error 0x00, last\_block 0xff

```
<ExtraAppData>
  <DataItem type="0" format="string" typehelp="ISO-8859-15 strings"/>
  <Data>First line of test data</Data>
  <Data>Second line of test data</Data>
</ExtraAppData>
```

```
Hexdump
Hex   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0x00  0c 00 00 ff 3c 45 78 74 72 61 41 70 70 44 61 74 ....<ExtraAppDat
0x10  61 3e 0a 20 20 3c 44 61 74 61 49 74 65 6d 20 74 a>. <DataItem t
0x20  79 70 65 3d 22 30 22 20 66 6f 72 6d 61 74 3d 22 ype="0" format="
0x30  73 74 72 69 6e 67 22 20 74 79 70 65 68 65 6c 70 string" typehelp
0x40  3d 22 49 53 4f 2d 38 38 35 39 2d 31 35 20 66 6f ="ISO-8859-15 fo
```

```

0x50  72 6d 61 74 65 64 20 73 74 72 69 6e 67 22 2f 3e  rminated string"/>
0x60  0a 20 20 3c 44 61 74 61 3e 46 69 72 73 74 20 6c  .  <Data>First l
0x70  69 6e 65 20 6f 66 20 74 65 73 74 20 64 61 74 61  ine of test data
0x80  3c 2f 44 61 74 61 3e 0a 20 20 3c 44 61 74 61 3e  </Data>.  <Data>
0x90  53 65 63 6f 6e 64 20 6c 69 6e 65 20 6f 66 20 74  Second line of t
0xa0  65 73 74 20 64 61 74 61 3c 2f 44 61 74 61 3e 0a  est data</Data>.
0xb0  3c 2f 45 78 74 72 61 41 70 70 44 61 74 61 3e 00  </ExtraAppData>.

```

## 5.3 XML format types

### 5.3.1 String

An example on some string data:

```

<ExtraAppData>
  <DataItem type="0" format="string" typehelp="some help text go here"/>
  <Data>First line of test data</Data>
  <Data>Second line of test data</Data>
</ExtraAppData>

```

### 5.3.2 Hexstring

An example on some hexstring data with 32 bytes:

```

<ExtraAppData>
  <DataItem type="0" format="hexstring" typehelp="example with 32 bytes"/>
  <Data>000102030405060708090A0B0C0D0E0F</Data>
  <Data>101112131415161718191A1B1C1D1E1F</Data>
</ExtraAppData>

```

An example on some hexstring data with 35 bytes:

```

<ExtraAppData>
  <DataItem type="0" format="hexstring" typehelp="example with 35 bytes"/>
  <Data>000102030405060708090A0B0C0D0E0F</Data>
  <Data>101112131415161718191A1B1C1D1E1F</Data>
  <Data>202122</Data>
</ExtraAppData>

```

### 5.3.3 Displayline

Current terminal only support line 1, max. length is 20 characters.  
Format can be one of



left  
center  
right

Xpos is number of spaces put in front of Text.  
An example on some displayline data:

```
<ExtraAppData>
  <DataItem type="0" format="displayline" typehelp="some help text go here"/>
  <Line>
    <Text>1 Cola          27,50</Text>
    <No>01</No>
    <Xpos>00</Xpos>
    <Format>center</Format>
  </Line>
</ExtraAppData>
```

### 5.3.4 Select pictures

Terminal has a selection of pictures, obtained by a download, the numbers given in this command tell the order and of the pictures to be shown and how long in ms. Because of a limitation in the terminal, the maximum size of the pictures is  $320 \times 181$  pixels when using this method to display idle images.

Timeout must be above 200 ms.

An example on some selectpictures data:

```
<ExtraAppData>
  <DataItem type="0" format="selectpictures" typehelp="3 pictures"/>
  <IdleScreen>
    <Frame>
      <Filename>matas_01</Filename>
      <Timeout>1000</Timeout>
      <Index>001</Index>
      <Description>Matas default image</Description>
    </Frame>
    <Frame>
      <Filename>matas_02</Filename>
      <Timeout>2500</Timeout>
      <Index>007</Index>
      <Description>Matas special offer 1</Description>
    </Frame>
    <Frame>
      <Filename>matas_03</Filename>
      <Timeout>2500</Timeout>
```

```

        <Index>002</Index>
        <Description>Matas special offer 2</Description>
    </Frame>
</IdleScreen>
</ExtraAppData>

```

### 5.3.5 Select picture reply

On each picture 3 buttons can be active, if one is pressed this is sent to the host.

```

<Filename>    The picture shown at button press
<Button>      The button pressed Corr, F1, F2, F3, Menu, Num, OK, Stop.
<ExtraAppData>
    <DataItem type="0" format="selectpicturereply" typehelp="some text"/>
    <Filename>netto_01</Filename>
    <Button>F2</Button>
</ExtraAppData>

```

### 5.3.6 Print receipt

The extra application may have some receipt data to be printed. The data must be formatted as RTF.

An example on some printreceipt data to print:

Some RTF test

```

<ExtraAppData>
    <DataItem type="0" format="printreceipttrtf" typehelp="RTF receipt"/>
    <RTF>{\rtf1\ansi\ansicpg1252\deff0\deflang1030{\fonttbl{\f0\fnil\fcharset0 Courier New;}
        {\f1\fnil\fcharset0 Calibri;}{\f2\fnil\fcharset0 Arial;}}
        {\*generator Msftedit 5.41.21.2510;}\viewkind4\uc1\pard\sa200\sl276\slmult1\lang6\f0\fs22 Some\f1  \b\fs28 RTF\b0\fs22  \f2\fs18 test\par
    }</RTF>
</ExtraAppData>

```

## 5.4 Generic LPP

Used to send generic LPP data to the ECR using known LPP tags.

### 5.4.1 Generic LPP start extra application - Not implemented - Do not use

```

<ExtraAppData>
    <DataItem type="0" format="LPPstarttransaction" typehelp="LPP tag and data"/>
    <MI>0</MI>
    <CU>0</CU>
    <TT>0</TT>
    <TR>0</TR>
    <REF_NO>12345678</REF_NO>
</ExtraAppData>

```

## 5.4.2 Generic LPP extra application transaction completed - Not implemented - Do not use

```
<ExtraAppData>
  <DataItem type="0" format="LPPtransactioncompleted" typehelp="LPP tag and data"/>
  <TEXT>Extra app transaction performed<br>Line 2 of transaction data<br>Line 3<br><br></TEXT>
  <RESULT>0</RESULT>
  <BINARY>0</BINARY>
  <REF_NO>12345678</REF_NO>
</ExtraAppData>
```

## 5.4.3 Generic LPP card data - Not implemented - Do not use

Tell the ECR the card number/crc going to be used for this transaction.

```
<ExtraAppData>
  <DataItem type="0" format="LPPcarddata" typehelp="LPP tag and data"/>
  <CARDNUMBER>92080057000012345678901234</CARDNUMBER >
  <CRCNUMBER>0</CRCNUMBER>
</ExtraAppData>
```

## 5.4.4 Generic LPP card data reply - Not implemented - Do not use

ECR reply with status indicating if it wants to continue with this transaction and the amount going to be used.

```
<RESULT>
0          Cardnumber not OK
1          Cardnumber OK
2          Cardnumber OK amount other
3          Cardnumber OK ask user to confirm amount
<AMOUNT>   Amount in smallest unit of currency
<CU>
208        DKK
352        ISK
392        JPY
578        NOK
752        SEK
756        CHF
826        GBP
840        USD
978        EUR
```

```
<ExtraAppData>
  <DataItem type="0" format="LPPcarddatareply" typehelp="LPP tag and data"/>
  <RESULT>0</ RESULT >
  <AMOUNT>0</AMOUNT>
  <CU>0</CU>
```

```
</ExtraAppData>
```

#### 5.4.5 Generic LPP info text

Display a line of text on the line no specified in <BINARY> on the terminal, uses ISO-8859-15 encoding.

```
<ExtraAppData>
  <DataItem type="0" format="LPPinfotext" typehelp="LPP tag and data"/>
  <TEXT>Disp. up to 20 char</TEXT>
  <BINARY>1</BINARY>
</ExtraAppData>
```

#### 5.4.6 Generic LPP info TSI (Transaction State Info)

Display a line of text (TSI) on the line no specified in <BINARY> on the terminal, using ISO-8859-15 encoding.

```
<ExtraAppData>
  <DataItem type="0" format="LPPinfotsi" typehelp="LPP tag and data"/>
  <TSI>Disp. up to 20 char</TSI>
  <BINARY>1</BINARY>
</ExtraAppData>
```

#### 5.4.7 Generic LPP menu

Display a menu of up to 10 lines each separated by <br> of max. 20 characters, using ISO-8859-15 encoding.

<TEXT> Up to 10 menu lines

<BINARY> Timeout in milliseconds

```
<ExtraAppData>
  <DataItem type="0" format="LPPmenu" typehelp="LPP tag and data - menu with 2 lines"/>
  <TEXT>Menuline 1 up to 20 char\nMenuline 2\nTEXT>
  <BINARY>6000</BINARY>
</ExtraAppData>
```

#### 5.4.8 Generic LPP menu reply

Get the menu reply

<RESULT>	
-2	Timeout
-1	No
0	No
1	First menu line selected/Yes
2	Second menu line selected
...	
10	10th menu line selected

```
<ExtraAppData>
  <DataItem type="0" format="LPPmenureply" typehelp="LPP tag and data line 1 selected"/>
  <RESULT>1</RESULT>
</ExtraAppData>
```

#### 5.4.9 Generic LPP admin

Start extra administration command, if some data needed it can be set in the <PARAM\_STR> else this can be omitted.

```
<ExtraAppData>
  <DataItem type="0" format="LPPadmin" typehelp="LPP tag and data"/>
  <COMMAND>1</COMMAND>
  <PARAM_STR>Some data to be used by the admin function;another
  parameter;234</PARAM_STR>
</ExtraAppData>
```

#### 5.4.10 Generic LPP admin reply

```
<ExtraAppData>
  <DataItem type="0" format="LPPadminreply" typehelp="LPP tag and data"/>
  <TEXT>Extra admin function stopped after performing some action
  in the extra application</TEXT>
  <PARAM_STR>Optional response data from admin function;another parameter;234</PARAM_STR>
  <RESULT>0</RESULT>
</ExtraAppData>
```

#### 5.4.11 Generic LPP receipt

Print receipt data.

Observe: The text is printed in a fixed pitch with no effects (bold/italic/underline).

<TEXT> the receipt data to be printed, using ISO-8859-15

<BINARY>

- 0 Complete - this is last/only part of receipt
- 1 Missing segments - more receipt parts to come
- 2 Need signature

```
<ExtraAppData>
  <DataItem type="0" format="LPPreceipt" typehelp="LPP tag and data"/>
  <TEXT>Receipt data to be printed\nMax 24 or 48 chars pr. line\n</TEXT>
  <BINARY>0</BINARY>
  <REF_NO>12345678</REF_NO>
</ExtraAppData>
```

### 5.4.12 Extra app special

Used to send special data - to/from the extra application.

A mix of data formats can be used, as long the extra application in the terminal and the ECR know how to parse the data.

Example of uses:

Configuration data to setup the extra application

Show special block on the ECR

...

An example on some ExAppCmd237 data:

```
<ExtraAppData>
  <DataItem type="0" format="ExAppCmd237" typehelp="some help text go here"/>
  <Data237>ExAppCmd237 specific data in format known to the
  ECR and Extra app.</Data237>
</ExtraAppData>
```

## 5.5 Flexdriver (DLL)

To support the extra applications in the terminal, Flexdriver (flxdrv.dll) now have a new callback and a function to reply to the extra application in the terminal.

### 5.5.1 flxExtraReply

This function is used to send data to the extra application in the terminal, our DLL kasse demo (no longer supported) show examples of use of the command.

The ECR must ensure connection to the terminal is running, before sending data to the extra application in the terminal.

This is done by running flxIdleIPforwarding or flxCardTransaction.

int flxExtraReply(struct ExtraAppReplyStruct\ \_T \*ExtraAppData, int extra\\_length)  
or the Visual Basic interface

int flxExtraReply(struct ExtraAppReplyStruct\ \_T \*ExtraAppData, int extra\\_length)  
Returns

0: All OK

-1 : Communication to terminal timed out (after 20 sec) and a callback is made to tell the extra app, this status.

### 5.5.2 Callback pcbExtraAppdataReceived

The callback is called as a result of the extra application performing a send data to the ECR. For this to work the ECR must be listening for the callback, if the function flxIdleIPforwarding(2, ref error); is called. We wait for a card swipe and if some extra application data is received it passed on in the pcbExtraAppdataReceived callback.

If a card has been swiped, the flxIdleIPforwarding must be stopped, and the flxCARDTransaction called to start a transaction.

This is done by setting the status to abort in the FLX\_CALLBACK\_ABORT.

In the switch between flxIdleIPforwarding and flxCARDTransaction the extra app will receive a response saying ECR not listening for data from the terminal, this should prevent packets being lost.

```
static int (*pcbExtraAppdataReceived)(size_t ExtraApplength,
const unsigned char *EXTRA\_APP\_DATA\_BLOCK\_block, char AppNo) = NULL;
or the Visual Basic interface
static void (WINAPI *pcbExtraAppdataReceivedVB)(int Extralength,
unsigned char *EXTRA\_APP\_DATA\_BLOCK\_block, char AppNo, int *res) = NULL;
```

### 5.5.3 Enabling the pcbExtraAppdataReceived callback

To register the callback in the flxdrv.dll you need to do an

```
flxInitCallback(FLX\_CALLBACK.FLX\_CALLBACK\_EXTRA\_APP\_DATA\_RECEIVED,
pcbExtraAppdataReceivedDelegate);
and set the EcrExtendedFunctions bit 0x0100.
```

This is be done by:

1. Obtaining EcrExtendedFunctions:

```
flxGetSetEcrExtendedFunctions(ADMIN\_GET\_ECR\_EXTENDED\_FUNCTIONS, & EcrExtendedFunctions);
```

2. Set the bit.

```
EcrExtendedFunctions = EcrExtendedFunctions | 0x0100;
```

3. Set the EcrExtendedFunctions in terminal to the new value:

```
flxGetSetEcrExtendedFunctions(ADMIN\_SET\_ECR\_EXTENDED\_FUNCTIONS, & EcrExtendedFunctions);
```

## 5.6 Local Payment Protocol (LPP)

Extra application uses new tag PTAG\_EXTRA\_APP – 0x53 to communicate the extra application data to/from the terminal.

### 5.6.1 LPP Extra application data from terminal to ECR

Data is put into a PTAG\_DATA container, this have two fields PTAG\_BINARY is used to hold a copy of the app\_no PTAG\_EXTRA\_APP hold the 4 byte header and up to 1000 bytes data.

### 5.6.2 Example LPP Extra application data from terminal to ECR

Here app\_no 12, version 0x00, error 0x00, last\_block 0xff

```
TimeStamp: 12:33:28
STX       : 02
SEQ       : 05
TAG       : 63 - PTAG_DATA Container
TAGlen    : 59
TAG       : 80 - PTAG_BINARY
TAGlen    : 01
TAGvalue  : 0c          ',.'

TAG       : 53 - PTAG_EXTRA_APP
TAGlen    : 48
TAGvalue  : 0c 00 00 ff 45 78 74 72 61 20 61 70 70 20 68 61 '...ÿExtra app ha'
TAGvalue  : 73 20 72 65 63 65 69 76 65 64 20 34 34 20 64 61 's received 44 da'
TAGvalue  : 74 61 20 62 79 74 65 73 20 61 6e 64 20 70 65 72 'ta bytes and per'
TAGvalue  : 66 6f 72 6d 65 64 20 73 6f 6d 65 20 61 63 74 69 'formed some acti'
TAGvalue  : 6f 6e 20 6f 6e 20 69 74          'on on it'

TAG       : 89 - PTAG_SEQNO
TAGlen    : 01
TAGvalue  : 01          ',.'

TAG       : 85 - PTAG_FROMSTATE
TAGlen    : 01
TAGvalue  : 07 - PSTATE_OPEN          ',.'

TAG       : 87 - PTAG_EVENT
TAGlen    : 01
TAGvalue  : 6c - PEVENT_SIG_EXTRA2HOST '1'

TAG       : 86 - PTAG_STATE
TAGlen    : 01
TAGvalue  : 07 - PSTATE_OPEN          ',.'

ETX       : 03
CRC       : ae35
```



### 5.6.3 LPP Extra application data from ECR to terminal

Data is put into a PTAG\_DATA\_1 container, this have one field PTAG\_EXTRA\_APP this hold the 4 byte header and up to 1000 bytes data.

### 5.6.4 Enable ExtraReply in the terminal

Use the ADMIN\_GET\_ECR\_EXTENDED\_FUNCTIONS to get current EcrExtendedFunctions value or the value with 0x0100 and use the ADMIN\_SET\_ECR\_EXTENDED\_FUNCTIONS

```
TimeStamp: 09:23:57
STX       : 02
SEQ       : 03
  TAG      : 67 - PTAG_ADMIN Container
  TAGlen   : DLE 03
    TAG    : 88 - PTAG_COMMAND
    TAGlen  : 01
    TAGvalue : 85 - ADMIN_GET_ECR_EXTENDED_FUNCTIONS      ',.'
```

```
ETX       : 03
CRC       : 48b2
```

```
TimeStamp: 09:23:57
ACK       : 06
SEQ       : 03
```

```
TimeStamp: 09:23:57
STX       : 02
SEQ       : 04
  TAG      : 60 - PTAG_INFO Container
  TAGlen   : 17
    TAG    : 82 - PTAG_TEXT
    TAGlen  : 00
    TAGvalue : 00 00 DLE 02 04                          ',....'
```

```
    TAG    : 90 - PTAG_ECREXTENDEDFUNCTIONS
    TAGlen  : 04
    TAGvalue : 00 00 DLE 02 04                          ',....'
```

```
    TAG    : 80 - PTAG_BINARY
    TAGlen  : 01
    TAGvalue : 00                                       ',.'
```

```
    TAG    : 89 - PTAG_SEQNO
    TAGlen  : 01
    TAGvalue : DLE 03                                   ',..'
```

```
    TAG    : 85 - PTAG_FROMSTATE
    TAGlen  : 01
    TAGvalue : 07 - PSTATE_OPEN                         ',.'
```

```
    TAG    : 87 - PTAG_EVENT
    TAGlen  : 01
    TAGvalue : 1f - PEVENT_ECR_ADMIN                     ',.'
```

```

        TAG      : 86 - PTAG_STATE
        TAGlen    : 01
        TAGvalue  : 2b - PSTATE_ADMIN          ' + '

ETX      : 03
CRC      : 2192

TimeStamp: 09:23:57
ACK      : 06
SEQ      : 04

TimeStamp: 09:23:57
STX      : 02
SEQ      : 05
        TAG      : 67 - PTAG_ADMIN Container
        TAGlen    : 0f
            TAG    : 84 - PTAG_RESULT
            TAGlen  : 01
            TAGvalue: 00 - OK                  ' , '

            TAG    : 89 - PTAG_SEQNO
            TAGlen  : 01
            TAGvalue: DLE 03                  ' , , '
TAG      : 85 - PTAG_FROMSTATE
        TAGlen    : 01
        TAGvalue  : 2b - PSTATE_ADMIN          ' + '

            TAG    : 87 - PTAG_EVENT
            TAGlen  : 01
            TAGvalue: 1e - PEVENT_ECR_TRANSACTION_COMPLETED ' , '

            TAG    : 86 - PTAG_STATE
            TAGlen  : 01
            TAGvalue: 07 - PSTATE_OPEN         ' , '

ETX      : 03
CRC      : 4f55

```

Now we have the current EcrExtendedFunctions value 0x0204 and need to set the 0x0100 bit by making EcrExtendedFunctions = EcrExtendedFunctions | 0x0100 giving the result 0x0304 and we call the admin function ADMIN\_SET\_ECR\_EXTENDED\_FUNCTIONS to set it.

```

TimeStamp: 09:23:57
STX      : 02
SEQ      : 04
        TAG      : 67 - PTAG_ADMIN Container
        TAGlen    : 09
            TAG    : 88 - PTAG_COMMAND
            TAGlen  : 01
            TAGvalue: 84 - ADMIN_SET_ECR_EXTENDED_FUNCTIONS ' , '

```

```

TAG      : 90 - PTAG_ECREXTENDEDFUNCTIONS
TAGlen   : 04
TAGvalue : 00 00 DLE 03 04          ',....'

ETX      : 03
CRC      : e8f3

TimeStamp: 09:23:57
ACK      : 06
SEQ      : 04

TimeStamp: 09:23:57
STX      : 02
SEQ      : 06
TAG      : 67 - PTAG_ADMIN Container
TAGlen   : 0f
TAG      : 84 - PTAG_RESULT
TAGlen   : 01
TAGvalue : 00 - OK                  ',.'

TAG      : 89 - PTAG_SEQNO
TAGlen   : 01
TAGvalue : 04                      ',.'

TAG      : 85 - PTAG_FROMSTATE
TAGlen   : 01
TAGvalue : 2b - PSTATE_ADMIN       ',+'

TAG      : 87 - PTAG_EVENT
TAGlen   : 01
TAGvalue : 1e - PEVENT_ECR_TRANSACTION_COMPLETED ',.'

TAG      : 86 - PTAG_STATE
TAGlen   : 01
TAGvalue : 07 - PSTATE_OPEN        ',.'

ETX      : 03
CRC      : 8729

```

### 5.6.5 Tell terminal ECR to listen for extra app data

We use a special version 0xfd and a status by to tell the terminal ECR to listen for data or has stopped to listen for data from the extra apps in the terminal.

```
Status      ECR stopped listen for extra apps data
0x00
```

```
Status      ECR listen for extra apps data
0x01
```

Ex. ECR listen for extra apps data

Here app\_no 12, version 0xfd, error 0x00, last\_block 0xff and Status 0x01

```
TimeStamp: 14:53:19
STX       : 02
SEQ       : 05
  TAG      : 62 - PTAG_DATA_1 Container
  TAGlen   : 07
    TAG     : 53 - PTAG_EXTRA_APP
    TAGlen  : 05
    TAGvalue: 0c fd 00 ff 01          '.ÿ.ÿ.'
```

```
ETX       : 03
CRC       : 9918
```

```
TimeStamp: 14:53:19
ACK       : 06
SEQ       : 05
```

### 5.6.6 Example LPP Extra application data from ECR to terminal

Here app\_no 12, version 0x00, error 0x00, last\_block 0xff

```
TimeStamp: 12:33:28
STX       : 02
SEQ       : 03
  TAG      : 62 - PTAG_DATA_1 Container
  TAGlen   : 33
    TAG     : 53 - PTAG_EXTRA_APP
    TAGlen  : 31
    TAGvalue: 0c 00 00 ff 53 6f 6d 65 20 74 65 73 74 20 64 61 '...ÿSome test da'
    TAGvalue: 74 61 20 74 6f 20 45 78 74 72 61 20 74 65 72 6d 'ta to Extra term'
    TAGvalue: 69 6e 61 6c 20 61 70 70 6c 69 63 61 74 69 6f 6e 'inal application'
    TAGvalue: 00          ',.'
```

```
ETX       : 03
CRC       : 13d2
```

## 5.7 Extra application example code

We have made an extra application example, to be used for testing the basic connection; it's up to the extra application programmer to implement this setup to make the connection to/from the ECR. The ECR must be able to handle the callback and the function to send data as shown in our DLL demo (no longer supported) and any parsing/actions to be taken on the data send.

The extra application uses functions in the pointlib.so to send/receive data to/from the ECR.

```
ExtraAppNo += 12;
SYSV_init(ExtraAppNo, 0, 0); // First extra app is 12 in applic.ini
```

Used to tell the system the application number of your extra application – numbered 12 – 19, Verifone Denmark uses 21 – for special cases.

```
rc = SYSV_get_next_event( Q_APP, &queue, &Event, timeout_in_ms, &orig_id );
```

Enables the extra application to listen for events containing extra app data from the ECR.

```
SYSV_send\event(APP_MERCHANT, &event);
```

Send data from the extra application to the ECR via the merchant handler running in the terminal.

### 5.7.1 Usage: Extra application example

```
Usage: ./extra2host_msq_test ExtraApplicationNo ExtraAppVersion
ExtraAppError Extradatafile timeout_in_ms
ExtraApplicationNo legal vaules 0 - 7 and 21-NN
ExtraAppVersion    legal vaules 0 - 255
ExtraAppError      legal vaules 0 - 255
timeout_in_ms      minimum 200
```

The header is filled with ExtraApplicationNo, ExtraAppVersion, ExtraAppError

The Extradatafile contains the data put into the data field after the header bytes. If the file is larger than 1000 bytes the more\_data field in the header is set to 0x00 until less than 1000 bytes remain and more\_data is set to 0xFF.

## 6 | PointTerminal (DLL)

### 6.1 Preface

#### Revision

##### 3.7.14

CTLSTransactions is altered to always return TRUE.

Updated with new added methods:

PTInitCallback, SetCardDataResult, PrintReceiptResult, GetTokenResult, PutTokenResult, PcbStopListResult, SetExtendedAmountResult, EIEDataReceivedResult, EIEData2HostResult, PutScanBarCodeResult and PcbBreakIPResult.

Updated with new added transaction events:

SetCardData, PrintReceipt, AdviceFlag, EarlyStanPan, GetToken, PutToken, PcbStopList, PcbBreakIP, GetExtendedAmount, SetExtendedAmount, EIEDataReceived, EIEData2Host and PutScanBarCode.

Added overloaded method "SetTerminalConfiguration", enables user to configure whether the receipt should be saved or not.

LoadTerminal only tries to Connect and Open 1 time, instead of trying 3 times with Connect/Open/Close/Disconnect.

Added the possibility to close "Terminal Display" window, and improved the logic of when it should reappear.

Pressing "Try Again" on a failed Connect/Open now sends the "LoadFinished" event correctly.

Manual MobilePay transaction functionality added - See Transaction barcode.

Added support for ÆØÅ in RAW print. (ExtendedAsciiPrint.txt)

Bug in program download with ip-routing fixed.

##### 3.7.12

Xenta and Xentissimo are no longer supported.

VAS transactions receipts now get's saved in "Kvitteringer\\*VASNAME\*

Admin receipts now get's saved in "Kvitteringer\Administrative".

PostPurchase and PostRefund added.

Manual Swipp transaction functionality added - See Transaction barcode.

TCC and JsonResult fields added in extended receipt

Now it's impossible to close the PointTerminal dialog on the red cross.

Colorcode:

Additions since last release

Changes since last release

Removals since last release



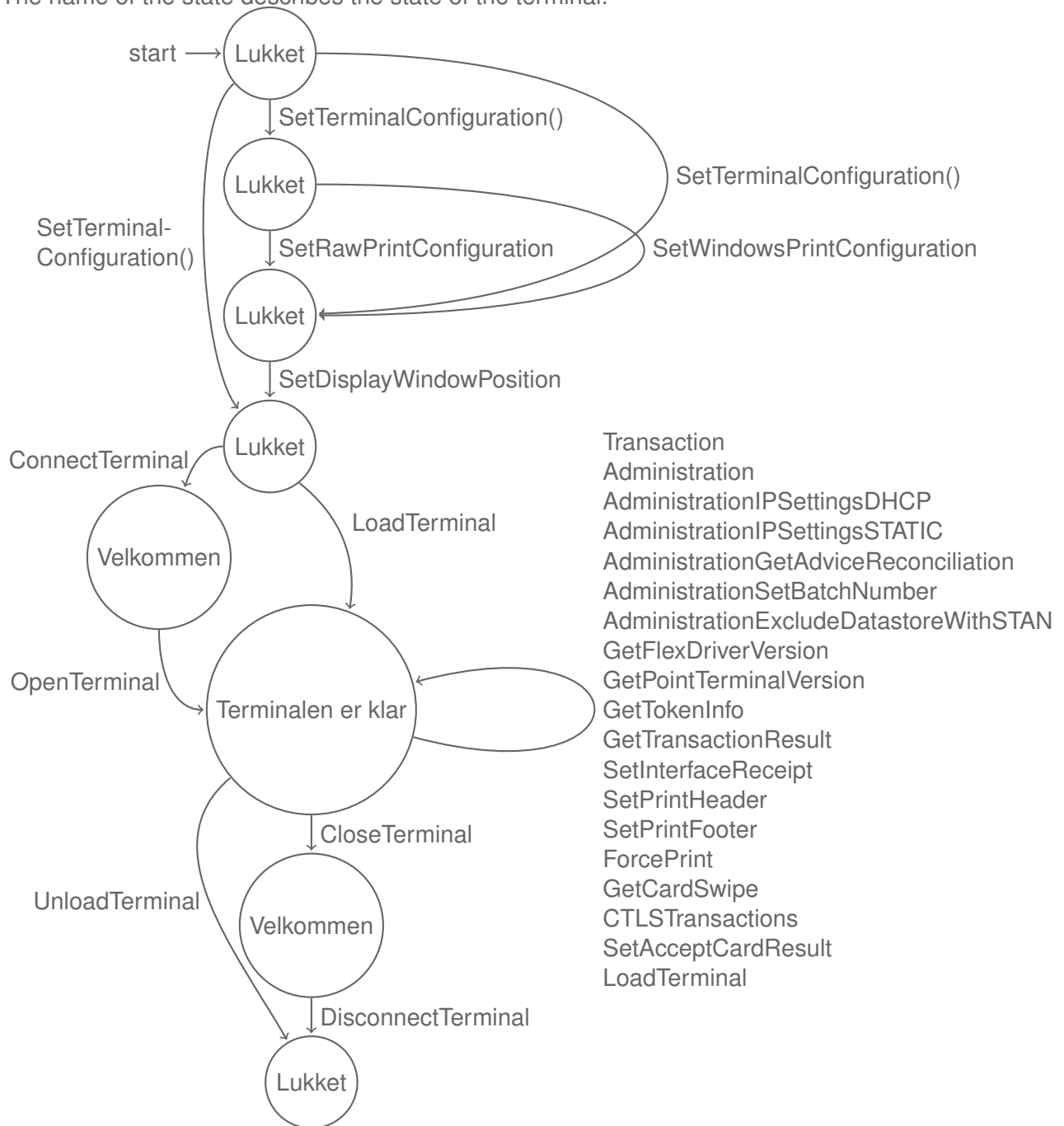
## 6.2 State machines for PointTerminalDLL

The following sections give an overview of the overall functionality and the sequences to follow when using this component.

### 6.2.1 Abstract state machine

This is an overall state machine. The machine does not handle different error scenarios but is intended for the user to have a glance at the whole setup and introduction to the general use of the DLL for the following sections.

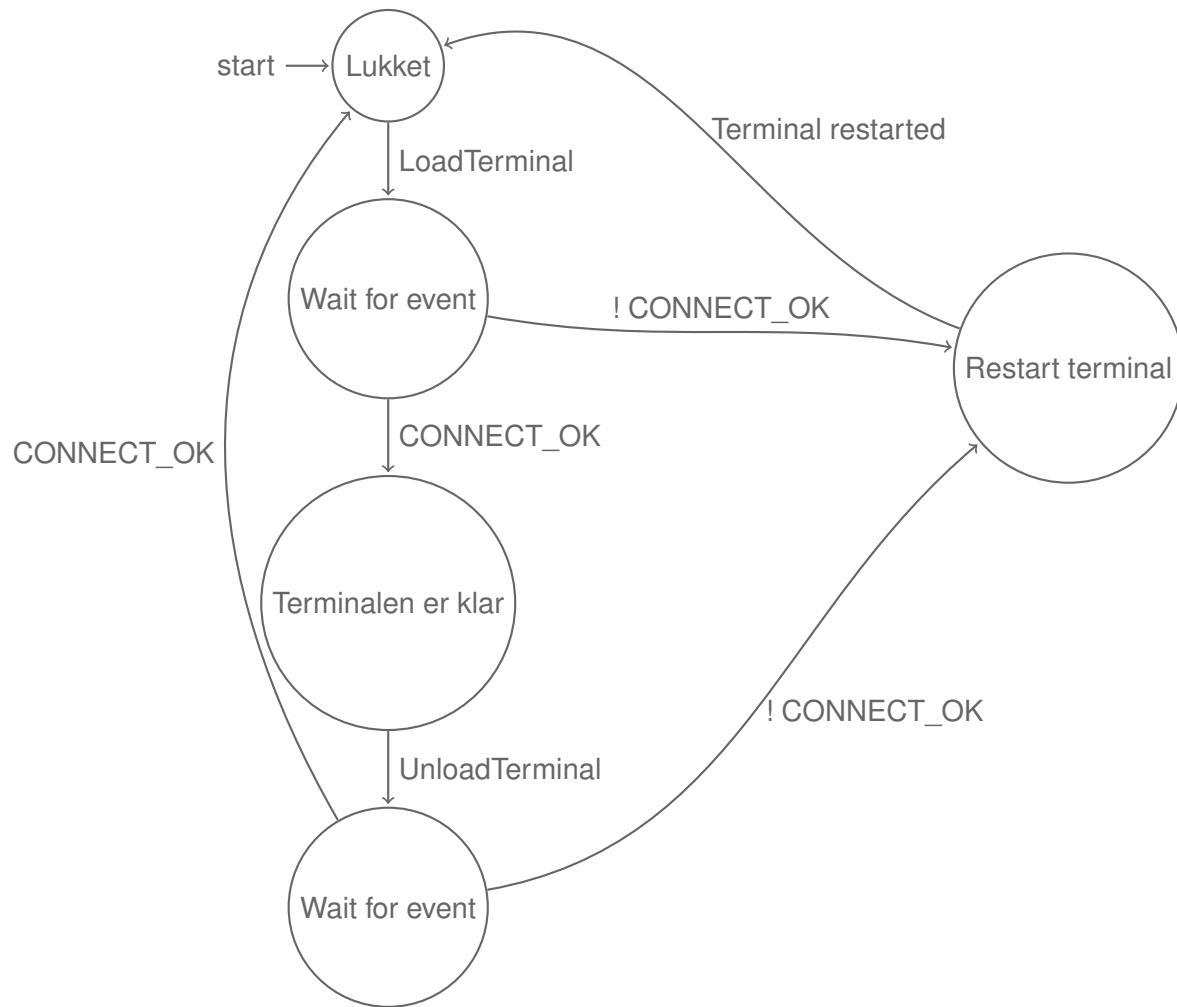
The name of the state describes the state of the terminal.





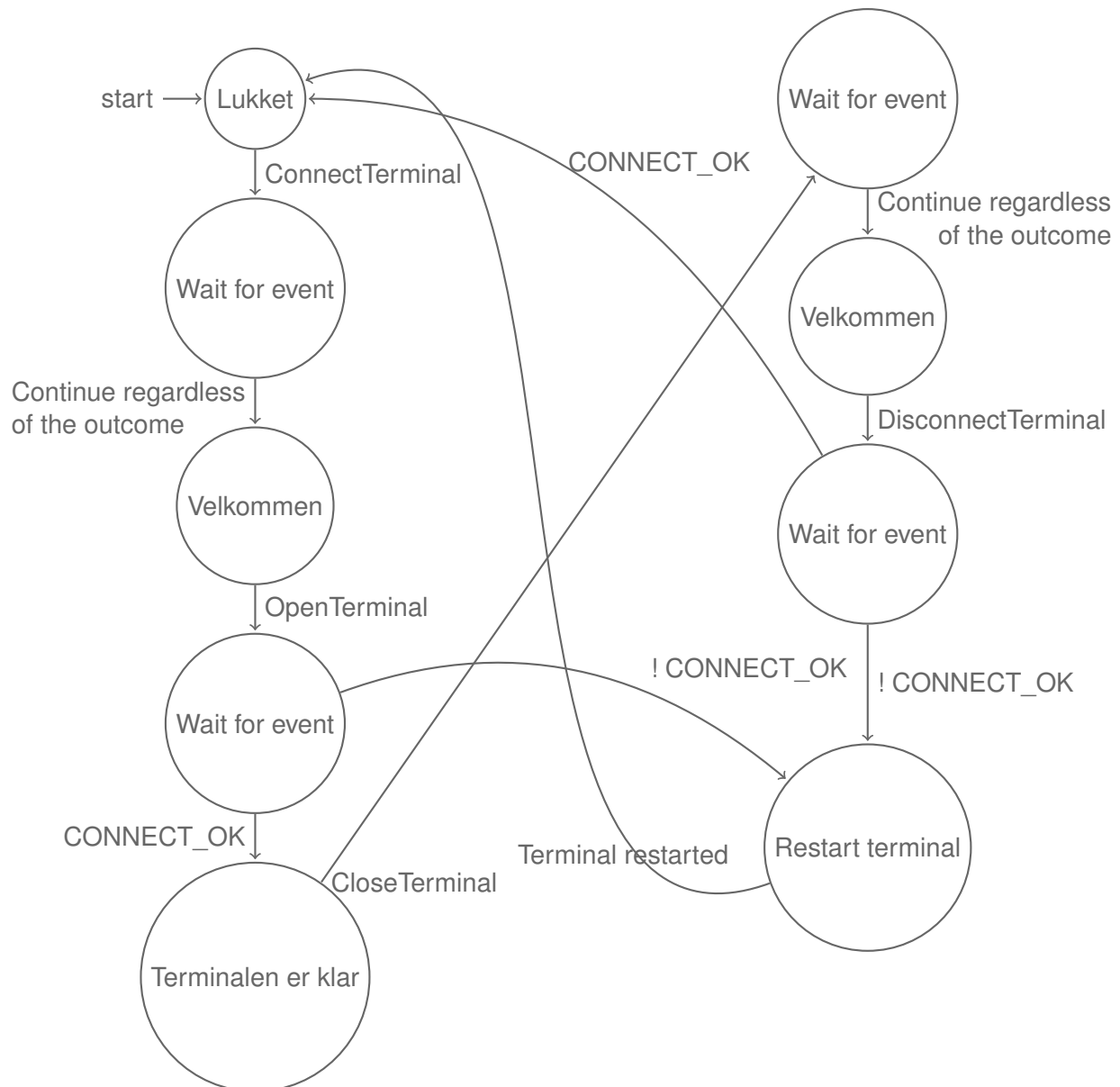
### 6.2.2 LoadTerminal and UnloadTerminal functionality

Connection state machine using the LoadTerminal and UnloadTerminal functionality.



### 6.2.3 ConnectTerminal, OpenTerminal, CloseTerminal and DisconnectTerminal functionality

Connection state machine using the ConnectTerminal, OpenTerminal, CloseTerminal and DisconnectTerminal functionality.



## 6.3 Interface

### 6.3.1 Enums

```
namespace PointTerminal
{
    internal enum RETURN_VALUES
    {
        //General return values
        SUCCESS = 0x00,
        FAILURE = 0x01,
        FILE_ERROR = 0x02,
        DATALINKERROR = 0x07,
        FUNCTION_NOT_POSSIBLE = 0x08,
        TIMEOUT_IN_COMMUNICATION = 0x09,
        CONNECT_NO_LICENCE = 0x0A,
        CONNECT_INTERNAL_ERROR = 0x0B,
        CONNECT_SYSTEM_ERROR = 0x0C,
        LOCALCARD_OK = 0x10,
        LOCALCARD_NOT_OK = 0x11,
        RETURN_INIT_VALUE = 0xFFFF,
    }

    internal enum PRERESULT_RETURN_VALUES
    {
        SUCCESS = 0x00,
        SUCCESS_SIGNATURE_ACCEPTED = 0x01,
        REJECTED_BY_NETS = 0x80,
        REJECTED_SIGNATURE = 0x81,
        REJECTED = 0x82,
    }

    internal enum CONNECT_RETURN_VALUES
    {
        //Return values related to connecting
        CONNECT_OK = 0x00,
        CONNECT_COMPORTINITFAILURE,
        CONNECT_SW_NOTCOMPATIBLE,
        CONNECT_TERM_NOT_RESP,
        CONNECT_COMPORTOPEN,
        DATA_LINK_ERROR,
        FUNCTION_NOT_POSSIBLE,
        TIMEOUT_IN_COMMUNICATION,
        CONNECT_NO_LICENCE,
        CONNECT_INTERNAL_ERROR,
        CONNECT_SYSTEM_ERROR,
        OPEN_NO_RECEIPT,
        CONNECT_INIT_VALUE,
    };

    internal enum FLX_TRACE_LEVEL
    {
        FLX_CONF_PARAM = 1, //Placement of inifile
    }
}
```

```

    FLX_CONF_TRACE, //Enable trace
    FLX_CONF_EXTTRACE, //Enable extensive tracing
    FLX_CONF_EXTTRACE_PLUS, //Enable extensive tracing with append
    FLX_NO_TRACE_AT_ALL,
};

internal enum COM_TYPE
{
    RS232 = 1,
    IP = 2,
}

public enum FLX_CALLBACK
{
    FLX_CALLBACK_SET_CARDDATA = 100, // this gets the _flxxxxVB@nn entries in dll
    FLX_CALLBACK_PRINT_RECEIPT,
    FLX_CALLBACK_DISPLAY_STATUS,
    FLX_CALLBACK_ABORT,
    FLX_CALLBACK_EXTCARDCONFIRM,
    FLX_CALLBACK_VERSIGCONFIRM, // 5
    FLX_CALLBACK_ADVICEFLAG,
    FLX_CALLBACK_SET_AMOUNT, // send amount to terminal from ecr
    FLX_CALLBACK_SET_AMOUNT_FEE, // send fee to terminal from ecr
    FLX_CALLBACK_GET_AMOUNT_FEE, // receive fee from terminal to ecr
    FLX_CALLBACK_STATUSTRACE, // 10
    FLX_CALLBACK_STATETRACE,
    FLX_CALLBACK_ADVICELOG,
    FLX_CALLBACK_MENU,
    FLX_CALLBACK_MENURESULT,
    FLX_CALLBACK_GET_BINARY_RECEIPT, // 15
    FLX_CALLBACK_EARLY_STAN_PAN,
    FLX_CALLBACK_BREAK_IP,
    FLX_CALLBACK_TIMER,
    FLX_CALLBACK_PUT_TOKEN,
    FLX_CALLBACK_GET_TOKEN, // 20
    FLX_CALLBACK_SET_AMOUNT GRATUITY, // from here is new from TAPA3 - send gratuity to terminal from ecr
    FLX_CALLBACK_STOPLIST,
    FLX_CALLBACK_PRE_RESULT,
    FLX_CALLBACK_SET_ECR_EXTENDED_FUNCTIONS,
    FLX_CALLBACK_HOSTADVICE, // 25

    FLX_CALLBACK_SET_EXTENDED_AMOUNT = 127,
    FLX_CALLBACK_GET_EXTENDED_AMOUNT,
    FLX_CALLBACK_EIE_DATA_RECEIVED,
    FLX_CALLBACK_EIE_DATA2HOST, // 30
    FLX_CALLBACK_PUT_SCAN_BAR_CODE,
    FLX_CALLBACK_EXTRA_APP_DATA_RECEIVED,
    FLX_CALLBACK_CARD_SWIPE,
    FLX_CALLBACK_PRINT_STATUS,
};

// ADMINISTRATION FUNCTIONS
internal enum FLX_ADMIN_FUNCTION
{

```

```

ADMIN_ENDOFDAY = 1,
ADMIN_ENDOFDAYLOG,
ADMIN_REPORT_TERMINALREPORT,
ADMIN_REPORT_TOTALS,
ADMIN_REPORT_LOG,      // 5
ADMIN_REPORT_OLDLOG,
ADMIN_LASTRECEIPT,
ADMIN_UNLOCK_RECEIPT,
ADMIN_CLOCKSYNCPBS,
ADMIN_CLOCKSYNCPPOINT, //10
ADMIN_SENDLOG,
ADMIN_CLEARDATASTORE,
ADMIN_DOWNLOADPROGRAM,
ADMIN_DOWNLOADPARAM,
ADMIN_DOWNLOADPAN,     //15
ADMIN_DOWNLOADTLCMDB,
ADMIN_RESTORETLCMDB,
ADMIN_CONTRASTUP,
ADMIN_CONTRASTDOWN,
ADMIN_RESTARTTERMINAL, //20
ADMIN_EJECTCARD,
ADMIN_MSC,              // For future use (fetch the card range table)
ADMIN_BACKLIGHT_ON,
ADMIN_BACKLIGHT_OFF,
ADMIN_NETWORK_REPORT,  // 25
ADMIN_RATES_REPORT,
ADMIN GRATUITY_RECEIPT,
ADMIN_EXCLUDE_DATASTORE_RECORD_WITH_STAN, // for internal use - uses flxTerminalProperties
ADMIN_ADVICEFORWARDING,
ADMIN_REPORT_FILE5STATUS,
ADMIN_RESERVED_FOR_OCX = 31,
ADMIN_REPORT_PCT,
ADMIN_DOWNLOADDCCRATES,
ADMIN_UPDATEPSAM,
ADMIN_UPDATEFEETABLE,  // 35
ADMIN_UPDATESALT,
ADMIN_GETADVICERECON,
ADMIN_SET_BATCH_NUMBERS, // for internal use - use flxTerminalProperties
ADMIN_REPORT_TCS,
ADMIN_REPORT_TPROPS_CVS, // 40
ADMIN_EVENTLOG_PRINT,
ADMIN_EVENTLOG_SEND,    // for internal use - uses flxTerminalProperties
ADMIN_EVENTLOG_DELETE,  // for internal use - uses flxTerminalProperties
ADMIN_GETIP_SETTINGS,
ADMIN_SETIP_SETTINGS,   // 45 for internal use - uses flxTerminalProperties
ADMIN_DOWNLOAD_IMAGES,  //
ADMIN_CHECK_CARD,
ADMIN_GETTELEDONE_SETTINGS,
ADMIN_SETTELEDONE_SETTINGS,
ADMIN_CTL5_REPORT, //50
ADMIN_USER_INPUT,
ADMIN_MAX,             // last entry
ADMIN_GET_DC_PROPERTIES_STAN = 0x80,
};

```

```

// TRANSACTION METHODS
internal enum FLX_TRANSTYPE
{
    FLX_TRANS_PURCHASE = 0,      // Debit transaction
    FLX_TRANS_REFUND,           // Credit transaction
    FLX_TRANS_ORIGINAL_AUTH,     // Original Authorization, outputs a token
    FLX_TRANS_SUPP_AUTH,        // Token based transaction
    FLX_TRANS_CAPTURE,          // Token based transaction
    FLX_TRANS_REVERSAL_AUTH,     // Token based transaction
    FLX_TRANS_PREPAID_PURCHASE,  // Debit prepaid transaction
    FLX_TRANS_PREPAID_REFUND,    // Credit prepaid transaction
    FLX_TRANS_PREPAID_ORIGINAL_AUTH, // Orig auth prepaid transaction, no token output
    FLX_TRANS_PREPAID_SCAN_PURCHASE, // Debit prepaid scan transaction
    FLX_TRANS_PREPAID_SCAN_REFUND, // Credit prepaid scan transaction
    FLX_TRANS_PREPAID_SCAN_ORIGINAL_AUTH, // Orig auth prepaid scan transaction, no token output
    FLX_TRANS_CANCELLATION,     // Cancellation transaction
    FLX_TRANS_CARDCHECK,        // card check
    FLX_TRANS_POST_PURCHASE,     // Post purchase transaction
    FLX_TRANS_POST_REFUND       // Post refund transaction
};

internal enum FLX_KEY_ENTRY
{
    KEY_ENTRY = 0x80
}

//CURRENCY CODES
public enum FLX_CURR_CODES
{
    DKK = 208, //DK Kroner
    ISK = 352, //IS Kroner
    JPY = 392, //JP Yen
    NOK = 578, //N Kroner
    SEK = 752, //S Kroner
    CHF = 756, //SW Francs
    GBP = 826, //GB Pound
    USD = 840, //US Dollar
    EUR = 978, // Euro
}

internal enum FLX_MERCHANT_INITIATIVE // As Defined by PBS
{
    MI_PIN = 0x00,
    MI_ONLINE = 0x50,
    MI_OFFLINE = 0x60,
    MI_FORCED_CVM = 0x80,
    MI_FORCED_PIN = 0x81,
    MI_FORCED_SIGNATURE = 0x82,
}

public enum FLX_RECEIPT_CALLBACK_RETURN_VALUE
{

```

```

    FLX_RECEIPT_NOTOK = 0,
    FLX_RECEIPT_OK
}

public enum FLX_CARDNUMBER_CMD
{
    FLX_CARDNUMBER_NOTOK = 0,
    FLX_CARDNUMBER_OK,
    FLX_CARDNUMBER_OK_CONFIRM = 3, //External card confirm ...
    FLX_EXTRA_APPLICATION_ACTION_0,
    FLX_EXTRA_APPLICATION_ACTION_1,
    FLX_EXTRA_APPLICATION_ACTION_2,
    FLX_EXTRA_APPLICATION_ACTION_3,
    FLX_EXTRA_APPLICATION_ACTION_4,
    FLX_EXTRA_APPLICATION_ACTION_5,
    FLX_EXTRA_APPLICATION_ACTION_6,
    FLX_EXTRA_APPLICATION_ACTION_7,
    FLX_EXTRA_APPLICATION_ACTION_8,
    FLX_EXTRA_APPLICATION_ACTION_9,
    FLX_EXTRA_APPLICATION_ACTION_10,
    FLX_EXTRA_APPLICATION_ACTION_11,
    FLX_EXTRA_APPLICATION_ACTION_12,
    FLX_EXTRA_APPLICATION_ACTION_13,
    FLX_EXTRA_APPLICATION_ACTION_14,
    FLX_EXTRA_APPLICATION_ACTION_15,
    FLX_EXTRA_APPLICATION_ACTION_16,
    FLX_CARDNUMBER_OK_SILENTABORT = 0x80,
}

public enum FLX_ABORT_CMD
{
    FLX_OPERATORWISHTOABORT = 1,
    FLX_OPERATORDONOTWISHTOABORT
}

public enum FLX_SIGNATURE
{
    SIGNATURE_NOTOK,
    SIGNATURE_OK,
    SIGNATURE_NOTKNOWNYET
}

public enum FLX_CARDTYPE
{
    PBS_CARD,
    OTHER_CARD
}

internal enum FLX_PROPSCOMMAND
{
    ADMIN_PROPS_NONE = 0,
    ADMIN_PROPS_GET,
    ADMIN_PROPS_PREPECEIPT,

```

```

    ADMIN_PROPS_DATASTORE,
    ADMIN_PROPS_SET,
    ADMIN_PROPS_ADVICERECON,
    ADMIN_PROPS_BATCH,
    ADMIN_PROPS_GET_LABELS,
    ADMIN_PROPS_EVENTLOG_SEND,
    ADMIN_PROPS_EVENTLOG_DELETE,
    ADMIN_PROPS_SETIP_SETTINGS,
    ADMIN_PROPS_GET_EXT_LABELS,
    ADMIN_PROPS_SETTELEDONE_SETTINGS
}

internal enum FLX_ECREXTCMD
{
    ADMIN_SET_ECR_EXTENDED_FUNCTIONS = 132,
    ADMIN_GET_ECR_EXTENDED_FUNCTIONS = 133,
}

public enum FLX_AMOUNT_TYPE
{
    FLX_AMOUNT_AMOUNT,
    FLX_AMOUNT_FEE,
    FLX_AMOUNT GRATUITY,
    FLX_AMOUNT_VAT,
    FLX_AMOUNT_BACK
}

internal enum CONNECTION_STATE
{
    DISCONNECTED = 0x00,
    CONNECTED,
    OPEN,
    OPEN_NP, // No receipt - INGEN KVITTERING
}

internal enum FONTSTYLE
{
    NORMAL,
    KURSIV,
    FED,
    FEDKURSIV
}

internal enum TERMINALTYPE
{
    PSAM,
    FOS5,
    SIRIUS
}

internal enum DISPLAY_STATE
{
    MESSAGE,
    RECONCILATION,
    VERIFYSIGNATURE,

```



```

        GODKENDLOCALCARD,
        AUTHORIZATIONCODE,
        MENU
    }

    internal enum DISPLAY_ANSWER
    {
        JA,
        NEJ,
        IKKEKLARENDNU
    }

    internal enum CARDSWIPE
    {
        CARD_SWIPE = 0x20,
        CARD_REMOVED = 0x21,
        TEDO = 0x5465446f // teleload done
    }

    internal enum PRINT_TYPE
    {
        SCREEN,
        WINDOWS,
        RAW_EPSON,
        RAW_AXIOHM,
        RAW_IBM,
        RAW_BIXOLON
    }

    internal enum CONF_E
    {
        CONF_PARAM = 1, // Placement of inifile
        CONF_TRACE, // Enable trace
        CONF_EXTTRACE, // Enable extensive tracing
        CONF_EXTTRACE_PLUS, // Enable extensive tracing with append
        CONF_FILEPATH, // Define directory path for flxdrv.ini and trace files
        CONF_KISSPARAM,
        CONF_NO_TRACE_AT_ALL = 16,
        CONF_RESETPORT,
        CONF_GUARDTIME,
        CONF_RECEIPT_WIDTH,
        CONF_PIP_TIMEOUT,
        CONF_DOWNLOAD_TIMEOUT,
        CONF_DOWNLOAD_BLOCK_SIZE,
        CONF_DOWNLOAD_TIMEOUT_NEXT,
        CONF_DOWNLOAD_BLOCK_SIZE_NEXT,
        CONF_DOWNLOAD_METODE,
        CONF_IP_ROUTING_RECV_BUFFER_SIZE,
        CONF_USE_GETTIMEOFDAY,
    }

    public enum PT_CALLBACK
    {
        PT_CALLBACK_SET_EXTENDED_AMOUNT,
    }

```

```
}  
}
```

### 6.3.2 Methods

The methods of PointTerminal.dll.

These methods expose all integration functionality in the terminal.

For rules regarding the use of functionality described in this chapter please consult the ORTS delivered by Nets.

#### 6.3.2.1 SetTerminalConfiguration(7)

SetTerminalConfiguration sets the configuration for the terminal.

That is:

- The terminal type
- Communication type
- Com port or IP address
- Baud rate or TCP/IP port
- Tracelevel on the POS
- Token deletion after transaction
- Saving of receipts in folder

Method: SetTerminalConfiguration -

**Prototype: PointTerminalUsercontrol.SetTerminalConfiguration(**

**Terminal type As int,**  
**Communication type As int,**  
**COM port or IP address As string,**  
**Baud rate or TCP/IP port as uint,**  
**Flex trace level as int,**  
**Token as bool,**  
**Saving of receipts in folder as bool);**

Input:

- Terminal type - terminal type used for the integration. Enum: TERMINALTYPE
- Communication type - IP or RS232 communication. Enum: COM\_TYPE
- COM port or IP address - example: "COM8" or "10.0.0.48"
- Baud rate or TCP/IP port - example: 38400 or 2000
- Flex trace level - tracelevel on the POS. Enum: FLX\_TRACE\_LEVEL
- Token deletion after transaction - Used if post purchase/refund is used.
- Saving of receipts in folder

Return: as boolean

- true - arguments were accepted and configuration set correctly
- false - arguments were not accepted. Configuration not set

**Fire event: -**

### 6.3.2.2 SetTerminalConfiguration(6)

SetTerminalConfiguration sets the configuration for the terminal.

That is:

- The terminal type
- Communication type
- Com port or IP address
- Baud rate or TCP/IP port
- Tracelevel on the POS
- Token deletion after transaction
- Saving of receipts in folder

Method: SetTerminalConfiguration -

**Prototype: PointTerminalUsercontrol.SetTerminalConfiguration(  
Terminal type As int,  
Communication type As int,  
COM port or IP address As string,  
Baud rate or TCP/IP port as uint,  
Flex trace level as int,  
Token as bool);**

Saving of receipts in folder is set to true.

Input:

- Terminal type - terminal type used for the integration. Enum: TERMINALTYPE
- Communication type - IP or RS232 communication. Enum: COM\_TYPE
- COM port or IP address - example: "COM8" or "10.0.0.48"
- Baud rate or TCP/IP port - example: 38400 or 2000
- Flex trace level - tracelevel on the POS. Enum: FLX\_TRACE\_LEVEL
- Token deletion after transaction - Used if post purchase/refund is used.

Return: as boolean

- true - arguments were accepted and configuration set correctly
- false - arguments were not accepted. Configuration not set

**Fire event: -**

### 6.3.2.3 SetTerminalConfiguration(5)

SetTerminalConfiguration sets the configuration for the terminal.

That is:

- The terminal type
- Communication type
- Com port or IP address
- Baud rate or TCP/IP port
- Tracelevel on the POS
- Token deletion after transaction
- Saving of receipts in folder

Method: SetTerminalConfiguration -

**Prototype: PointTerminalUsercontrol.SetTerminalConfiguration(  
Terminal type As int,  
Communication type As int,  
COM port or IP address As string,  
Baud rate or TCP/IP port as uint,  
Flex trace level as int);**

Token is set to true. Saving of receipts in folder is set to true.

Input:

Terminal type - terminal type used for the integration. Enum: TERMINALTYPE

Communication type - IP or RS232 communication. Enum: COM\_TYPE

COM port or IP address - example: "COM8" or "10.0.0.48"

Baud rate or TCP/IP port - example: 38400 or 2000

Flex trace level - tracelevel on the POS. Enum: FLX\_TRACE\_LEVEL.

Return: as boolean

true - arguments were accepted and configuration set correctly

false - arguments were not accepted. Configuration not set

**Fire event: -**

#### 6.3.2.4 SetWindowsPrintConfiguration

SetWindowsPrintConfiguration sets the configuration for the windows printer used.

Method: SetWindowsPrintConfiguration -

**Prototype: PointTerminalUsercontrol.SetWindowsPrintConfiguration(  
Windows printer As string,  
Windows print font As string,  
Windows print font style As int,  
Windows print font size As int,  
Windows print top margin As uint,  
Windows print left margin As uint);**

Input:

Windows printer - the name of the printer.

Windows print font - the name of the font.

Windows print font style - the font style. Enum FONTSTYLE can be used.

Windows print font size - the font size.

Windows print top margin - the top margin on the receipt.

Windows print left margin - the left margin on the receipt.

Return: as boolean

true - arguments were accepted

false - arguments were not accepted

**Fire event: -**

### 6.3.2.5 SetRawPrintConfiguration

SetRawPrintConfiguration sets the configuration for the raw printer used.

Method: SetRawPrintConfiguration -

**Prototype: PointTerminalUsercontrol.SetRawPrintConfiguration(Printer As string);**

Input:

Printer - the name of the printer.

Return: as boolean

true - argument were accepted

false - argument were not accepted

**Fire event: -**

### 6.3.2.6 SetDisplayWindowPosition

SetDisplayWindowPosition sets the position of the display window delivered by PointTerminal.dll.

Method: SetDisplayWindowPosition -

**Prototype: PointTerminalUsercontrol.SetDisplayWindowPosition(Display top position As uint, Display left position As uint);**

Input:

Display top position - the vertical position of the display window.

Display left position - the horizontal position of the display window.

Return: as boolean

true - arguments were accepted

false - arguments were not accepted

**Fire event: -**

### 6.3.2.7 LoadTerminal

LoadTerminal connects and opens the terminal.

This method is intended to replace use of the ConnectTerminal and OpenTerminal.

Method: LoadTerminal -

**Prototype: PointTerminalUsercontrol.LoadTerminal();**

Input: -

Return: as void

**Fire event: LoadFinished, TerminalDataFinished if connect and open was successful**

### 6.3.2.8 UnloadTerminal

UnloadTerminal closes and disconnects the terminal. This method is intended to replace use of the CloseTerminal and OpenTerminal methods.

Method: UnloadTerminal -

**Prototype: PointTerminalUsercontrol.UnloadTerminal();**

Input: -

Return: as void

**Fire event: UnloadFinished**

#### **6.3.2.9 ConnectTerminal**

ConnectTerminal establishes a connection to the terminal. The terminal goes to 'Velkommen' if the method call was succesfull.

Method: ConnectTerminal -

**Prototype: PointTerminalUsercontrol.ConnectTerminal();**

Input: -

Return: as void

**Fire event: ConnectFinished**

#### **6.3.2.10 OpenTerminal**

OpenTerminal opens the terminal so it is ready to do transactions. The terminal goes to 'Terminalen er klar' if the method call was succesfull.

Method: OpenTerminal -

**Prototype: PointTerminalUsercontrol.OpenTerminal();**

Input: -

Return: as void

**Fire event: OpenFinished, TerminalDataFinished if the open was succesfull**

#### **6.3.2.11 CloseTerminal**

CloseTerminal closes the terminal. The terminal goes to 'Velkommen' if the method call was succesfull.

Method: CloseTerminal -

**Prototype: PointTerminalUsercontrol.CloseTerminal();**

Input: -

Return: as void

**Fire event: CloseFinished**

#### **6.3.2.12 DisconnectTerminal**

DisconnectTerminal disconnects and releases the connection to the terminal. The terminal goes to 'Lukket' if the method call was succesfull.

Method: DisconnectTerminal -

**Prototype: PointTerminalUsercontrol.DisconnectTerminal();**

Input: -

Return: as void

**Fire event: DisconnectFinished**

### 6.3.2.13 Transaction(13)

Starts a transaction.

Method: Transaction -

**Prototype: PointTerminalUsercontrol.Transaction(  
Transaction type As int,  
Transaction amount As uint,  
Fee amount As uint,  
Gratuity amount As uint,  
VAT amount As uint,  
Cash back amount As uint,  
Currency As int,  
Merchant initiative As int,  
Barcode As string,  
Key entry As boolean,  
Token path As string,  
Reference number As uint,  
Authorization code As string);**

Input:

Transaction type - the type of the transaction. Enum: FLX\_TRANSTYPE.

Transaction Amount - the amount without fee, gratuity, VAT and cashback.

Fee amount - fee amount for the transaction.

Gratuity amount - gratuity amount for the transaction.

VAT amount - VAT amount for the transaction.

Cash back amount - cash back amount for the transaction.

Currency - the currency for the transaction. Enum: FLX\_CURR\_CODES.

Merchant initiative - MI for the transaction. Enum: FLX\_MERCHANT\_INITIATIVE.

Barcode - for barcode transactions.

For manuel Swipp transaction: "sw:\*landcode\*\*phonenummer\* e.g. "sw:4530303030"

Key entry - enable key entry for manual entry of the card number.

Token path - full path for the token for token transactions.

Reference number - internal reference number for the transaction.

Authorization - auth code for offline transactions.

Return: as Boolean

true - arguments were accepted

false - arguments were not accepted

**Fire event: TransactionFinished**

#### 6.3.2.14 Transaction(11)

Starts a transaction.

Reference number set to 0.

Authorization set to an empty string.

Method: Transaction -

**Prototype: PointTerminalUsercontrol.Transaction(**

**Transaction type As int,**  
**Transaction Amount As uint,**  
**Fee amount As uint,**  
**Gratuity amount As uint,**  
**VAT amount As uint,**  
**Cash back amount As uint,**  
**Currency As int,**  
**Merchant initiative As int,**  
**Barcode As string,**  
**Key entry As boolean,**  
**Token path As string);**

Input:

Transaction type - the type of the transaction. Enum: FLX\_TRANSTYPE.

Transaction Amount - the amount without fee, gratuity, VAT and cashback.

Fee amount - fee amount for the transaction.

Gratuity amount - gratuity amount for the transaction.

VAT amount - VAT amount for the transaction.

Cash back amount - cash back amount for the transaction.

Currency - the currency for the transaction. Enum: FLX\_CURR\_CODES.

Merchant initiative - MI for the transaction. Enum: FLX\_MERCHANT\_INITIATIVE.

Barcode - for barcode transactions.

For manuel Swipp transaction: "sw:\*landcode\*\*phonenummer\* e.g. "sw:4530303030"

Key entry - enable key entry for manual key entry of the card number.

Token path - full path for the token for token transactions.

Return: as Boolean

true - arguments were accepted

false - arguments were not accepted

**Fire event: TransactionFinished**

#### 6.3.2.15 Transaction(8)

Starts a transaction.

Barcode set to an empty string.

Key entry set to false.



Token path set to an empty string.  
Reference number set to 0.  
Authorization set to an empty string.

Method: Transaction -

**Prototype: PointTerminalUsercontrol.Transaction(**  
    **Transaction type As int,**  
    **Transaction Amount As uint,**  
    **Fee amount As uint,**  
    **Gratuity amount As uint,**  
    **VAT amount As uint,**  
    **Cash back amount As uint,**  
    **Currency As int,**  
    **Merchant initiative As int);**

Input:

Transaction type - the type of the transaction. Enum: FLX\_TRANSTYPE.  
Transaction Amount - the amount without fee, gratuity, VAT and cashback.  
Fee amount - fee amount for the transaction.  
Gratuity amount - gratuity amount for the transaction.  
VAT amount - VAT amount for the transaction.  
Cash back amount - cash back amount for the transaction.  
Currency - the currency for the transaction. Enum: FLX\_CURR\_CODES.  
Merchant initiative - MI for the transaction. Enum: FLX\_MERCHANT\_INITIATIVE.

Return: as Boolean

    true - arguments were accepted  
    false - arguments were not accepted

**Fire event: TransactionFinished**

#### 6.3.2.16 Transaction(5)

Starts a transaction.

Fee amount set to 0.  
Gratuity amount set to 0.  
VAT amount set to 0.  
Barcode set to an empty string.  
Key entry set to false.  
Token path set to an empty string.  
Reference number set to 0.  
Authorization set to an empty string.

Method: Transaction -

**Prototype: PointTerminalUsercontrol.Transaction(**  
    **Transaction type As int,**

**Transaction Amount As uint,  
Cash back amount As uint,  
Currency As int,  
Merchant initiative As int);**

Input:

Transaction type - the type of the transaction. Enum: FLX\_TRANSTYPE.  
Transaction Amount - the amount without fee, gratuity, VAT and cashback.  
Cash back amount - cash back amount for the transaction.  
Currency - the currency for the transaction. Enum: FLX\_CURR\_CODES.  
Merchant initiative - MI for the transaction. Enum: FLX\_MERCHANT\_INITIATIVE.

Return: as Boolean

true - arguments were accepted

false - arguments were not accepted **Fire event: TransactionFinished**

### **6.3.2.17 Transaction(3)**

Starts a transaction.

Currency set to DKK.

Merchant initiative set to online PIN (default).

Fee amount set to 0.

Gratuity amount set to 0.

VAT amount set to 0.

Barcode set to an empty string.

Key entry set to false.

Token path set to an empty string.

Reference number set to 0.

Authorization set to an empty string.

Method: Transaction -

**Prototype: PointTerminalUsercontrol.Transaction(**

**Transaction type As int,  
Transaction Amount As uint,  
Cash back amount As uint);**

Input:

Transaction type - the type of the transaction. Enum: FLX\_TRANSTYPE.  
Transaction Amount - the amount without fee, gratuity, VAT and cashback.  
Cash back amount - cash back amount for the transaction.

Return: as Boolean

true - arguments were accepted

false - arguments were not accepted

**Fire event: TransactionFinished**

### 6.3.2.18 Transaction(1)

Starts a transaction.

Transaction type set to purchase (default).

Cash back amount set to 0.

Currency set to DKK.

Merchant initiative set to online PIN (default).

Fee amount set to 0.

Gratuity amount set to 0.

VAT amount set to 0.

Barcode set to an empty string.

Key entry set to false.

Token path set to an empty string.

Reference number set to 0.

Authorization set to an empty string.

Method: Transaction -

**Prototype: PointTerminalUsercontrol.Transaction(  
Transaction Amount As uint);**

Input:

Transaction Amount - the amount without fee, gratuity, VAT and cashback.

Return: as Boolean

true - argument were accepted

false - argument were not accepted

**Fire event: TransactionFinished**

### 6.3.2.19 Refund

Starts a refund in DKK.

Method: Refund -

**Prototype: PointTerminalUsercontrol.Refund(  
Transaction Amount As uint);**

Input: Refund Amount - the amount in DKK without fee, gratuity, VAT and cashback.

Return: as Boolean

true - arguments were accepted

false - arguments were not accepted

**Fire event: TransactionFinished**

### 6.3.2.20 Cancellation

Starts a cancellation of the previous transaction.

Method: Cancellation -

**Prototype: PointTerminalUsercontrol.Cancellation();**

Input: -

Return: as Boolean

    true - arguments were accepted and cancellation started correctly.

    false - arguments were not accepted

**Fire event: TransactionFinished**

#### 6.3.2.21 Administration

Starts an administration function on the terminal.

For administration functions with arguments this method cannot be used.

For ADMIN\_SETIP\_SETTINGS use the method AdministrationIPSettingsDHCP() or AdministrationIPSettingsSTATIC().

For ADMIN\_GETADVICERECON use the method AdministrationGetAdviceReconciliation().

For ADMIN\_SET\_BATCH\_NUMBERS use the method AdministrationSetBatchNumber().

For ADMIN\_EXCLUDE\_DATASTORE\_RECORD\_WITH\_STAN use the method AdministrationExcludeDatastoreWithSTAN().

Method: Administration -

**Prototype: PointTerminalUsercontrol.Administration(  
Administration function number As int);**

Input: Administration function number - Enum FLX\_ADMIN\_FUNCTION can be used.

Return: as Boolean

    true - argument were accepted.

    false - argument were not accepted

**Fire event: AdministrationFinished**

#### 6.3.2.22 AdministrationIPSettingsDHCP

Starts the administration function on the terminal, setting the terminal to use DHCP.

Method: AdministrationIPSettingsDHCP -

**Prototype: PointTerminalUsercontrol.AdministrationIPSettingsDHCP();**

Input: -

Return: as Boolean

    true - administration function started correctly.

    false - administration function not started correctly.

**Fire event: AdministrationFinished**

#### 6.3.2.23 AdministrationIPSettingsSTATIC

Starts the administration function on the terminal, setting the terminal to use a static IP address.

Method: AdministrationIPSettingsSTATIC -

**Prototype: PointTerminalUsercontrol.AdministrationIPSettingsSTATIC(  
IP address As string,**

**Subnet As string,  
Gateway As string,  
DNS1 As string,  
DNS2 As string,  
Domain As string);**

Input:

IP address - IP address of the terminal.  
Subnet - the subnet that the terminal is connected to.  
Gateway - the gateway the terminal should use.  
DNS1 - the first DNS the terminal should use.  
DNS2 - the second DNS the terminal should use.  
Domain - the domain the terminal belongs to.

Return: as Boolean

true - arguments accepted and administration function started correctly.  
false - administration function not started correctly.

**Fire event: AdministrationFinished**

#### **6.3.2.24 AdministrationGetAdviceReconciliation**

Starts the administration function on the terminal getting the reconciliation for {argument} days back.

Method: AdministrationGetAdviceReconciliation -

**Prototype: PointTerminalUsercontrol.AdministrationGetAdviceReconciliation(  
Days back As uint);**

Input: Days back - days back from current date that the reconciliation should be.

Return: as Boolean

true - arguments were accepted and administration function was started.  
false - arguments were not accepted and administration function not started.

**Fire event: AdministrationFinished**

#### **6.3.2.25 AdministrationSetBatchNumber**

Starts the administration function on the terminal, setting the batch number.

Method: AdministrationSetBatchNumber -

**Prototype: PointTerminalUsercontrol.AdministrationSetBatchNumber(  
Currency As int,  
Batch number As string);**

Input:

Currency - currency to be used in the batch. Enum: FLX\_CURR\_CODES.  
Batch number - the batch number for the current batch.

Return: as Boolean

true - arguments were accepted and administration function was started.  
false - arguments were not accepted and administration function not started.

**Fire event: AdministrationFinished**

#### **6.3.2.26 AdministrationExcludeDatastoreWithSTAN**

Starts the administration function on the terminal that excludes a transaction with the given STAN.

Method: AdministrationExcludeDatastoreWithSTAN -

**Prototype: PointTerminalUsercontrol.AdministrationExcludeDatastoreWithSTAN(  
PSAM slot As uint,  
File ID As uint,  
File type As uint,  
STAN As uint);**

Input:

PSAM slot - the slot number the PSAM is placed in.

File ID - ID of the file to look in.

File type - ID of the file that the advice should be moved to.

STAN - the STAN of the transaction that should be excluded.

Return: as Boolean

true - arguments were accepted and administration function was started.

false - arguments were not accepted and administration function not started.

**Fire event: AdministrationFinished**

#### **6.3.2.27 AdministrationSetTeleDoneSettings**

Set the TeleDone settings on the terminal to send a TCP message to the given IP and port and to receive the result of the program download.

Method: AdministrationSetTeleDoneSettings -

**Prototype: PointTerminalUsercontrol.AdministrationSetTeleDoneSettings(  
IP As String,  
Port As String);**

Input:

IP - the IP of the TCP socket listening for the message. An empty string clears the data.

Port - Port number of the TCP socket. An empty string clears the data.

Return: as Boolean

true - arguments were accepted and administration function was started.

false - arguments were not accepted and administration function not started.

**Fire event: AdministrationFinished**

#### **6.3.2.28 PTInitCallback**

Initializes callbacks in PointTerminal.dll

If the callback isn't initialized, it will be handled in PointTerminal.dll and not sent as a event.

See Events

Method: PTInitCallback

**Prototype:** PointTerminalUsercontrol.PTInitCallback(  
    **callback as FLX\_CALLBACK**);

Input: FLX\_CALLBACK - The callback you want to initialize.

Return: as void

**Fire event:** -

**Prototype:** PointTerminalUsercontrol.PTInitCallback(  
    **List of callbacks as List<FLX\_CALLBACK>**);

Input: List<FLX\_CALLBACK> - A list of the callbacks you want to initialize.

Return: as void

**Fire event:** -

#### 6.3.2.29 SetAcceptCardResult

If the integration has signed up for the AcceptCard event, then this method will have to be called after the AcceptCard event was received.

This method is replaced by SetCardDataResult if implemented.

Method: SetAcceptCardResult -

**Prototype:** PointTerminalUsercontrol.SetAcceptCardResult(  
    **Accept Card As boolean**);

Input: Accept Card - Accepts (true) or declines (false) the card.

Return: as void

**Fire event:** -

#### 6.3.2.30 SetCardDataResult

If the integration has signed up for the SetCardData event, then this method will have to be called after the SetCardData event was received.

This method replaces SetAcceptCardResult if implemented.

Method: SetCardDataResult -

**Prototype:** PointTerminalUsercontrol.SetCardDataResult(  
    **Return Code As LX\_CARDNUMBER\_CMD**);

Input: Return code - Option to accept (true) or decline (false) the card.

Return: as void

**Fire event:** -

#### 6.3.2.31 PrintReceiptResult

If the integration has signed up for the PrintReceipt event, then this method will have to be called after the PrintReceipt event was received.

Method: PrintReceiptResult -

**Prototype: PointTerminalUsercontrol.PrintReceiptResult(  
Return Code As FLX\_RECEIPT\_CALLBACK\_RETURN\_VALUE);**

Input: Return code - Determine whether to print receipt (0x0X) or not (0x00).

Return: as void

**Fire event: -**

#### **6.3.2.32 GetTokenResult**

If the integration has signed up for the GetToken event, then this method will have to be called after the GetToken event was received.

Method: GetTokenResult -

**Prototype: PointTerminalUsercontrol.GetTokenResult(  
Return Code As FLX\_RECEIPT\_CALLBACK\_RETURN\_VALUE);**

Input: Return code - Option to accept or decline the authorization.

Return: as void

**Fire event: -**

#### **6.3.2.33 PutTokenResult**

If the integration has signed up for the PutToken event, then this method will have to be called after the PutToken event was received.

Method: PutTokenResult -

**Prototype: PointTerminalUsercontrol.PutTokenResult(  
Arguments As PutTokenEventArgs);**

Input: Arguments - Used in Capture or Reversal.

Return: as void

**Fire event: -**

#### **6.3.2.34 PcbStopListResult**

If the integration has signed up for the PcbStopList event, then this method will have to be called after the PcbStopList event was received.

Method: PcbStopListResult -

**Prototype: PointTerminalUsercontrol.PcbStopListResult(  
Arguments As PcbStopListEventArgs);**

Input: Arguments - Returns 1 if code is ready.

Return: as void

**Fire event: -**

#### **6.3.2.35 SetExtendedAmountResult**

If the integration has signed up for the SetExtendedAmount event, then this method will have to be called after the SetExtendedAmount event was received.



Method: SetExtendedAmountResult -

**Prototype: PointTerminalUsercontrol.SetExtendedAmountResult(Arguments As ExtendedAmountEventArgs);**

Input: Arguments - Used to send information from ECR to terminal.

Return: as void

**Fire event: -**

#### **6.3.2.36 EIEDataReceivedResult**

If the integration has signed up for the EIEDataReceived event, then this method will have to be called after the EIEDataReceived event was received.

Method: EIEDataReceivedResult -

**Prototype: PointTerminalUsercontrol.EIEDataReceivedResult(Arguments As EIEDataReceivedEventArgs);**

Input: Arguments - Determine whether data is sent/received.

Return: as void

**Fire event: -**

#### **6.3.2.37 EIEData2HostResult**

If the integration has signed up for the EIEData2Host event, then this method will have to be called after the EIEData2Host event was received.

Method: EIEData2HostResult -

**Prototype: PointTerminalUsercontrol.EIEData2HostResult(Arguments As EIEData2HostEventArgs);**

Input: Arguments - Determine whether block contains valid data.

Return: as void

**Fire event: -**

#### **6.3.2.38 PutScanBarCodeResult**

If the integration has signed up for the PutScanBarCode event, then this method will have to be called after the PutScanBarCode event was received.

Method: PutScanBarCodeResult -

**Prototype: PointTerminalUsercontrol.PutScanBarCodeResult(Arguments As PutScanBarCodeEventArgs);**

Input: Arguments - Option to initiate bar code transaction.

Return: as void

**Fire event: -**

#### 6.3.2.39 PcbBreakIPResult

If the integration has signed up for the PcbBreakIP event, then this method will have to be called after the PcbBreakIP event was received.

Method: PcbBreakIPResult -

**Prototype: PointTerminalUsercontrol.PcbBreakIPResult(  
Return Code As Int);**

Input: Return Code - Option to continue IP forwarding.

Return: as void

**Fire event: -**

#### 6.3.2.40 CTLSTransactions

Enable or disable contactless transactions.

Contactless transactions is disabled by default.

Always returns TRUE since it is handled by terminal.

Method: CTLSTransactions -

**Prototype: PointTerminalUsercontrol.CTLSTransactions(  
Enable Contactless As boolean);**

Input: Enable Contactless - Enable contactless transactions.

Return: as void

**Fire event: -**

#### 6.3.2.41 GetFlexDriverVersion

Get the version of the FlexDriver.

Method: GetFlexDriverVersion -

**Prototype: PointTerminalUsercontrol.GetFlexDriverVersion();**

Input: -

Return: as string

Example: 3.6.01

**Fire event: -**

#### 6.3.2.42 GetPointTerminalVersion

Get the version of PointTerminalDLL.

Method: GetPointTerminalVersion -

**Prototype: PointTerminalUsercontrol.GetPointTerminalVersion();**

Input: -

Return: as string

Example: 1.0.0.0

**Fire event: -**

#### **6.3.2.43 GetTokenInfo**

Get all info about a token.

Method: GetTokenInfo -

**Prototype: PointTerminalUsercontrol.GetTokenInfo( Token path As string);**

Input: Token path - full path for the token

Return: as string

Example: -

**Fire event: -**

#### **6.3.2.44 GetTransactionResult**

Get info about a transaction.

Method: GetTransactionResult -

**Prototype: PointTerminalUsercontrol.GetTransactionResult( STAN As String);**

Input: STAN - STAN for the requested transaction info

Return: as Tuple<String,String,String,String,String> -

Item1: Transaction amount

Item2: Currency code - FLX\_CURR\_CODES

Item3: -

Item4: -

Item5: STAN

Example (approved transaction):

Item1: 66

Item2: 208

Item3: 2

Item4: 1406251622

Item5: 003916

Example (not approved transaction):

Item1: 0

Item2: 0

Item3: 0

Item4: 0000000000

Item5: 000000

**Fire event: -**

#### **6.3.2.45 SetInterfaceReceipt**

Set the interface receipt on the next printed receipt.

This gives the option to set a receipt that is printed before the creditcard receipt from the terminal.

Method: SetInterfaceReceipt -

**Prototype:** `PointTerminalUsercontrol.SetInterfaceReceipt( Receipt As string);`

Input: Receipt - the receipt text

Return: as void

**Fire event:** -

#### 6.3.2.46 SetPrintHeader

Set the header on the next printed receipt.

Method: SetPrintHeader -

**Prototype:** `PointTerminalUsercontrol.SetPrintHeader( Receipt As string);`

Input: Receipt - the receipt header

Return: as void

**Fire event:** -

#### 6.3.2.47 SetPrintFooter

Set the footer on the next printed receipt.

Method: SetPrintFooter -

**Prototype:** `PointTerminalUsercontrol.SetPrintFooter( Receipt As string);`

Input: Receipt - the receipt footer

Return: as void

**Fire event:** -

#### 6.3.2.48 ForcePrint

Force a print of a receipt.

Method: ForcePrint -

**Prototype:** `PointTerminalUsercontrol.ForcePrint();`

Input: -

Return: as void

**Fire event:** -

#### 6.3.2.49 GetCardSwipe

Start listening for cardswipe.

Has to be restarted when the event is received.

Method: GetCardSwipe -

**Prototype:** `PointTerminalUsercontrol.GetCardSwipe();`

Input: -

Return: as void

**Fire event: CardSwiped**

#### 6.3.2.50 ShowDisplayWindow

Option to force the display window to be hidden when it would normally be shown.

Method: ShowDisplayWindow -

**Prototype: PointTerminalUsercontrol.ShowDisplayWindow(showWindow As Boolean);**

Input: showWindow - Boolean deciding if the display window should be forcibly hidden.

Return: as void

**Fire event: -**

#### 6.3.2.51 GetLicense

Get the license flag from the terminal.

Method: GetLicense -

**Prototype: PointTerminalUsercontrol.GetLicense();**

Input: -

Return: as Integer

FlexDriver / LPP: 1

PointWare OCX / PointTerminal.dll: 2

PointWare Expedient: 3

**Fire event: -**

### 6.3.3 Events

#### 6.3.3.1 Terminal connection

Events regarding the terminal connection.

The events are fired once for each method call made, if you have signed up to receive them.

**6.3.3.1.1 LoadFinished** This event is fired when load of the terminal (LoadTerminal) has finished.

The event contains the field:

Name	Data type	Description
Result	uint	Result of the called method. Also describes the terminal state. Enum: CONNECT_RETURN_VALUES

**6.3.3.1.2 UnloadFinished** This event is fired when unload of the terminal (UnloadTerminal) has finished.

The event contains the same fields as LoadFinished.

**6.3.3.1.3 ConnectFinished** This event is fired when connect to the terminal (ConnectTerminal) has finished.

The event contains the same fields as LoadFinished.

**6.3.3.1.4 OpenFinished** This event is fired when opening of the terminal (OpenTerminal) has finished.

The event contains the same fields as LoadFinished.

**6.3.3.1.5 CloseFinished** This event is fired when closing of the terminal (CloseTerminal) has finished.

The event contains the same fields as LoadFinished.

**6.3.3.1.6 DisconnectFinished** This event is fired when disconnect from the terminal (DisconnectTerminal) has finished.

The event contains the same fields as LoadFinished.

**6.3.3.1.7 TerminalDataFinished** This event is fired when the terminal is opened correctly, either by calling LoadTerminal or OpenTerminal.

The event contains the fields:

Name	Data type	Description
TERMINALID	string	ID of the terminal
RECEIPTTYPE	string	Internal representation of how the receipt should be delivered
GRATUITYMODEL	string	Gratuity on transactions. Represents a boolean.
DCC	string	DCC on transactions. Represents a boolean.
TOKEN	string	Token transactions enabled. Represents a boolean.
PREPAID	string	Prepaid transactions enabled. Represents a boolean.
KEYENTRY	string	Key entry on transactions enabled. Represents a boolean.
OFFLINE	string	Offline transactions enabled. Represents a boolean.
IP_ROUTING	string	IP routing enabled. Represents a boolean.
MERCHANTNUMBER	string	Merchant number
MERCHANTNAME	string	Merchant name
MERCHANTCITY	string	Merchant city
MERCHANTADDRESS	string	Merchant address
MERCHANTZIP	string	Merchant zip code
MERCHANTPHONE	string	Merchant phone number
MERCHANTBRN	string	-
FLXTRACELEVEL	string	Flex trace level. '16' is no trace.

LANGUAGE	string	-
VAT	string	Additional VAT information. Eg. 250 is 25% VAT.
USERINPUT	string	Value: 0, 1 or 255. Not used anymore.
COUNTRYCODE	string	Country code in PSAM. Enum: FLX_CURR_CODES
DEFAULTCURRENCY	string	Default currency on transactions. Enum: FLX_CURR_CODES
CONTACTLESSFLAGS	string	Not used anymore.
TERMINAL_TYPE	string	Type of the terminal. See <TERMINALTYPE> in Flexdriver documentation.
DCCMARKUP	string	DCC markup
TCS	string	TCS level. Can be 0, 1 or 2.
SOFTWARE_VERSION	string	Terminal software version.
CDP	string	Card Data Protection level.
PSAM_CDP	string	PSAM Card Data Protection level.
HARWARENAME	string	Name of the terminal.
EIE	string	Extended Issuer Envelope enabled. Represents boolean.
PSAM_EIE	string	PSAM Extended Issuer Envelope enabled. Represents boolean.
ISSUER_ENVELOPE_NON_EMV_SIZE	string	EMV data length
ISSUER_ENVELOPE_EMV_SIZE	string	EMV data length
TOTAL_ISSUER_ENVELOPE_NON_EMV_SIZE	string	EMV data length
TOTAL_ISSUER_ENVELOPE_EMV_SIZE	string	EMV data length
ERECEIPTSCHEMES	string	E-receipt scheme. See OTRS.
PSAMVERSION	string	PSAM version

### 6.3.3.2 Transactions

**6.3.3.2.1 TransactionFinished** This event is fired when a transaction is finished.  
The event contains the fields:

Name	Data type	Description
Result	uint	Transaction result. Enum: RETURN_VALUES
Error	uint	Transaction error code. '0' is approved transaction.
Amount	uint	Transaction amount at start of transaction.
AmountFee	uint	Fee amount at start of transaction.

AmountGratuity	uint	Gratuity amount at start of transaction.
AmountVAT	uint	VAT amount at start of transaction.
AmountCashback	uint	Cashback amount at start of transaction.
Currency	uint	Currency at start of transaction. Enum: FLX_CURR_CODES
STAN	string	Initial STAN set by terminal.
PAN	string	Card number
CRC	string	Card circuit ID
DTHR	string	Transaction time (5 byte)
TransType	uint	Transaction type. Enum: FLX_TRANSTYPE
MI	uint	Transaction merchant initiative. Enum: FLX_MERCHANT_INITIATIVE
AuthCode	string	Authorization code
Token	string	Full path to token for transaction
RefNr	uint	Reference number at start of transaction
BarCode	string	Bar code used for a bar code transaction
BinaryTransactionTime	string	Transaction time
BinaryAmountTotal	string	Final transaction total
BinaryAmountExtra	string	Final extra amount for the transaction
BinaryAmountFee	string	Final fee amount on the transaction
BinaryAmountGratuity	string	Final gratuity amount on transaction
BinaryCurrencyCode	string	Final currency code on transaction. Enum: FLX_CURR_CODES
BinaryStan	string	Final STAN of the transaction
BinaryPSAMCreator	string	PSAM creator
BinaryPSAMID	string	PSAM ID
BinaryStatus	string	Final transaction status
BinaryASW1ASW2	string	Transaction error code
BinaryCVMStatus	string	Final card verification status
BinaryAuthorizationCode	string	Authorization code for transaction
BinaryCardName	string	Card name
BinaryTerminalID	string	ID of terminal used for transaction
BinaryAgreementNumber	string	Merchant agreement number
BinaryName	string	Name on merchant agreement
BinaryCity	string	Merchant city
BinaryAddress	string	Merchant address
BinaryZip	string	Merchant zip code
BinaryPhone	string	Merchant phone number
BinaryCVR	string	Merchant CVR number
BinaryRefNumber	string	Final reference number in transaction
BinaryDCCCurrency	string	DCC currency used in transaction



BinaryDCCRate	string	DCC rate used in transaction
BinaryCRC	string	Final card group number (card circuit ID)
BinaryBatchNumber	string	Batch number of transaction
BinaryCancellationActive	string	Cancellation active during transaction
BinaryVAT	string	Final VAT on the transaction. Not used in Denmark.
BinaryEReceipt	string	E-receipt ID
BinaryBalance	string	-
BinaryBack	string	Final cashback amount on transaction
BinaryDCCTotal	string	Final total DCC amount on transaction
BinaryDCCFee	string	Final DCC fee amount on transaction
BinaryDCCGratuity	string	Final DCC gratuity amount on transaction
BinaryCardDataSource	string	-
BinaryAID	string	EMV application identification
BinaryATC	string	EMV value. See EMV specification.
BinaryAED	string	EMV value. See EMV specification.
BinaryARC	string	EMV value. See EMV specification.
BinaryExpiryData	string	-
JsonResult	string	VAS result.
TCC	string	EMV value. See EMV specification.

#### 6.3.3.2.2 **AcceptCard** NB! Only called if SetCardData is not implemented.

This event is fired first time the cardnumber is known during a transaction.

It gives the option to accept or decline the card by using the SetAcceptCardResult method. This method has to be called if the integration has signed up for the event.

The event contains the fields:

Name	Data type	Description
CardNumber	string	Card number of card swiped.
CardGroup	string	Card group of card used. Defined by a table in the terminal created by Verifone.
CardType	string	Type of card used.

#### 6.3.3.2.3 **SetCardData** NB! Replaces AcceptCard if implemented.

This event is fired first time the cardnumber is known during a transaction.

It gives the option to accept or decline the card by using the SetCardDataResult method. This is a mandatory function to implement.

The event contains the fields:

Name	Data type	Description
CardNumber	string	Card number of card swiped.
CardType	FLX_CARDTYPE	Type card used.
RC	FLX_CARDNUMBER_CMD	Return code - 0x00 is CardNumber_NotOK, others (0x01 etc) are OK.

**6.3.3.2.4 PrintReceipt** This event is fired first time the terminal is asked to print the receipt. It gives the option to print receipt by using the PrintReceiptResult method. Receipt can be divided into smaller subsections in case of large text.

In case of return 0x00 (error) no receipt will be printed.

The event contains the fields:

Name	Data type	Description
ReceiptStatus	int	Whether receipt is printed or not; Receipt_OK = 0x01 for received and printed receipt.
Data	string	Text on receipt.
RC	FLX_RECEIPT_CALLBACK_RETURN_VALUE	Return code.

**6.3.3.2.5 EarlyStanPan** This event is fired first time transaction STAN and PCI padded PAN have to be sent.

It is used to give transaction STAN and PAN.

The event contains the field:

Name	Data type	Description
Data	string	Card data - STAN and PAN of card used.

**6.3.3.2.6 GetToken** This event is fired first time an original authorization is performed. It gives the option to accept or decline the authorization by using the GetTokenResult method. Receipt\_OK is returned if token received and saved OK.

The event contains the fields:

Name	Data type	Description
TokenLength	int	Length of Token.
TokenData	byte[ ]	Data from Token.
TokenID	string	ID Token.
RC	FLX_RECEIPT_CALLBACK_RETURN_VALUE	Return code.

**6.3.3.2.7 PutToken** This event is fired first time capture or reversal are performed.

Token is used in capture and reversal by applying the PutTokenResult method.

The event contains the fields:

Name	Data type	Description
TokenLength	int	Length of Token.
TokenData	byte[ ]	Data from Token.

**6.3.3.2.8 PcbStopList** This event is fired repeatedly until 1 is returned indication that the code is ready, based on the PcbStopListResult method.

It must be activated if offline transactions are performed.

The event contains the fields:

Name	Data type	Description
Code	string	Authorization code - 6 characters + 2 character stop list status.
RC	int	Return code.

**6.3.3.2.9 GetExtendedAmount** This event is fired first time a user entered amount is present. Used to send the amount to the ECR.

The event contains the fields:

Name	Data type	Description
Value	uint	The entered amount.
Type	FLX_AMOUNT_TYPE	Type of entered amount (e.g gratuity, fee etc.).
CurrencyCode	FLX_CURR_CODES	E.g. 208 for DKK.

**6.3.3.2.10 SetExtendedAmount** This event is fired first time a user entered amount is present. Used to send the amount from the ECR to the terminal by applying the SetExtendedAmountResult method.

The event contains the same fields as GetExtendedAmount.

**6.3.3.2.11 EIEDataReceived** This event is fired when data send by issuer host is delivered to the terminal as part of the transaction flow. Determine whether data is sent/received using the EIEDataReceivedResult method.

The event contains the fields:

Name	Data type	Description
EIELength	int	Length of data; 1 for resp_mode, +4 for length_ie and length_eie.
EIE_DATA_BLOCK	byte[ ]	Struct containing the data.
Result	int	Return code.

**6.3.3.2.12 EIEData2Host** This event is fired as EIEDataReceived but gives a return value of 1 if the block contains valid data, 0 otherwise. To get result the EIEData2HostResult method is applied.

The event contains the fields:

Name	Data type	Description
EIELength	int	Length of data; 1 for resp_mode, +4 for length_ie and length_eie.
EIE_DATA_BLOCK	byte[ ]	Struct containing the data.

**6.3.3.2.13 PutScanBarCode** This event is fired first time the bar code gets scanned. It gives the option whether to initiate a bar code transaction based on the PutScanBarCodeResult method.

The event contains the fields:

Name	Data type	Description
Length	int	Length of bar code.
Barcode	string	Bar code used for a bar code transaction.
RC	int	Return code.

### 6.3.3.3 Administration

**6.3.3.3.1 AdministrationFinished** This event is fired when an administration function has finished.

The event contains the fields:

Name	Data type	Description
Result	uint	Result of the called administration function. Enum: RETURN_VALUES.
Error	uint	Error code of the administration function. 0 = success.

### 6.3.3.4 Other

**6.3.3.4.1 CardSwiped** This event is fired when a card is swiped in the terminal. The event does not contain any additional data.

**6.3.3.4.2 GetReceipt** This event is fired when a receipt is sent from the terminal.

The event contains the field:

Name	Data type	Description
Receipt	String	Receipt from terminal with the ISO-8859-15 encoding

**6.3.3.4.3 AdviceFlag** This event is fired if the host indicates that a balancing/EndofDay routine must be called.

This function is mandatory to implement.

The event does not contain any additional data.

**6.3.3.4.4 PcbBreakIP** This event is fired repeatedly to check if IP forwarding can be continued by using the PcbBreakIPResult method.

The event contains the fields:

Name	Data type	Description
Sync	int	1 if terminal synchronize frame detected, 0 if not.
RC	int	Return code.

# 7 | PointWare Expedient File Interface

## 7.1 Interface

The file interface is a simple two component interface.

The interface works the way that a command is written in the command file. PWE automatically gets the event that the command file is rewritten and reads the command.

PWE then processes the command and writes the answer in the answer file.

The different commands is listed in the Commands section.

The structure for the answer written in the answer file depends on the command given.

The overall the structure is:

Integer;Integer, hex or text;Integer;Integer;XML  
A;B;C;D;E

Enums from Enums.cs (see PointTerminal section) will be the reference for enums in this chapter.

For rules regarding the use of functionality described in this chapter please consult the OTRS delivered by Nets.

### **For connection commands the answer structure is:**

- A:
  - 0: Only possible value.
- B:
  - Command Done: Only possible value.
- C:
  - 0: Only possible value.
- D:
  - Integer: Can be parsed to CONNECT\_RETURN\_VALUES.
- E:
  - Terminal data in XML. Only when LoadTerminal and Ready finishes with CONNECT\_OK.

### **For transactions the structure is:**

- A:
  - 0: Transaction approved
  - 1: Transaction not approved
- B:
  - ASW1ASW2: 0 for an approved transaction. Other values represents a failed transaction.

- C:  
0: Always 0 for transactions.
- D:  
0: Always 0 for transactions.
- E:  
XML with all transaction info. For failed transactions most fields are empty.

**For administration the structure is:**

- A:  
0: Only possible value.
- B:  
Command Done: Only possible value.
- C:  
Integer: Can be parsed to RETURN\_VALUES.
- D:  
Integer: 0 represents success. Other values represent failure.
- E:  
Terminal data in XML when LoadTerminal or Ready finishes with D = (CONNECT\_OK).

**For the card swiped event the answer is:**

0;Card Swiped;0;0;

## 7.2 Commands

Commands accepted by PointWare Expedient

### 7.2.1 Connection

#### 7.2.1.1 LoadTerminal

Terminal goes to "Terminalen er klar" if successful. Command:

```
-loadterminal
```

Example results:

Successful load of terminal:

```
0;Command Done;0;0
<TERMINALID>00990977</TERMINALID>
<RECEIPTTYPE>189</RECEIPTTYPE>
<GRATUITYMODEL>0</GRATUITYMODEL>
<DCC>0</DCC>
<TOKEN>1</TOKEN>
<PREPAID>0</PREPAID>
<KEYENTRY>5</KEYENTRY>
```

<OFFLINE>1</OFFLINE>  
 <IP\_ROUTING>0</IP\_ROUTING>  
 <MERCHANTNUMBER>0002122227</MERCHANTNUMBER>  
 <MERCHANTNAME>Hotel DCC Test</MERCHANTNAME>  
 <MERCHANTCITY>Ballerup</MERCHANTCITY>  
 <MERCHANTADDRESS>Lautrupbjerg 10</MERCHANTADDRESS>  
 <MERCHANTZIP>2750</MERCHANTZIP>  
 <MERCHANTPHONE>46352966</MERCHANTPHONE>  
 <MERCHANTBRN></MERCHANTBRN>  
 <FLXTRACELEVEL>16</FLXTRACELEVEL>  
 <LANGUAGE>da</LANGUAGE>  
 <VAT>100</VAT>  
 <USERINPUT>255</USERINPUT>  
 <COUNTRYCODE>208</COUNTRYCODE>  
 <DEFAULTCURRENCY>208</DEFAULTCURRENCY>  
 <CONTACTLESSFLAGS>0</CONTACTLESSFLAGS>  
 <TERMINAL\_TYPE>34</TERMINAL\_TYPE>  
 <DCCMARKUP>3.0000</DCCMARKUP>  
 <TCS>1</TCS>  
 <SOFTWARE\_VERSION>3.5.04</SOFTWARE\_VERSION>  
 <CDP>12</CDP>  
 <PSAM\_CDP>12</PSAM\_CDP>  
 <HARWARENAME>VX820</HARWARENAME>  
 <EIE>0</EIE>  
 <PSAM\_EIE>0</PSAM\_EIE>  
 <ISSUER\_ENVELOPE\_NON\_EMV\_SIZE>0</ISSUER\_ENVELOPE\_NON\_EMV\_SIZE>  
 <ISSUER\_ENVELOPE\_EMV\_SIZE>0</ISSUER\_ENVELOPE\_EMV\_SIZE>  
 <TOTAL\_ISSUER\_ENVELOPE\_NON\_EMV\_SIZE>0</TOTAL\_ISSUER\_ENVELOPE\_NON\_EMV\_SIZE>  
 <TOTAL\_ISSUER\_ENVELOPE\_EMV\_SIZE>0</TOTAL\_ISSUER\_ENVELOPE\_EMV\_SIZE>  
 <ERECEIPTSCHEMES>0</ERECEIPTSCHEMES>  
 <PSAMVERSION>8.00.07</PSAMVERSION>  
 <TERMINALID>00990977</TERMINALID>  
 <RECEIPTTYPE>189</RECEIPTTYPE>  
 <GRATUITYMODEL>0</GRATUITYMODEL>  
 <DCC>0</DCC>  
 <TOKEN>1</TOKEN>  
 <PREPAID>0</PREPAID>  
 <KEYENTRY>5</KEYENTRY>  
 <OFFLINE>1</OFFLINE>  
 <IP\_ROUTING>0</IP\_ROUTING>  
 <MERCHANTNUMBER>0002122227</MERCHANTNUMBER>  
 <MERCHANTNAME>Hotel DCC Test</MERCHANTNAME>  
 <MERCHANTCITY>Ballerup</MERCHANTCITY>  
 <MERCHANTADDRESS>Lautrupbjerg 10</MERCHANTADDRESS>



```

<MERCHANTZIP>2750</MERCHANTZIP>
<MERCHANTPHONE>46352966</MERCHANTPHONE>
<MERCHANTBRN></MERCHANTBRN>
<FLXTRACELEVEL>16</FLXTRACELEVEL>
<LANGUAGE>da</LANGUAGE>
<VAT>100</VAT>
<USERINPUT>255</USERINPUT>
<COUNTRYCODE>208</COUNTRYCODE>
<DEFAULTCURRENCY>208</DEFAULTCURRENCY>
<CONTACTLESSFLAGS>0</CONTACTLESSFLAGS>
<TERMINAL_TYPE>34</TERMINAL_TYPE>
<DCCMARKUP>3.0000</DCCMARKUP>
<TCS>1</TCS>
<SOFTWARE_VERSION>3.5.04</SOFTWARE_VERSION>
<CDP>12</CDP>
<PSAM_CDP>12</PSAM_CDP>
<HARWARENAME>VX820</HARWARENAME>
<EIE>0</EIE>
<PSAM_EIE>0</PSAM_EIE>
<ISSUER_ENVELOPE_NON_EMV_SIZE>0</ISSUER_ENVELOPE_NON_EMV_SIZE>
<ISSUER_ENVELOPE_EMV_SIZE>0</ISSUER_ENVELOPE_EMV_SIZE>
<TOTAL_ISSUER_ENVELOPE_NON_EMV_SIZE>0</TOTAL_ISSUER_ENVELOPE_NON_EMV_SIZE>
<TOTAL_ISSUER_ENVELOPE_EMV_SIZE>0</TOTAL_ISSUER_ENVELOPE_EMV_SIZE>
<ERECEIPTSCHEMES>0</ERECEIPTSCHEMES>
<PSAMVERSION>8.00.07</PSAMVERSION>

```

Unsuccesfull load of terminal:

```
0;Command Done;0;3
```

### 7.2.1.2 UnloadTerminal

Terminal goes to "Lukket" if succesfull. Command:

```
-unloadterminal
```

Example results:

Successful load of terminal:

```
0;Command Done;0;0
```

Unsuccesfull unload of terminal:

```
0;Command Done;0;3
```

### 7.2.1.3 Ready

Terminal goes to "Terminalen er klar" if successful. The "-welcome" command will have to be used first. Command:

-ready

Example results:

Successful open of the terminal:

```
0;Command Done;0;0
<TERMINALID>00990977</TERMINALID>
<RECEIPTTYPE>189</RECEIPTTYPE>
<GRATUITYMODEL>0</GRATUITYMODEL>
<DCC>0</DCC>
<TOKEN>1</TOKEN>
<PREPAID>0</PREPAID>
<KEYENTRY>5</KEYENTRY>
<OFFLINE>1</OFFLINE>
<IP_ROUTING>0</IP_ROUTING>
<MERCHANTNUMBER>0002122227</MERCHANTNUMBER>
<MERCHANTNAME>Hotel DCC Test</MERCHANTNAME>
<MERCHANTCITY>Ballerup</MERCHANTCITY>
<MERCHANTADDRESS>Lautrupbjerg 10</MERCHANTADDRESS>
<MERCHANTZIP>2750</MERCHANTZIP>
<MERCHANTPHONE>46352966</MERCHANTPHONE>
<MERCHANTBRN></MERCHANTBRN>
<FLXTRACELEVEL>16</FLXTRACELEVEL>
<LANGUAGE>da</LANGUAGE>
<VAT>100</VAT>
<USERINPUT>255</USERINPUT>
<COUNTRYCODE>208</COUNTRYCODE>
<DEFAULTCURRENCY>208</DEFAULTCURRENCY>
<CONTACTLESSFLAGS>0</CONTACTLESSFLAGS>
<TERMINAL_TYPE>34</TERMINAL_TYPE>
<DCCMARKUP>3.0000</DCCMARKUP>
<TCS>1</TCS>
<SOFTWARE_VERSION>3.5.04</SOFTWARE_VERSION>
<CDP>12</CDP>
<PSAM_CDP>12</PSAM_CDP>
<HARWARENAME>VX820</HARWARENAME>
<EIE>0</EIE>
<PSAM_EIE>0</PSAM_EIE>
<ISSUER_ENVELOPE_NON_EMV_SIZE>0</ISSUER_ENVELOPE_NON_EMV_SIZE>
```

```
<ISSUER_ENVELOPE_EMV_SIZE>0</ISSUER_ENVELOPE_EMV_SIZE>  
<TOTAL_ISSUER_ENVELOPE_NON_EMV_SIZE>0</TOTAL_ISSUER_ENVELOPE_NON_EMV_SIZE>  
<TOTAL_ISSUER_ENVELOPE_EMV_SIZE>0</TOTAL_ISSUER_ENVELOPE_EMV_SIZE>  
<ERECEIPTSCHMES>0</ERECEIPTSCHMES>  
<PSAMVERSION>8.00.07</PSAMVERSION>
```

Unsuccesfull open of the terminal:

```
0;Command Done;0;3
```

#### **7.2.1.4 Welcome**

Terminal goes to "Velkommen" if successful. Command:

```
-welcome
```

Example results: Succesfull connect of the terminal:

```
0;Command Done;0;0
```

Unsuccesfull connect of the terminal:

```
0;Command Done;0;3
```

#### **7.2.1.5 Close**

Terminal goes to "Lukket" if successful. Command:

```
-close
```

Example results:

Succesfull closing of the connection to the terminal:

```
0;Command Done;0;0
```

Unsuccesfull closing of the connection to the terminal:

```
0;Command Done;0;3
```

#### **7.2.1.6 Exit**

Terminal goes to "Lukket" if successful and PWE is closed. Command:

```
-exit
```

Example results:

Succesfull closing the connection to the terminal and exiting of PWE:

```
0;Command Done;0;0
```

Unsuccesfull closing of the connection to the terminal:

```
0;Command Done;0;7
```

## 7.2.2 Transactions

Transactions has a lot of different arguments that can be combined in numerous ways.

All arguments:

Argument	Values	Description
-type	Card ForcedPin ForcedSignature ForcedOfflineSignature Refund Offline OfflineRefund OriginalPinAuthorization OriginalSignatureAuthorization SupplementalAuthorization Capture Reversal Cancellation Scanbarcodepurchase Scanbarcoderefund Scanbarcodeoriginalauth PostPurchase PostRefund	Transaction type. Card is default.
-amount	Unsigned integer	Transaction amount
-back	Unsigned integer	Cash back amount
-refnr	Unsigned integer	Internal reference number
-currency	See FLX_CURR_CODES	Ex. "DKK" or "NOK"
-password	String	Password for transactions and administration functions
-authcode	See "pcbStopList" in Flex-Driver documentation	Ex. "123456,00"
-noauthcode	-	This is used if a business decision is done, that no code is required.
-token	Token ID	ID of the token for token transactions
-gratuity	Unsigned integer	Gratuity amount
-barcode	String	Card number for barcode transactions
-keyentry	-	If present the transaction will be performed as key-entered

Sample commands:

```
-amount 3423
```

-amount 3412 -back 9  
 -amount 33156 -refnr 125  
 -amount 1231 -currency USD  
 -type Card -amount 234  
 -type ForcedPin -amount 3661  
 -type ForcedSignature -amount 771  
 -type ForcedOfflineSignature -amount 436  
 -type Refund -currency DKK -amount 946 -password 2730  
 -type Offline -amount 2312 -authCode PBSNUM,00  
 -type OfflineRefund -amount 1123 -AuthCode PBSNUM,00  
 -type OriginalPinAuthorization -currency DKK -amount 1733  
 -type OriginalSignatureAuthorization -amount 123222  
 -type SupplementalAuthorization -amount 5677 -token 013401  
 -type Capture -amount 33466 -token 013399  
 -type Capture -amount 76823 -token 002846 -gratuity 100  
 -type Reversal -amount 666 -token 002845 -refnr 1235  
 -type Cancellation -amount 6577 -password 2730  
 -type scanbarcodepurchase -amount 445 -barcode 6075999999047802240  
 -type postpurchase -amount 2442 -token 013454  
 -type postrefund -amount 2441 -token 013454

Sample answer:

0;0;0;0

<TYPE>CARD</TYPE>  
 <MANUEL>Y</MANUEL>  
 <TID>2014-07-01 12:49</TID>  
 <PAN>541333AAAAAA0078</PAN>  
 <TOTAL>500</TOTAL>  
 <EXTRA>0</EXTRA>  
 <GEBYR>0</GEBYR>  
 <DRIKKEPENGE>0</DRIKKEPENGE>  
 <VALUTAKODE>208</VALUTAKODE>  
 <STAN>3967</STAN>  
 <PSAMCREATOR>2129592318</PSAMCREATOR>  
 <PSAMID>105087</PSAMID>  
 <STATUS>0000</STATUS>  
 <ASW1ASW2>0000</ASW1ASW2>  
 <CVMSTATUS>12</CVMSTATUS>  
 <KORTNAVN>MasterCard</KORTNAVN>  
 <REFNR>0</REFNR>  
 <CRC>3</CRC>  
 <BATCH>AUTO</BATCH>  
 <MOMS>0</MOMS>  
 <E-KVITTERING></E-KVITTERING>

<ATC>1</ATC>  
<AED>040101</AED>  
<AUTHCODE>006118</AUTHCODE>  
<BARCODE></BARCODE>  
<ADDRESS>Lautrupbjerg 10</ADDRESS>  
<AGREEMENTNUMBER>2122227</AGREEMENTNUMBER>  
<AID>A0000000041010</AID>  
<ARC>00</ARC>  
<CASHBACK>0</CASHBACK>  
<BALANCE>0</BALANCE>  
<CARDDATASOURCE>0</CARDDATASOURCE>  
<CITY>Ballerup</CITY>  
<CVR></CVR>  
<DCCCURRENCY>208</DCCCURRENCY>  
<DCCFEE>0</DCCFEE>  
<DCCGRATUITY>0</DCCGRATUITY>  
<DCCRATE>1</DCCRATE>  
<DCCTOTAL>500</DCCTOTAL>  
<EXPDATE>EEEE</EXPDATE>  
<NAME>Hotel DCC Test</NAME>  
<PHONE>46352966</PHONE>  
<ZIP>2750</ZIP>  
<TERMINALID>00990977</TERMINALID>  
<CURRENCY>DKK</CURRENCY>  
<DTHR>140701</DTHR>  
<MI>MI\_PIN</MI>  
<REFNR>0</REFNR>  
<STAN>003967</STAN>  
<TOKEN></TOKEN>  
<TRANSTYPE>FLX\_TRANS\_PURCHASE</TRANSTYPE>

### 7.2.3 Administration

Commands (one command pr. line)	Description
-endofday -admincommand 1	End of day reconciliation
-endofdaylog -admincommand 2	End of day with log
-terminalreport -admincommand 3	Terminal report
-currenttotals -admincommand 4	Transaction total report
-currenttransactions -admincommand 5	Transaction log report
-previoustransactions -admincommand 6	Old transaction log report
-lastreceipt -admincommand 7	Get last receipt
-clocksyncpbs -admincommand 9	Sync the clock against PBS/Nets
-clocksyncpoint -admincommand 10	Sync the clock against Verifone
-sendlog -admincommand 11	Send log to Verifone Server System
-cleardatastore -admincommand 12	Clear datastore
-downloadprog -admincommand 13	Download Program
-downloadparam -admincommand 14	Download Param
-downloadpan -admincommand 15	Download PAN
-downloadtlcmdb -admincommand 16	Download TLC
-restoretlcmdb -admincommand 17	Restore communication file to default
-contrastup -admincommand 18	Terminal contrast up
-contrastdown -admincommand 19	Terminal contrast down
-restartterminal -admincommand 20	Restart terminal
-ejectcard -admincommand 21	Force an eject of the card

-msc -admincommand 22	Retrieve card range table
-backlighton -admincommand 23	Terminal backlight on
-backlightoff -admincommand 24	Terminal backlight off
-networkreport -admincommand 25	Network report
-ratesreport -admincommand 26	DCC currency rates report
-gratuityreceipt -admincommand 27	
-excludedatastorewithstan 5 4 3 2 -admincommand 28 5 4 3 2	Remove/delete defect advice file
-advicereport -admincommand 29	Emptying of the Terminals transactions database against the host (Nets)
-file5statusreport -admincommand 30	File5 Status report
-reportpct -admincommand 32	Report PCT
-updatedccrates -admincommand 33	Download DCC Rates
-updatepsam -admincommand 34	Update PSAM
-updatefeetable -admincommand 35	Update Fee Table
-updatesalt -admincommand 36	Update salt value for eKvittering
-getadvicerecon {days back: 0-12} Examples: -getadvicerecon 0 -getadvicerecon 7 -getadvicerecon 12  -admincommand 37 {days back: 0-12} Examples: -admincommand 37 0 -admincommand 37 7 -admincommand 37 12	Reconcillation with parameter



-batchnumber {batchnumber: 1-12 digits} Examples: -batchnumber 1212  -setbatchnumbers {FLX_CURR_CODES} {batch- number: 1-12 digits/AUTO} Examples: -setbatchnumbers DKK 1 -setbatchnumbers ISK 1234567 -setbatchnumbers JPY 123456789123 -setbatchnumbers DKK AUTO  -admincommand 38 {FLX_CURR_CODES} {batch- number: 1-12 digits/AUTO} Examples: -admincommand 38 DKK 1 -admincommand 38 ISK 133 -admincommand 38 DKK AUTO	Set batch number
-tcsreport -admincommand 39	TCS report
-reporttpropscvs -admincommand 40	Get and save the Terminal Properties used in the flexDriver to control Extended Issuer Envelope (EIE).
-eventreport -admincommand 41	Print the PCI eventlog report
-sendeventlog -admincommand 42	Send the PCI eventlog to a syslog server, with IP-address that you provide.
-removeeventlog -admincommand 43	Delete the PCI eventlog
-getipsettings -admincommand 44	Get IP settings

-setipsettings STATIC {IP} {Subnet} {Gateway} {DNS1} {DNS2} {Domain} Examples: -setipsettings STATIC 10.0.5.123 255.255.0.0 10.0.10.1 10.0.10.204 10.1.10.10 domain -setipsettings DHCP  -admincommand 45 STATIC {IP} {Subnet} {Gateway} {DNS1} {DNS2} {Domain} Examples: -admincommand 45 STATIC 10.0.5.123 255.255.0.0 10.0.10.1 10.0.10.204 10.1.10.10 domain -admincommand 45 DHCP	Set IP settings
-downloadimages -admincommand 46	Download images to terminal
-checkcard -admincommand 47	Check if card in terminal after a transaction
-getteledone -admincommand 48	Get TeleDone report
-setteledone {IP,Port} Examples: -setteledone , -setteledone 192.168.1.13,2310  -admincommand 49 {IP,Port} Examples: -admincommand 49 , -admincommand 49 192.168.1.13,2310	Set TeleDone settings
-ctlsreport -admincommand 50	Contactless report from the terminal
-gettransactionresult {STAN} Examples: -gettransactionresult 000123  -admincommand 80 {STAN} Examples: -admincommand 80 000123	Get Transaction Result from terminal It's only possible to get the latest 8 transactions from the PSAM. Note: Both declined transactions and transactions not know to the PSAM, will result in "NOT KNOWN".

### 7.2.4 Print

PWE gives the option the add extra information on the receipts.

The printfile command has three arguments

- File name: For a txt file that contains text to be inserted before the credit card receipt.
- Header: Text to be inserted above credit card receipt and txt file if this arguments is provided.

- Footer: Text to be inserted at the credit card receipt.

Examples:

```
-printfile printfiletest.txt  
-printfile printfiletest.txt -footer "Dette er en footer 1"  
-printfile printfiletest.txt -header "Dette er en header 1"  
-printfile -header "Dette er en header 1" -footer "Dette er en footer 1"  
-printfile printfiletest.txt -header "Dette er en header 1" -footer "Dette er en footer 1"  
-printfile -header "Dette er en header 2"  
-printfile -footer "Dette er en footer 2"
```

### 7.2.5 Show or minimize PWE

The file interface gives the option to control when PWE should be open for manual use and when it should be minimized for integration use only.

The manual command opens PWE for manual use:

```
-manual
```

The automatic command minimizes PWE to tray mode:

```
-automatic
```

## 8 | PointContactless

### 8.1 Document History

Version	Date	Comments
1.0–1.1	22. November 2013	Added information about CancellationRequest and CancellationAdvice

### 8.2 Introduction

We plan to use eight message types in the communication between the terminal and the host, four in each direction. All messages are sent in XML format complying with the EPAS specifications.

The transaction flow is described in detail in section “Flow of Transactions (Sequence Diagrams)” on page 387.

### 8.3 Other Documentation

The official documentation for the ISO 20022 standard can be found here: [ISO 20022 Documentation](#).

A more detailed usage guide of all available commands can be found here: [EPAS Usage Guide](#).

### 8.4 Format for Sending

All messages are prefixed by four bytes specifying the length of the message. The length is sent in network order, i.e. most significant byte first.

### 8.5 Message Structure

Every message contains three building blocks, a header, a message block and a security trailer. The security trailer contains a message authentication code, computed on the message building block with a cryptographic key. It allows the authentication of the initiator and protects the content of the message building block against any unauthorized alteration. The security trailer is only described in the AcceptorAuthorisationRequest section as it is identical, except for the MAC value, for all section types.

The documentation is built up in tables with this form:

Message Item	XML Tag	Description
Element		Description or possible value.
AnotherElement		
SubElement		Possible value: Explanation.

## 8.6 Notes on the Security Trailer

As the platform does not support DUKPT (Derived Unique Key Per Transaction) at the moment, we are using a trimmed version of the security trailer that only includes the actual MAC and no info about the KEK (key encryption key). This means that the security trailer is not EPAS compliant until the platform supports DUKPT.

## 8.7 AcceptorAuthorisationRequest

The AcceptorAuthorisationRequest message is sent by the card acceptor to the acquirer or its agent when an online authorization is required for the card payment transaction.

Message Item	XML Tag	Description
AcceptorAuthorisationRequest	AccptrAuthstnReq	

Message Item	XML Tag	Description
Header	Hdr	
MessageFunction	MsgFctn	<b>AUTQ</b> : Request for authorisation without financial capture. ( <i>TransactionCapture</i> =FALSE) <b>FAUQ</b> : Request for authorisation with financial capture. ( <i>TransactionCapture</i> =TRUE)
ProtocolVersion	PrtcolVrsn	MM.mm (assigned by EPASOrg). Current version is 1.0.
ExchangeIdentification	XchgId	Used in combination with <i>CreationDateTime</i> to allow the Recipient to identify re-transmissions. It is a cyclic counter that increments by one with each new message, starting at 0.
CreationDateTime	CreDtTm	Time accuracy has to be at least tenth of a second. (ISO 8601 format)
InitiatingParty	InitgPty	
Identification	Id	Terminal id.
RecipientParty	RcptPty	
Identification	Id	Unique name for recipient. Must match an entry in TLCMDB on terminal.

Message Item	XML Tag	Description
AuthorisationRequest	AuthstnReq	
Environment	Envnt	
POI	POI	
Identification	Id	
Identification	Id	Terminal id.
Card	Card	
PlainCardData	PlainCardData	
PAN	PAN	n-digit PAN without spaces.
ExpiryDate	XpryDt	Format: YYYY-MM
Context	Cntxt	
PaymentContext	PmtCntxt	
CardDataEntryMode	CardDataNtryMd	<b>CTLS:</b> Contactless proximity reader. <b>MGST:</b> Magnetic stripe.
Transaction	Tx	
TransactionCapture	TxCaptr	TRUE/FALSE based on <i>MessageFunction</i> .
TransactionType	TxTp	<b>BALC:</b> Balance enquiry. <b>CACT:</b> Card activation. <b>CAFT:</b> Transfer of funds to and/or from a card account. <b>CAVR:</b> Card verification. <b>CRDP:</b> Card payment. <b>RFND:</b> Refund transaction. <b>VALC:</b> Card validity check.
MerchantCategoryCode	MrchntCtgyCd	Category code conform to ISO 18245, related to the type of services or goods the merchant provides for the transaction. List of MCC codes: <a href="http://www.irs.gov/irb/2004-31_IRB/ar17.html">www.irs.gov/irb/2004-31_IRB/ar17.html</a>
TransactionIdentification	TxId	
TransactionDateTime	TxDtTm	UTC date time with offset or local date time. (ISO 8601 format)
TransactionReference	TxRef	Identification of the transaction that has to be unique for a time period. Max 35 characters.
TransactionDetails	TxDtls	
Currency	Ccy	
TotalAmount	TtlAmt	Use a “.” (dot) as decimal point.

Message Item	XML Tag	Description
SecurityTrailer	Scty	

Message Item	XML Tag	Description
ContentType	CnttTp	<b>AUTH:</b> MAC (Message Authentication Code), with encryption key – (ASN.1 Object Identifier: id-ct-authData).
AuthenticatedData	AuthntcdData	Data protection by a message authentication code (MAC).
Recipient	Rcpt	Information related to the transport key.
KEK	KEK	Encryption key using previously distributed symmetric key.
KEKIdentification	KEKId	
KeyIdentification	KeyId	Maximum 140 characters.
KeyVersion	KeyVrsn	Activation date or version of the key to differentiate several keys with the same name ( <i>KeyIdentification</i> ) using the format YYYYMMDDhh where: YYYY is a 4-digits numeral representing the year, 0000 is prohibited MM is a 2-digits numeral representing the month (from 01 to 12) DD is a 2-digits numeral representing the day of the month (from 01 to 31) hh is a 2-digits numeral representing the hours (from 00 to 23)
DerivationIdentification	DerivtnId	Identification used for derivation of a unique key from a master key provided for the data protection. Between 5 and 16 bits.
KeyEncryptionAlgorithm	KeyNcrptnAlgo	Algorithm to encrypt the KEK.
Algorithm	Algo	<b>DKPT:</b> DUKPT (Derived Unique Key Per Transaction) algorithm, as specified in ANSI X9.24-2004, Annex A, and ISO/DIS 13492-2006 – (ASN.1 Object Identifier: id-dukpt-wrap).
EncryptedKey	NcrptdKey	Maximum 140 bits.
MACAlgorithm	MACAlgo	Algorithm to compute MAC.
Algorithm	Algo	<b>MCCS:</b> Retail-CBC-MAC with SHA-256 (Secure Hash standard) – (ASN.1 Object Identifier: id-retail-cbc-mac-sha-256).
EncapsulatedContent	NcpsltdCntt	Data to authenticate.
ContentType	CnttTp	<b>DATA:</b> Generic, non cryptographic or unqualified data content – (ASN.1 Object Identifier: id-data).

Message Item	XML Tag	Description
MAC	MAC	Encrypted data which authenticates the data.

### 8.7.1 Example

Below is shown an example AcceptorAuthorisationRequest with a full security trailer, which includes the “Recipient” section containing the info about the KEK (key encryption key).

#### Basic merchant info

Merchant type	Men’s, Women’s Clothing Store ( <b>5691</b> )
Terminal Id	990001

#### Payment card info

PAN	1234 1234 1234 1234
Expiration date	December 2014 ( <b>2014-12</b> )
Type	Contactless card ( <b>CTLS</b> )

#### Transaction info

Start time	January 24, 2012 @ 9 am ( <b>2012-01-24T09:00:00.00</b> )
Type	Card payment ( <b>CRDP</b> )
Currency	Euro ( <b>EUR</b> )
Amount	20.00

The resulting XML file is shown in Listing 8.1.

Type of data protection	AuthenticatedData/MAC ( <b>AUTH</b> )
KeyIdentification	SpecV1TestKey <sup>1</sup>
KeyVersion	2010060715 <sup>1</sup>
DerivationIdentification	398725A501 ( <b>OYclpQE=</b> ) <sup>1</sup>
Key Encryption Algorithm	DUKPT ( <b>DKPT</b> )
EncryptedKey	E290200017 ( <b>4pAgABc=</b> ) <sup>1</sup>
MAC Algorithm	RetailSHA256MAC ( <b>MCCS</b> )
Type of data	PlainData ( <b>DATA</b> )
MAC	15 20 4F 17 68 48 5B 13 ( <b>FSBPF2hIWxM=</b> ) <sup>2</sup>

#### Listing 8.1: AcceptorAuthorisationRequest Example

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

<sup>1</sup>Values taken directly from example “8.5.3.2 MAC Computation” in *EPAS Usage guide*, page 306 and 307.

<sup>2</sup>Key used to compute MAC is the derived key found on page 306 in *EPAS Usage guide*.

Key = 5E 64 F1 AB F2 5D 3B A1 7F 62 9E C2 B3 02 F8 EA



```

xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AcceptorAuthstnReq>
    <Hdr>
      <MsgFctn>FAUQ</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
      <XchgId>0</XchgId>
      <CreDtTm>2012-01-24T09:00:05.00+01:00</CreDtTm>
      <InitgPty>
        <Id>990001</Id>
      </InitgPty>
      <RcptPty>
        <Id>test_acceptor</Id>
      </RcptPty>
    </Hdr>
    <AuthstnReq>
      <Envt>
        <POI>
          <Id>
            <Id>990001</Id>
          </Id>
        </POI>
        <Card>
          <PlainCardData>
            <PAN>1234123412341234</PAN>
            <XpryDt>2014-12</XpryDt>
          </PlainCardData>
        </Card>
      </Envt>
      <Cntxt>
        <PmtCntxt>
          <CardDataNtryMd>CTLS</CardDataNtryMd>
        </PmtCntxt>
      </Cntxt>
      <Tx>
        <TxCaptr>true</TxCaptr>
        <TxTp>CRDP</TxTp>
        <MrchntCtgyCd>5691</MrchntCtgyCd>
        <TxId>
          <TxDtTm>2012-01-24T09:00:00.00+01:00</TxDtTm>
          <TxRef>000001</TxRef>
        </TxId>
        <TxDtls>
          <Ccy>EUR</Ccy>
          <TtlAmt>20.00</TtlAmt>
        </TxDtls>
      </Tx>
    </AuthstnReq>
    <SctyTrlr>
      <Cnttp>AUTH</Cnttp>
      <AuthntcdData>
        <Rcpt>
          <KEK>
            <KEKId>
              <KeyId>SpecV1TestKey</KeyId>
            </KEKId>
          </Rcpt>
        </AuthntcdData>
      </SctyTrlr>
    </AuthstnReq>
  </AcceptorAuthstnReq>

```

```

        <KeyVrsn>2010060715</KeyVrsn>
        <DerivtnId>0YclpQE=</DerivtnId>
      </KEKId>
      <KeyNcrptnAlgo>
        <Algo>DKPT</Algo>
      </KeyNcrptnAlgo>
      <NcrptdKey>4pAgABc=</NcrptdKey>
    </KEK>
  </Rcpt>
  <MACAlgo>
    <Algo>MCCS</Algo>
  </MACAlgo>
  <NcpsltdCntt>
    <CnttTp>DATA</CnttTp>
  </NcpsltdCntt>
  <MAC>FSBPF2hIWxM=</MAC>
</AuthntcdData>
</SctyTrlr>
</AccptrAuthstnReq>
</Document>

```

## 8.8 AcceptorAuthorisationResponse

The AcceptorAuthorisationResponse message is sent by the acquirer to inform the card acceptor of the outcome of the authorisation process.

The AcceptorAuthorisationResponse message is used to indicate one of the possible outcomes of an authorisation process:

- A successful authorisation
- A decline from the acquire for financial reasons
- A decline from the acquire for technical reasons (for instance, a timeout).

Message Item	XML Tag	Description
AcceptorAuthorisationResponse	AccptrAuthstnRspn	

Message Item	XML Tag	Description
Header	Hdr	
MessageFunction	MsgFctn	<b>AUTP:</b> Response for authorisation without financial capture. ( <i>TransactionCapture</i> =FALSE) <b>FAUP:</b> Response for authorisation with financial capture. ( <i>TransactionCapture</i> =TRUE)
ProtocolVersion	PrtcolVrsn	Copy from AcceptorAuthorisationRequest
ExchangeIdentification	XchgId	Copy from AcceptorAuthorisationRequest

Message Item	XML Tag	Description
CreationDateTime	CreDtTm	See <i>AcceptorAuthorisationRequest</i>
InitiatingParty	InitgPty	Copy from AcceptorAuthorisationRequest
Identification	Id	
RecipientParty	RcptPty	Copy from AcceptorAuthorisationRequest
Identification	Id	

Message Item	XML Tag	Description
AuthorisationResponse	AuthstnRspn	
Environment	Envt	
POI Identification	POIID	
Identification	Id	Can be different from the request
Transaction	Tx	
TransactionIdentification	TxId	Copy from AcceptorAuthorisationRequest
TransactionDateTime	TxDtTm	
TransactionReference	TxRef	
TransactionDetails	TxDtls	Copy from AcceptorAuthorisationRequest
Currency	Ccy	
TotalAmount	TtlAmt	
TransactionResponse	TxRspn	
AuthorisationResult	AuthstnRslt	
ResponseToAuthorisation	RspnToAuthstn	
Response	Rspn	<p><b>APPR:</b> (Approved) Authorisation is approved for the full amount requested, including capture if requested.</p> <p><b>DECL:</b> (Declined) Authorisation is declined or the requested capture is not performed.</p> <p><b>TECH:</b> (TechnicalError) Service cannot be provided for technical reason (e.g. timeout contacting the Issuer, security problem).</p>
Balance	Bal	Balance of the account, related to the payment. (Optional)

### Listing 8.2: AcceptorAuthorisationResponse Example

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AccptrAuthstnRspn>
    <Hdr>
      <MsgFctn>FAUP</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
```

```

    <XchgId>0</XchgId>
    <CreDtTm>2012-01-24T09:00:05.00+01:00</CreDtTm>
    <InitgPty>
      <Id>990001</Id>
    </InitgPty>
    <RcptPty>
      <Id>test_acceptor</Id>
    </RcptPty>
  </Hdr>
  <AuthstnRspn>
    <Envt>
      <POIID>
        <Id>990001</Id>
      </POIID>
    </Envt>
    <Tx>
      <TxId>
        <TxDtTm>2012-01-24T09:00:00.00+01:00</TxDtTm>
        <TxRef>000001</TxRef>
      </TxId>
      <TxDtls>
        <Ccy>EUR</Ccy>
        <TtlAmt>20.00</TtlAmt>
      </TxDtls>
    </Tx>
    <TxRspn>
      <AuthstnRslt>
        <RspnToAuthstn>
          <Rspn>APPR</Rspn>
        </RspnToAuthstn>
      </AuthstnRslt>
    </TxRspn>
  </AuthstnRspn>
</AccptrAuthstnRspn>
</Document>

```

## 8.9 AcceptorCompletionAdvice

The AcceptorCompletionAdvice message is sent by a card acceptor to notify an acquirer about the completion and final outcome of a card payment transaction.

The AcceptorCompletionAdvice message is used either to:

- Reverse a transaction which was not successfully completed (for example, cancellation of transaction by the cardholder), but where an authorization had been previously given. A reversal **must always** be approved!

Message Item	XML Tag	Description
AcceptorCompletionAdvice	AccptrCmpltnAdvce	

Message Item	XML Tag	Description
Header	Hdr	
MessageFunction	MsgFctn	<b>FRVA:</b> Advice for reversal with financial capture. ( <i>TransactionCapture</i> =TRUE, <i>Reversal</i> =TRUE) <b>RVRA:</b> Advice for reversal without financial capture. ( <i>TransactionCapture</i> =FALSE, <i>Reversal</i> =TRUE)
ProtocolVersion	PrtcolVrsn	See <i>AcceptorAuthorisationRequest</i>
ExchangeIdentification	XchgId	See <i>AcceptorAuthorisationRequest</i>
CreationDateTime	CreDtTm	See <i>AcceptorAuthorisationRequest</i>
InitiatingParty	InitgPty	See <i>AcceptorAuthorisationRequest</i>
Identification	Id	
RecipientParty	RcptPty	See <i>AcceptorAuthorisationRequest</i>
Identification	Id	

Message Item	XML Tag	Description
CompletionAdvice	CmpltnAdvc	
Environment	Envt	See <i>AcceptorAuthorisationRequest</i>
POI	POI	See <i>AcceptorAuthorisationRequest</i>
Identification	Id	
Identification	Id	
Card	Card	See <i>AcceptorAuthorisationRequest</i>
PlainCardData	PlainCardData	
PAN	PAN	
ExpiryDate	XpryDt	
Transaction	Tx	See <i>AcceptorAuthorisationRequest</i>
TransactionCapture	TxCaptr	
MerchantCategoryCode	MrchntCtgyCd	
TransactionIdentification	TxId	
TransactionDateTime	TxDtTm	
TransactionReference	TxRef	
TransactionSuccess	TxSucss	FALSE
Reversal	Rvsl	TRUE
TransactionDetails	TxDtls	See <i>AcceptorAuthorisationRequest</i>
Currency	Ccy	
TotalAmount	TtlAmt	

**Listing 8.3:** AcceptorCompletionAdvice Example

```

<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AccptrCmpltnAdv<
    <Hdr>
      <MsgFctn>FRVA</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
      <XchgId>0</XchgId>
      <CreDtTm>2012-01-24T09:00:05.00+01:00</CreDtTm>
      <InitgPty>
        <Id>990001</Id>
      </InitgPty>
      <RcptPty>
        <Id>test_acceptor</Id>
      </RcptPty>
    </Hdr>
    <CmpltnAdv<
      <Env<
        <POI>
          <Id>
            <Id>990001</Id>
          </Id>
        </POI>
        <Card>
          <PlainCardData>
            <PAN>1234123412341234</PAN>
            <XpryDt>2014-12</XpryDt>
          </PlainCardData>
        </Card>
      </Env>
      <Tx>
        <TxCaptr>TRUE</TxCaptr>
        <MrchntCtgyCd>5691</MrchntCtgyCd>
        <TxId>
          <TxDtTm>2012-01-24T09:00:00.00+01:00</TxDtTm>
          <TxRef>000001</TxRef>
        </TxId>
        <TxSucss>FALSE</TxSucss>
        <Rvsl>TRUE</Rvsl>
        <TxDtls>
          <Ccy>EUR</Ccy>
          <TtlAmt>20.00</TtlAmt>
        </TxDtls>
      </Tx>
    </CmpltnAdv>
  </AccptrCmpltnAdv>
</Document>

```

## 8.10 AcceptorCompletionAdviceResponse

The AcceptorCompletionAdviceResponse message is sent by the acquirer to acknowledge the proper receipt of an AcceptorCompletionAdvice.

Message Item	XML Tag	Description
AcceptorCompletionAdviceResponse	AccptrCmpltnAdvcRspn	

Message Item	XML Tag	Description
Header	Hdr	
MessageFunction	MsgFctn	<b>FRVR:</b> Advice response for reversal with financial capture. <b>RVRR:</b> Advice response for reversal without financial capture.
ProtocolVersion	PrtcolVrsn	Copy from AcceptorCompletionAdvice.
ExchangeIdentification	XchgId	Copy from AcceptorCompletionAdvice.
CreationDateTime	CreDtTm	See <i>AcceptorAuthorisationRequest</i>
InitiatingParty	InitgPty	Copy from AcceptorCompletionAdvice.
Identification	Id	
RecipientParty	RcptPty	Copy from AcceptorCompletionAdvice
Identification	Id	

Message Item	XML Tag	Description
CompletionAdviceResponse	CmpltnAdvcRspn	
Environment	Envt	
POIIdentification	POIId	
Identification	Id	Copy from AcceptorCompletionAdvice.
Transaction	Tx	
TransactionIdentification	TxId	Copy from AcceptorCompletionAdvice.
TransactionDateTime	TxDtTm	
TransactionReference	TxRef	
Response	Rspn	<b>APPR:</b> (Approved) Service has been successfully provided.

#### Listing 8.4: AcceptorCompletionAdviceResponse Example

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AccptrCmpltnAdvcRspn>
    <Hdr>
      <MsgFctn>FRVR</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
      <XchgId>0</XchgId>
      <CreDtTm>2012-01-24T09:00:05.00+01:00</CreDtTm>
      <InitgPty>
        <Id>990001</Id>
      </InitgPty>
      <RcptPty>
```

```

        <Id>test_acceptor</Id>
      </RcptPty>
    </Hdr>
    <CmpltnAdvcrspn>
      <Envt>
        <POIID>
          <Id>990001</Id>
        </POIID>
      </Envt>
      <Tx>
        <TxId>
          <TxDtTm>2012-01-24T09:00:00.00+01:00</TxDtTm>
          <TxRef>000001</TxRef>
        </TxId>
        <Rspn>APPR</Rspn>
      </Tx>
    </CmpltnAdvcrspn>
  </AcceptorCmpltnAdvcrspn>
</Document>

```

## 8.11 AcceptorCancellationRequest

The AcceptorCancellationRequest message is sent by the card acceptor to the acquirer or its agent to request a cancellation for a previously completed transaction.

Message Item	XML Tag	Description
AcceptorCancellationRequest	AcceptorCxlReq	

Message Item	XML Tag	Description
Header	Hdr	
MessageFunction	MsgFctn	<b>CCAQ:</b> CancellationRequest
ProtocolVersion	PrtcolVrsn	See <i>AcceptorAuthorisationRequest</i>
ExchangeIdentification	XchgId	See <i>AcceptorAuthorisationRequest</i>
CreationDateTime	CreDtTm	See <i>AcceptorAuthorisationRequest</i>
InitiatingParty	InitgPty	See <i>AcceptorAuthorisationRequest</i>
Identification	Id	
RecipientParty	RcptPty	See <i>AcceptorAuthorisationRequest</i>
Identification	Id	

Message Item	XML Tag	Description
CancellationRequest	CxlReq	
Environment	Envt	Copy of original transaction.
POI	POI	
Identification	Id	



Message Item	XML Tag	Description
Identification	Id	
Card	Card	Copy of original transaction.
PlainCardData	PlainCardData	
PAN	PAN	
ExpiryDate	XpryDt	
Context	Cntxt	Copy of original transaction.
PaymentContext	PmtCntxt	
CardDataEntryMode	CardDataNtryMd	
Transaction	Tx	
TransactionCapture	TxCaptr	Copy of original transaction.
MerchantCategoryCode	MrchntCtgyCd	Copy of original transaction.
TransactionIdentification	TxId	
TransactionDateTime	TxDtTm	
TransactionReference	TxRef	
OriginalTransaction	OrgnlTx	
TransactionIdentification	TxId	Copy of original transaction.
TransactionDateTime	TxDtTm	
TransactionReference	TxRef	
TransactionType	TxTp	TransactionType of the original transaction.
TransactionDetails	TxDtls	Copy of original transaction.
Currency	Ccy	
TotalAmount	TtlAmt	

### Listing 8.5: AcceptorCancellationRequest Example

```

<?xml version="1.0" encoding="utf-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AcceptorCxlReq>
    <Hdr>
      <MsgFctn>CCAQ</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
      <XchgId>0</XchgId>
      <CreDtTm>2012-01-24T09:05:00.00+01:00</CreDtTm>
      <InitgPty>
        <Id>990001</Id>
      </InitgPty>
      <RcptPty>
        <Id>test_acceptor</Id>
      </RcptPty>
    </Hdr>
    <CxlReq>
      <Envt>
        <POI>
          <Id>

```

```

        <Id>990001</Id>
      </Id>
    </POI>
    <Card>
      <PlainCardData>
        <PAN>1234123412341234</PAN>
        <XpryDt>2014-12</XpryDt>
      </PlainCardData>
    </Card>
  </Envt>
  <Cntxt>
    <PmtCntxt>
      <CardDataNtryMd>CTLS</CardDataNtryMd>
    </PmtCntxt>
  </Cntxt>
  <Tx>
    <TxCaptr>TRUE</TxCaptr>
    <MrchntCtgyCd>5691</MrchntCtgyCd>
    <TxId>
      <TxDtTm>2012-01-24T09:00:00.00+01:00</TxDtTm>
      <TxRef>000001</TxRef>
    </TxId>
    <OrgnlTx>
      <TxId>
        <TxDtTm>2012-05-31T11:49:50.00+02:00</TxDtTm>
        <TxRef>287</TxRef>
      </TxId>
      <TxTp>CRDP</TxTp>
    </OrgnlTx>
    <TxDtls>
      <Ccy>EUR</Ccy>
      <TtlAmt>20.00</TtlAmt>
    </TxDtls>
  </Tx>
</CxlReq>
</AccptrCxlReq>
</Document>

```

## 8.12 AcceptorCancellationResponse

The AcceptorCancellationResponse message is sent by the acquirer to acknowledge the proper receipt of an AcceptorCancellationRequest.

Message Item	XML Tag	Description
AcceptorCancellationResponse	AccptrCxlRspn	

Message Item	XML Tag	Description
Header	Hdr	
MessageFunction	MsgFctn	<b>CCAP:</b> CancellationResponse

Message Item	XML Tag	Description
ProtocolVersion	PrtcolVrsn	Copy from AcceptorCancellationRequest.
ExchangeIdentification	XchgId	Copy from AcceptorCancellationRequest.
CreationDateTime	CreDtTm	See <i>AcceptorAuthorisationResponse</i>
InitiatingParty	InitgPty	Copy from AcceptorCancellationRequest.
Identification	Id	
RecipientParty	RcptPty	Copy from AcceptorCancellationRequest
Identification	Id	

Message Item	XML Tag	Description
CancellationResponse	CxlRspn	
Environment	Envt	
POIIdentification	POIId	
Identification	Id	Copy from AcceptorCancellationRequest
Transaction	Tx	
TransactionIdentification	TxId	Copy from AcceptorCancellationRequest
TransactionDateTime	TxDtTm	
TransactionReference	TxRef	
TransactionDetails	TxDtls	
Currency	Ccy	Copy from AcceptorCancellationRequest
TotalAmount	TtlAmt	Amount to cancel which is always the TotalAmount of the transaction to cancel. Only the full amount of the transaction can be cancelled.
TransactionResponse	TxRspn	
AuthorisationResult	AuthstnRslt	
ResponseToAuthorisation	RspnToAuthstn	
Response	Rspn	See <i>AcceptorAuthorisationResponse</i>

### Listing 8.6: AcceptorCancellationResponse Example

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AcptrCxlRspn>
    <Hdr>
      <MsgFctn>CCAP</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
      <XchgId>0</XchgId>
      <CreDtTm>2012-01-24T09:00:05.00+01:00</CreDtTm>
      <InitgPty>
        <Id>990001</Id>
      </InitgPty>
      <RcptPty>
        <Id>test_acceptor</Id>
      </RcptPty>
    </Hdr>
  </AcptrCxlRspn>
</Document>
```

```

    </RcptPty>
  </Hdr>
  <CxlRspn>
    <Envt>
      <POIID>
        <Id>990001</Id>
      </POIID>
    </Envt>
    <Tx>
      <TxId>
        <TxDtTm>2012-01-24T09:00:00.00+01:00</TxDtTm>
        <TxRef>000001</TxRef>
      </TxId>
      <TxDtls>
        <Ccy>EUR</Ccy>
        <TtlAmt>20.00</TtlAmt>
      </TxDtls>
    </Tx>
    <TxRspn>
      <AuthstnRslt>
        <RspnToAuthstn>
          <Rspn>APPR</Rspn>
        </RspnToAuthstn>
      </AuthstnRslt>
    </TxRspn>
  </CxlRspn>
</AccptrCxlRspn>
</Document>

```

## 8.13 AcceptorCancellationAdvice

The AcceptorCancellationAdvice message is sent by a card acceptor to notify an acquirer about the completion and final outcome of a cancellation.

Message Item	XML Tag	Description
AcceptorCancellationAdvice	AccptrCxlAdvc	

Message Item	XML Tag	Description
Header	Hdr	
MessageFunction	MsgFctn	<b>CCAV:</b> CancellationAdvice
ProtocolVersion	PrtcolVrsn	See <i>AcceptorAuthorisationRequest</i>
ExchangeIdentification	XchgId	See <i>AcceptorAuthorisationRequest</i>
CreationDateTime	CreDtTm	See <i>AcceptorAuthorisationRequest</i>
InitiatingParty	InitgPty	See <i>AcceptorAuthorisationRequest</i>
Identification	Id	
RecipientParty	RcptPty	See <i>AcceptorAuthorisationRequest</i>
Identification	Id	

Message Item	XML Tag	Description
CancellationAdvice	CxlAdvC	
Environment	Envt	See <i>AcceptorAuthorisationRequest</i>
POI	POI	See <i>AcceptorAuthorisationRequest</i>
Identification	Id	
Identification	Id	
Transaction	Tx	See <i>AcceptorAuthorisationRequest</i>
MerchantCategoryCode	MrchntCtgyCd	
TransactionIdentification	TxId	
TransactionDateTime	TxDtTm	
TransactionReference	TxRef	
TransactionSuccess	TxSucss	FALSE
Reversal	Rvsl	TRUE
TransactionDetails	TxDtls	See <i>AcceptorAuthorisationRequest</i>
Currency	Ccy	
TotalAmount	TtlAmt	

### Listing 8.7: AcceptorCancellationAdvice Example

```

<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AccptrCxlAdvC>
    <Hdr>
      <MsgFctn>FRVA</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
      <XchgId>0</XchgId>
      <CreDtTm>2012-01-24T09:00:05.00+01:00</CreDtTm>
      <InitgPty>
        <Id>990001</Id>
      </InitgPty>
      <RcptPty>
        <Id>test_acceptor</Id>
      </RcptPty>
    </Hdr>
    <CxlAdvC>
      <Envt>
        <POI>
          <Id>
            <Id>990001</Id>
          </Id>
        </POI>
      </Envt>
      <Tx>
        <TxCaptr>TRUE</TxCaptr>
        <MrchntCtgyCd>5691</MrchntCtgyCd>
        <TxId>
          <TxDtTm>2012-01-24T09:00:00.00+01:00</TxDtTm>
          <TxRef>000001</TxRef>
        </TxId>
      </Tx>
    </CxlAdvC>
  </AccptrCxlAdvC>
</Document>

```

```

        </TxId>
        <TxSucss>FALSE</TxSucss>
        <Rvsl>TRUE</Rvsl>
        <TxDtls>
            <Ccy>EUR</Ccy>
            <TtlAmt>20.00</TtlAmt>
        </TxDtls>
    </Tx>
</CxlAdv>
</AcceptorCxlAdv>
</Document>

```

## 8.14 AcceptorCancellationAdviceResponse

The AcceptorCancellationAdviceResponse message is sent by the acquirer to acknowledge the proper receipt of an AcceptorCancellationAdvice.

Message Item	XML Tag	Description
AcceptorCancellationAdviceResponse	AccptrCxlAdvRspn	

Message Item	XML Tag	Description
Header	Hdr	
MessageFunction	MsgFctn	<b>CCAK:</b> CancellationAdviceResponse
ProtocolVersion	PrtcolVrsn	Copy from AcceptorCancellationAdvice.
ExchangeIdentification	XchgId	Copy from AcceptorCancellationAdvice.
CreationDateTime	CreDtTm	See AcceptorAuthorisationRequest
InitiatingParty	InitgPty	Copy from AcceptorCancellationAdvice.
Identification	Id	
RecipientParty	RcptPty	Copy from AcceptorCancellationAdvice
Identification	Id	

Message Item	XML Tag	Description
CancellationAdviceResponse	CxlAdvRspn	
Environment	Envt	
POIIdentification	POIId	
Identification	Id	Copy from AcceptorCancellationAdvice.
Transaction	Tx	
TransactionIdentification	TxId	Copy from AcceptorCancellationAdvice.
TransactionDateTime	TxDtTm	
TransactionReference	TxRef	

**Listing 8.8:** AcceptorCancellationAdviceResponse Example

```

<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AcceptorCxlAdvCspn>
    <Hdr>
      <MsgFctn>FRVR</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
      <XchgId>0</XchgId>
      <CreDtTm>2012-01-24T09:00:05.00+01:00</CreDtTm>
      <InitgPty>
        <Id>990001</Id>
      </InitgPty>
      <RcptPty>
        <Id>test_acceptor</Id>
      </RcptPty>
    </Hdr>
    <CxlAdvCspn>
      <Envt>
        <POIID>
          <Id>990001</Id>
        </POIID>
      </Envt>
      <Tx>
        <TxId>
          <TxDtTm>2012-01-24T09:00:00.00+01:00</TxDtTm>
          <TxRef>000001</TxRef>
        </TxId>
      </Tx>
    </CxlAdvCspn>
  </AcceptorCxlAdvCspn>
</Document>

```

## 8.15 Communication Security

### 8.15.1 Security Trailer

The following description is taken from the *EPAS Usage Guide* from page 288 onwards.

#### 8.15.1.1 Key Management

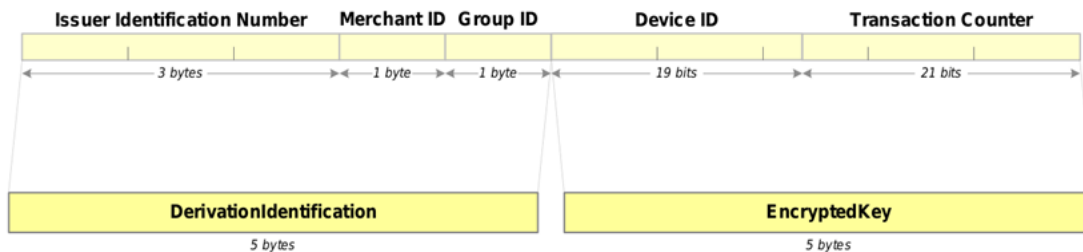
Test key identification is distinguished from production key by a name including the suffix “Test-Key”.

The DUKPT key management mechanism uses 10 bytes of information (Key Serial Number or KSN) sent by the *InitiatingParty* in the message to uniquely identified the derived key at the *RecipientParty*.

This KSN contains the following information:

- Issuer Identification Number (3 bytes): a collision free 6 digit number which will ensure the uniqueness of the KSN.
- Merchant ID (1 byte): can be used by an acquirer or manufacturer to differentiate merchants from each other.
- Group ID (1 byte): can be used by an acquirer or manufacturer to classify devices for a given merchants.
- Device ID (19 bits): can be used to identify a device inside a specific group ID.
- Transaction Counter (21 bits): the counter value can be used to detect message replay.

The 3 first elements (5 bytes) are sent in the *Recipient.KEK.KEKIdentification.DerivationIdentification* item of the EnvelopedData component, the last 2 elements (5 bytes) are sent in the *Recipient.KEK.EncryptedKey* of the EnvelopedData component.



**Figure 8.1:** Key Serial Number Details

After derivation of the resultant key, a XOR with the hexadecimal value 00000000 0000FF00 00000000 0000FF00 (MACmask) is applied to the resultant key in order to use a variant of the key for MAC computation.

The same key is used for the MAC of a message request and its corresponding message response, i.e. the Base Derivation Key (as the Terminal Initial Key) and the KSN are the same.

### 8.15.1.2 MAC Computation

The following explanation is taken from the *EPAS Usage guide*, page 296.

MAC computation uses Triple DES algorithm with double length key (112 Bit), using the retail CBC (Chaining Block Cipher) mode as defined in ISO 9807 and ANSI X9.19, on the result of the SHA-256 digest of the message body as defined in FIPS 180-1 and 2. Before encryption, the digest is padded according to the ISO/DIS 7816-4.

MAC computation and MAC verification use the same algorithm presented below.

MAC Computation Process:

- (i) Compute the SHA-256 digest *D* on the body of the message, including the XML envelope, and as transmitted by the transport level.



- For the MAC verification of a received message, the digest is computed on the body as received by the transport level.
  - For the MAC generation of a message to send, the body shall have no transformation after the computation of the digest.
- (ii) Padding of the data to encrypt  $D$ : the hexadecimal byte 80 is added to  $D$ . If the new length is not a multiple of 8,  $D$  is extended by null bytes (hexadecimal 0x00), to reach a length multiple of 8.
- (iii) The result  $D$  of the padded data is split in blocks of 8 bytes  $D_1 \dots D_n$ . With the left part  $K_L$  of key  $K$ , and initializing  $C_0$  by 8 null bytes, compute the sequence  $C_1 \dots C_{n-1}$ , where

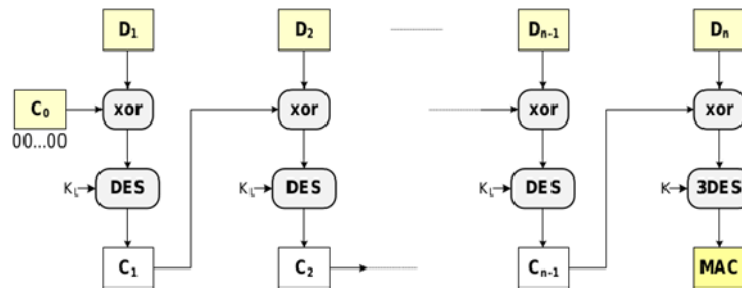
$$C_i = E_{K_L}(C_{i-1} \oplus D_i)$$

$E_{K_L}$  being the DES encryption with  $K_L$ .

- (iv) The MAC is the result of:

$$\text{MAC} = E_K(C_{n-1} \oplus D_n)$$

$E_K$  being the Triple-DES encryption with  $K$ .



**Figure 8.2:** MAC Computation Process

### 8.15.1.3 Example

We use a small example to show how to compute the MAC. The example is from the *EPAS Usage guide*.

The XML message is:

```

<DgnstcReq>
  <Env>
    <AcqrrParamsVrsn>2010-01-01T08:00:00</AcqrrParamsVrsn>
    <MrchntId>
      <Id>EPASMER001</Id>
      <Tp>MERC</Tp>
    </MrchntId>
    <POIID>
      <Id>66000001</Id>
      <Tp>OP0I</Tp>
  
```

```
<Issr>ACQR</Issr>
</POIID>
</Evt>
</DgnstcReq>
```

Derived key: 5E 64 F1 AB F2 5D 3B A1 7F 62 9E C2 B3 02 F8 EA.

The SHA256 digest of the *DiagnosticRequest* message body is:

```
0000 C4 11 A9 4F 56 97 8E A1 8B 9D CA F4 A0 DE 5B 44
0010 09 BE A9 93 87 58 1A CA E5 01 3D 4A 55 38 AF B0
```

This message is then padded with 0x80 followed by 7 null bytes:

```
0000 C4 11 A9 4F 56 97 8E A1 8B 9D CA F4 A0 DE 5B 44
0010 09 BE A9 93 87 58 1A CA E5 01 3D 4A 55 38 AF B0
0020 80 00 00 00 00 00 00 00 00
```

Now we encrypt the first 32 bytes using DES CBC and the left half of the shared key:

```
0000 0C 39 D3 CF 05 F9 F4 97 E0 1E 69 DE 5F 23 F8 72
0010 81 EC 98 C5 B4 12 CD A4 19 E8 06 D6 F2 03 9F B3
```

We encrypt the final 8 bytes using 3DES CBC and get:

```
0000 0C 39 D3 CF 05 F9 F4 97 E0 1E 69 DE 5F 23 F8 72
0010 81 EC 98 C5 B4 12 CD A4 19 E8 06 D6 F2 03 9F B3
0020 21 86 58 17 8E B7 E8 F6
```

The MAC is the last 8 bytes: 21 86 58 17 8E B7 E8 F6

The base64 conversion of this value is: IYZYF4636PY=

The resulting security trailer with no info about the KEK looks like:

```
<SctyTrlr>
  <CnttTp>AUTH</CnttTp>
  <AuthntcdData>
    <MACAlgo>
      <Algo>MCCS</Algo>
    </MACAlgo>
    <NcpsltdCntt>
      <CnttTp>DATA</CnttTp>
    </NcpsltdCntt>
    <MAC>IYZYF4636PY=</MAC>
  </AuthntcdData>
</SctyTrlr>
```

## 8.15.2 Encrypting the Communication

The communication between the terminal and the server can be secured using SSL encryption. If this shall be enabled, the integrator must send the CA certificate to Verifone Denmark. The

terminal does not use client certificate. The terminal will validate the server certificate and ensure that the server certificate is signed by the CA and that the Common Name (CN) is the same as the IP address of the server.

The terminal supports the cipher “AES256-SHA”. It is up to the integrator to make sure that the latest requirements from PAN Nordic and PCI regarding key sizes are met.

Please note that you should create a new certificate for each server installation. All of these certificates should be signed by the same CA certificate, which is the certificate that you sent to Verifone Denmark.

## 8.16 Flow of Transactions (Sequence Diagrams)

All transactions can be put into one of two boxes, depending on whether the request was successful or not. In a successful case we send the `AcceptorAuthorisationRequest` and receive an approved `AcceptorAuthorisationResponse` from the Acquirer. In all other cases, we finish the message exchange by sending an `AcceptorCompletionAdvice`, telling the Acquirer to abort the transaction. The possible different scenarios are shown below.

### 8.16.1 Successful Authentication

In Figure 8.3 is shown a successful transaction, thus no `AcceptorCompletionAdvice` is sent.

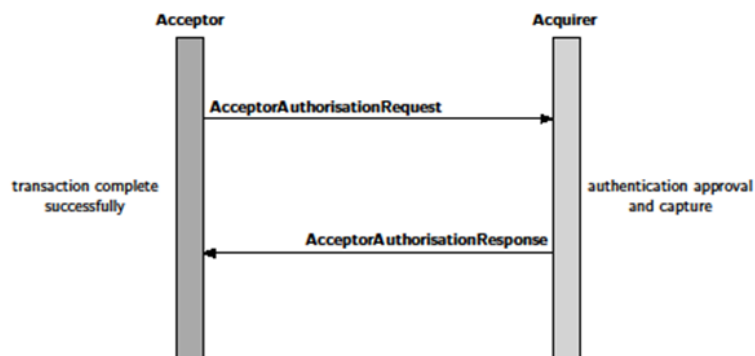


Figure 8.3: Successful authentication and money transfer

### 8.16.2 Failed Authentication

Figure 8.4 shows a transaction where the authentication fails. The Acquirer informs the Acceptor and the Acceptor acknowledges the response by sending an `AcceptorCompletionAdvice`.

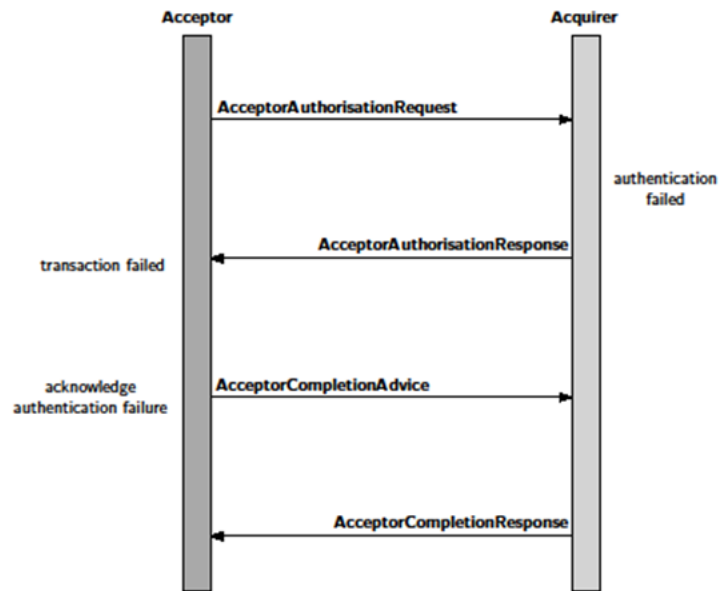


Figure 8.4: Authentication failure

### 8.16.3 No Authorisation Response

Figure 8.5 shows an example of a communication error. In this example the Acceptor never receives the AcceptorAuthorisationResponse from the Acquirer. The Acceptor handles this by sending an AcceptorCompletionAdvice to inform the Acquirer to reverse the transaction. The Acquirer must always accept this reversal and return an approval.

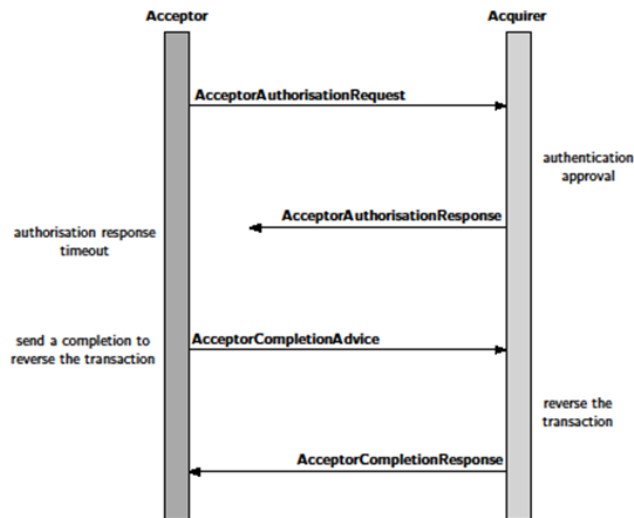


Figure 8.5: Example of communication error

### 8.16.4 No Completion Response

The next example scenario is shown in Figure 8.6 Here the transaction fails, either because the `AcceptorAuthorisationResponse` never arrives or because the authentication fails for some other reason. In this example the response never arrives. As the transaction fails, the Acceptor sends an `AcceptorCompletionAdvice`, but never receives a response. When this happens, the Acceptor resends the `AcceptorCompletionAdvice` until a response is received.

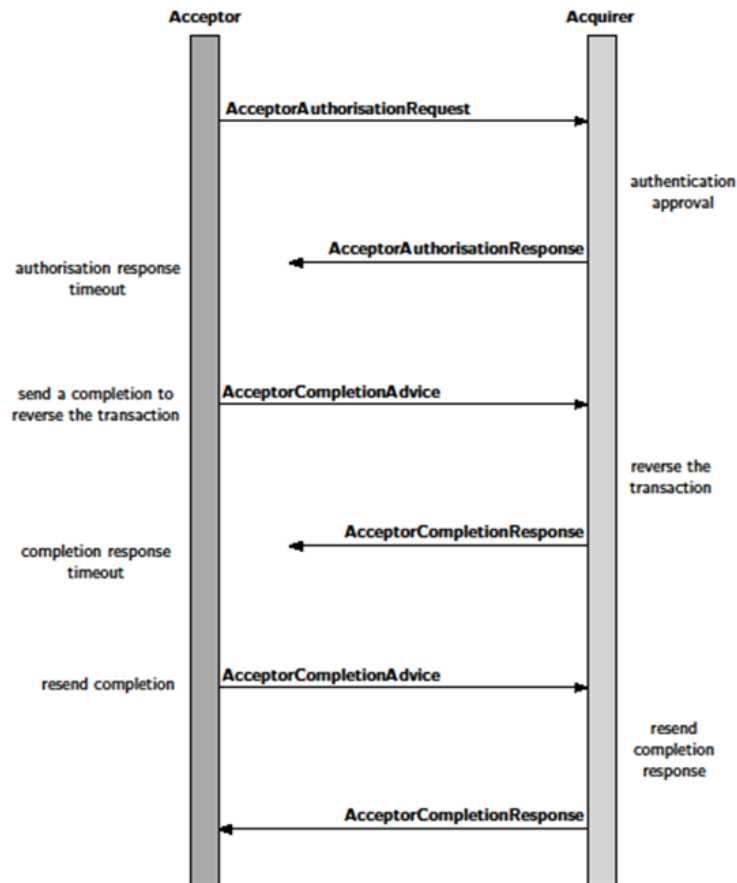


Figure 8.6: Example of retransmission

## 8.17 Suggestions for Additional Message Items

Message Item	XML Tag	Description
AuthorisationRequest		
Environment		
POI		

Message Item	XML Tag	Description
Identification		
...		
SystemName	SysNm	Common name assigned by the acquirer to the POI system, i.e. "Xenta", "Yomani".
Card		
PlainCardData		
...		
AdditionalCardData	AddtlCardData	UUID/RFIDID

### 8.17.1 Example with no PAN and UUID

**Listing 8.9:** Example Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AccptrAuthstnReq>
    <Hdr>
      <MsgFctn>FAUQ</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
      <XchgId>0</XchgId>
      <CreDtTm>2011-06-31T00:00:00.00+01:00</CreDtTm>
      <InitgPty>
        <Id>Point</Id>
      </InitgPty>
      <RcptPty>
        <Id>Host</Id>
      </RcptPty>
    </Hdr>
    <AuthstnReq>
      <Envt>
        <POI>
          <Id>
            <Id>990001</Id>
          </Id>
          <SysNm>Yomani</SysNm>
        </POI>
        <Card>
          <PlainCardData>
            <PAN>99990000000001</PAN>
            <XpryDt>2014-12</XpryDt>
          </PlainCardData>
          <AddtlCardData></AddtlCardData> <!-- UID -->
        </Card>
      </Envt>
      <Cntxt>
        <PmtCntxt>
          <CardDataNtryMd>CTLS</CardDataNtryMd>
        </PmtCntxt>
      </Cntxt>
    </AuthstnReq>
  </AccptrAuthstnReq>
</Document>
```

```

    <Tx>
      <TxCaptr>TRUE</TxCaptr>
      <TxTp>CRDP</TxTp>
      <MrchntCtgyCd>5691</MrchntCtgyCd>
      <TxId>
        <TxDtTm>2011-06-31T00:00:00.00+01:00</TxDtTm>
        <TxRef>1234567890</TxRef>
      </TxId>
      <TxDtls>
        <Ccy>DKK</Ccy>
        <TtlAmt>20.00</TtlAmt>
      </TxDtls>
    </Tx>
  </AuthstnReq>
</AccptrAuthstnReq>
</Document>

```

Message Item	XML Tag	Description
AuthorisationResponse		
TransactionResponse		
AuthorisationResult		
ResponseToAuthorisation		
Response		
ResponseReason	RspnRsn	
Action	Actn	
ActionType	ActnTp	<b>DISP</b> : Display a message. <b>PRNT</b> : Print a message.
MessageToPresent	MsgToPres	
MessageDestination	MsgDstn	<b>CDSP</b> : Cardholder display or interface. <b>CRCP</b> : Cardholder receipt. <b>MDSP</b> : Merchant display or interface. <b>MRCP</b> : Merchant receipt.
MessageContent	MsgCntt	Text or graphic data to be display or printed to the cardholder or the cashier. Maximum 256 characters.

### Listing 8.10: Example Response

```

<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:iso:std:iso:20022:tech:xsd:caaa.003.001.01">
  <AccptrAuthstnRspn>
    <Hdr>
      <MsgFctn>FAUP</MsgFctn>
      <PrtcolVrsn>1.0</PrtcolVrsn>
      <XchgId>0</XchgId>
      <CreDtTm>2011-06-31T00:00:00.00+01:00</CreDtTm>
      <InitgPty>

```

```

        <Id>Point</Id>
    </InitgPty>
    <RcptPty>
        <Id>Host</Id>
    </RcptPty>
</Hdr>
<AuthstnRspn>
    <Envt>
        <POIID>
            <Id>990001</Id>
        </POIID>
    </Envt>
    <Tx>
        <TxId>
            <TxDtTm>2011-06-31T00:00:00.00+01:00</TxDtTm>
            <TxRef>1234567890</TxRef>
        </TxId>
        <TxDtls>
            <Ccy>EUR</Ccy>
            <TtlAmt>20.00</TtlAmt>
        </TxDtls>
    </Tx>
    <TxRspn>
        <AuthstnRslt>
            <RspnToAuthstn>
                <Rspn>APPR</Rspn> <!-- Status = OK -->
                <RspnRsn>1</RspnRsn> <!-- StatusTextID = 1 -->
            </RspnToAuthstn>
        </AuthstnRslt>
        <Actn>
            <ActnTp>DISP</ActnTp>
            <MsgToPres>
                <MsgDstn>CDSP</MsgDstn>
                <MsgCntt>2</MsgCntt> <!-- LogoID = 2 -->
            </MsgToPres>
        </Actn>
    </TxRspn>
</AuthstnRspn>
</AccptrAuthstnRspn>
</Document>

```



# 9 | Point Payment Interface

## 9.1 Preface

Point Payment Interface (PPI) is an extended integration protocol for integration between POS application and the EPS application. The protocol is included in Point Payment Module (EPS) that interfaces the different terminal types in Points product range - both PSAM and non PSAM based terminals.

Besides the the actual payment terminal interface the Payment Module includes additional modules such as:

- Admin interface module that enables remote administration and support via an administrative application. This application is provided as a part of the solution.
- Logging facilities that makes it possible to save and retrieve logs on different levels.
- SNMP interface to provide data for network monitoring.
- POS interface to handle communication with the POS system.
- Opportunity to select different terminals integration interfaces - as long as these are available.

With EPS application refers to the software that handles the interaction with the payment device - regardless of the type of device. Be it a dedicated terminal with embedded software, or a distributed solution with separate card reader, display, printer, etc.

The PPI protocol is based on TCP connections and the XML data format. All XML messages exchanged via TCP must be tagged with a 4 byte prefix (big indian – most significant byte first). The 4 byte prefix is counted in the length declaration.

All messages in the PPI protocol are defined as request and a subsequent response. The requests are in this paper divided into 2 categories - mandatory or "implementation specific" request, elements and or attributes - the frase "implementation specific" implies that each specific project must include a list of which that should be implemented in the solution. In this paper the term "Optional" is used as a synonym for the frase "implementation specific" - this indicated by a blue headline for the section title.

It is recommended that when an integration project start draw up a design guide of the Request /Response to be implemented and also defines which elements and attributes to be used. As an example a card payment scheme could be shown as...

### 9.3.1 CardRequest(CardPayment) + (A.1, A.2, A.5, E.1, E.2, E.3, E.4, E.5)

In this case indicating to implement all Attributes and elements except attributes A.3 + A.4 and except elements E.6 + E.7 + E.8. Request and Response where there is no set numbers for attributes or elements must be implemented in full - if implemented

## Revision

3.7.15	Changes in CardResponse, Merchant number is now included in CardResponse
3.7.12	PPI now follows the version number of the VIP package. PostPurchase and PostRefund is now added in RequestType. Prepaid scanned barcode is now implemented.
1.0.29	Support for key entered transactions added.
1.0.28	ReloadConfig method added.
1.0.27	TerminalPort option added in GetInitializedRequest.
1.0.26	Fixed ports for request and device ports added for client-server setups.
1.0.25	Added authorisation transaction types.
1.0.24	GetInitializedTerminal request changed. TCP port taken out and Terminal number added. Clarification of send and receive on channel 1.
1.0.23	Cancellation functionality added.
1.0.22	Clarification of functionality. Init, Open, Close and UnInit changed to GetInitializedTerminal.
1.0.21	DLL update added.
1.0.20	AskForAuthCode and trace level added.
1.0.19	

**Revision**

	SNMP settings added to config file.
1.0.18	Log file clean up.
1.0.17	FOS5 administration command table added. PPIService checks for updates every hour.
1.0.16	DeviceRequests removed from ServiceResponse. These are sent on channel 1.
1.0.15	Section about update logic added. Log file section added. TransactionID element added. Updated XML example.
1.0.14	OpenTerminal and CloseTerminal requests added. Added configuration parameters.
1.0.13	DailyLog and LogCleanUpAfterDays added to config. How to register the FOS5 DLL.
1.0.12	Section 7.3 (Config.xml) removed. Dublicate to section 10. Signatur parameter updated to Signature. SendAdviceFlag element added. AdviceFlag Request and Response (optional) added. Updated Admin Command table. Updated Diagnosis Response informations.
1.0.11	Corrected so that the value returned is the Command value. Clarification about how SignatureOnOff and OfflineOnOff works. Encoding in XML examples are updated to ISO-8859-15.
1.0.10	Elements that are not yet available are marked red. TCC and AcquiererID. Command attribute added to GetMenu request. Default error messages section added.

**Revision**

	Configuration via the config.xml file added.
1.0.09	PPIService section added.
1.0.08	Updated command table. Stan added to CardResponse.
1.0.07	Reversal section removed.
1.0.06	SubCode (section 4.7) table added.
1.0.05	Correcting typos and removed WorkstationID property. Added Administration command table.
1.0.04	Updated initialize terminal request port and XML examples.
1.0.03	Section "Configuration and Deployment" added.
1.0.02	TransactionID is added to the CardRequest and CardResponse (Attribute 4). GetPreviousResult function added to interface. EReceiptToken added to CardResponse. Obsolete section regarding Capture Data is deleted.
1.0.01	CardRequest/Cardresponse: TransactionCurrency made as XML tag instead of XML Property.
1.0.00	Initial document made by Morten Kalland.

## 9.2 System Concept

As mentioned in the introduction, the PPI protocol is based on communication between the POS and EPS application via TCP and with XML request and responses. This interface is divided into 2 channels - TCP connection 0 and 1. An action (request) is always initiated from the POS application via channel 0, where the EPS application is server and the POS application is client. Any action is finished when the EPS application returns a response for the overall action, and then the channel is closed. An Action/Request must be tagged by a unique RequestID and the response must contain the same tag.

During this action the EPS initiates a secondary channel 1 towards the POS application, which is used for printing, showing display messages and so forth. These messages are labeled device request and the POS (server) acts as a “device proxy” for the EPS application (Client). Each DeviceRequest must be replied by a DeviceResponse indicating if the request was received as intended. The Device Request connection on Channel 1 is open during the entire transaction on Channel 0 and is closed just before sending the overall result.1. Each of these DeviceRequest must contain the RequestID referring to the overall Request.

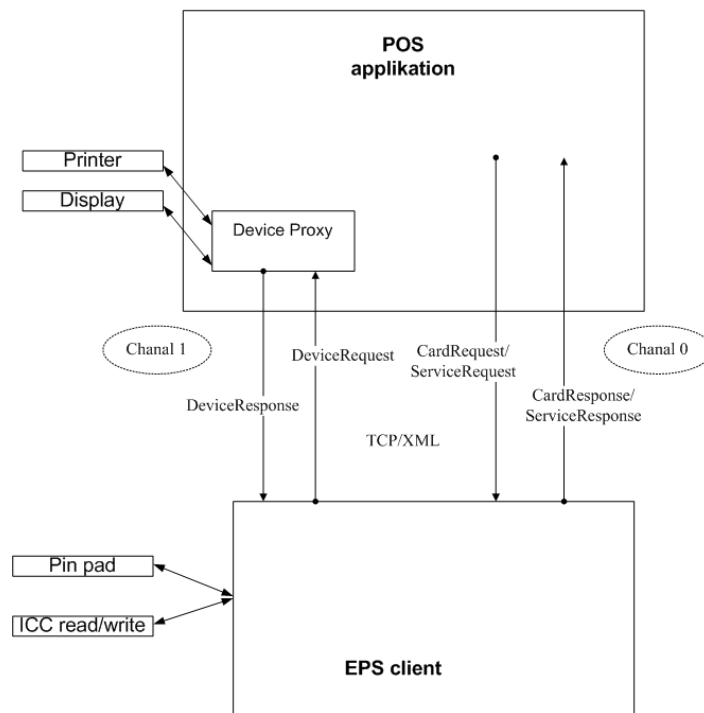


Figure: 9.2 The pin pad, Payment Card reader is handled by and OTRS compliant application, that provides an interface to the EPS application.

## Connection Proces model

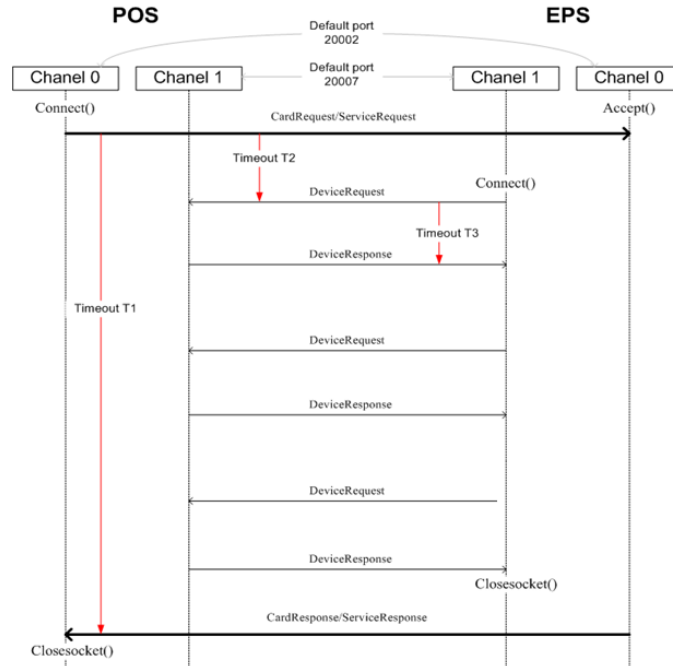


Figure: 9.2 PPI flow model illustrating the process regarding the channel 0 and 1.

Timeout	Description
<b>Timeout 0</b>	Is maximum interval between successful <code>Connect()</code> and reception of an valid XML message. In case of a timeout the connections should be closed.
<b>Timeout 1</b>	Is maximum interval between start of a <code>CardService/Service-Request</code> to reception of a <code>CardService/Service-Response</code> . In case of a timeout the POS closes the connection.
<b>Timeout 2</b>	Is maximum interval between last message from POS before response from EPS. In case of a timeout the POS closes the connection.
<b>Timeout 3</b>	Is maximum interval between <code>DeviceRequest</code> and <code>DeviceResponse</code> . In case of a timeout the EPS Closes channel 1.

## 9.3 CardRequest and CardResponse

CardRequest is the general term for actions relating to payments that always are initiated from the POS application via Channel 0.

The CardRequest is used with a number of Request types – each having a set of elements and attributes in the XML message.

Optional elements and attributes are shown in blue text color.

Elements and attributes that are not yet available are shown in red text color.

There are 4 different CardRequests:

- CardPayment
- CardRefund
- AbortRequest
- Cancellation

### 9.3.1 CardRequest (Payment)

The Request Type Payment is used to initiate a payment transaction. See elements and attributes below.

RequestType		Attribute - A.1
Value	Implies	
Text	Defines the type of request - in this case Payment	

PosID		Attribute - A.2
Value	Implies	
AN8	Unique value for each POS workstation that in combination with the RequestID secures a unique identifier.	

RequestID		Attribute - A.3
Value	Implies	
AN12	RequestID identifies the specific CardService/Service-Request and must be unique within each session. Is returned as a part of the response.	

TransactionID		Attribute - A.4
Value	Implies	
AN40	TransactionID is a logical ID provided by the POS system and should be unique for each transaction. This value will be returned in the CardResponse and is a prerequisite for the GetPreviousResult ServiceRequest.	

BatchNo (optional)		Element - E.1
--------------------	--	---------------

<b>Value</b>	<b>Implies</b>
N12	If implemented this batch number replaces the auto generated batch number in the terminal.

<b>ForcedCVM (Optional - used to assign cardholder verification method)</b>		Element - E.2
<b>Value</b>	<b>Implies</b>	
Default	Default transaction type where the PSAM and cards decides – In Denmark this normally results to pin code verification.	
Pin	Forced Pin – If supported the transaction will be forced to use Pin verification.	
Signature	Forced Signature – If supported the transaction will be forced to signature verification.	

<b>ForcedCapability (Optional - used to assign transaction capability)</b>		Element - E.3
<b>Value</b>	<b>Implies</b>	
Default	Default transaction which equals not assigning this element – In denmark this normally is equal to an online transaction.	
Online	Forced online – If supported the transaction will be forced online.	
Offline	Forced offline – If supported the transaction will be forced offline.	

<b>VatAmount (Optional – used in swedish context)</b>		Element - E.4
<b>Value</b>	<b>Implies</b>	
Amount	Specifies the value added tax amount. This can also be assigned via the GetCardConfirmation DeviceRequest.	

<b>TransactionAmount</b>		Element - E.5
<b>Value</b>	<b>Implies</b>	
Amount	The amount that is to be used in the transaction. “.” as decimal separator. The amount includes VatAmount. The Currency must be specified as an attribute to the TransactionAmount element.	

<b>CashBackAmount (Optional)</b>		Element - E.6
<b>Value</b>	<b>Implies</b>	
Amount	The cash back amount. This amount is not included within the transaction amount.	

<b>TransactionCurrency</b>		Element - E.7
<b>Value</b>	<b>Implies</b>	
Amount	Is the transaction currency for all the amounts.	

<b>TokenData</b>		Element - E.7
<b>Value</b>	<b>Implies</b>	
Base64 string	Token data used for authorization transactions (SuppAuthorization, CaptureAuthorization and CancelAuthorization)	



KeyEntered		Element - E.7
Value	Implies	
String "Y" or "N"	Whether the transaction should be key entered or not. Default is "N".	

ScanBarcode		Element - E.7
Value	Implies	
String barcode	Barcode for prepaid card. Also used for manuel Swipp transactions: sw:4530303030.	

### Example 1 – CardRequest

```
<?xml version="1.0"?>
<CardRequest RequestType="Payment" PosID="Kasse1" RequestID="1" TransactionID="201308131001">
  <BatchNo>789012</BatchNo>
  <ForcedCVM>Default</ForcedCVM>
  <ForcedCapability>Default</ForcedCapability>
  <VatAmount>40.00</VatAmount>
  <TransactionAmount>200.00</TransactionAmount>
  <CashBackAmount>1000.00</CashBackAmount>
  <TransactionCurrency>DKK</TransactionCurrency>
  <TokenData></TokenData>
  <KeyEntered></KeyEntered>
  <ScanBarcode></ScanBarcode>
</CardRequest>
```

If TransactionAmount is set to 0 when initiating a CardRequest then the amount must be added in the DeviceRequest GetCardConfirmation. See section 9.4.1

### 9.3.2 CardResponse (Payment)

There are 3 different possible outcomes of a Payment transaction (OverallResult):

- Success - The transaction is authorized.
- Failure - The transaction is not authorized.
- Aborted - The transaction was aborted by the customer or the clerk.

The elements and attributes that are explained in the previous section is not included below.

RequestType		Attribute - A.1
Value	Implies	
Text	Defines the type of request - in this case Payment.	

PosID		Attribute - A.2
Value	Implies	
AN8	Unique value for each POS workstation that in combination with the RequestID secures a unique identifier.	

---

<b>RequestID</b>		Attribute - A.3
<b>Value</b>	<b>Implies</b>	
AN12	RequestID identifies the specific CardService/Service-Request and must be unique. Is returned as a part of the response.	

<b>TransactionID</b>		Attribute - A.4
<b>Value</b>	<b>Implies</b>	
AN40	TransactionID is a logical ID provided by the POS system and should be unique for each transaction. This value will be returned in the CardResponse and is a prerequisite for the GetPreviousResult ServiceRequest.	

<b>OverallResult</b>		Attribute - A.5
<b>Value</b>	<b>Implies</b>	
Success	The transaction is authorized.	
Failure	The transaction is not authorized.	
Aborted	The transaction was aborted by the customer or the clerk.	

<b>SubCode (optional)</b>		Attribute - A.6
<b>Value</b>	<b>Implies</b>	
AN	An option to save extra return codes from the internal application handling like PPI error codes.	

<b>Asw1Asw2 (optional)</b>		Attribute - A.7
<b>Value</b>	<b>Implies</b>	
AN4	Addition transaction code from issuer host.	

<b>ResponseDetails – TerminalID (optional)</b>		Element - E.1
<b>Value</b>	<b>Implies</b>	
Text	Terminal identification provided from the terminal.	

<b>ResponseDetails - RefNo</b>		Element - E.2
<b>Value</b>	<b>Implies</b>	
Text	Number that contains transaction reference number. Very useful in terms for reconciliation or in case of problems.	

<b>ResponseDetails - PSAM (optional)</b>		Element - E.3
<b>Value</b>	<b>Implies</b>	
Text	A unique number indentifying the PSAM used during the transaction. Only feasible in a NETS solution – Handy for tracking purposes.	

---

<b>ResponseDetails - Merchant (optional)</b>		Element - E.4
<b>Value</b>	<b>Implies</b>	
Text	A number identifying the merchant.	

<b>ResponseDetails - TotalAmount</b>		Element - E.5
<b>Value</b>	<b>Implies</b>	
Amount	The total amount includes all amounts in the transaction. This amount includes the transaction amount + fee + cashback. The Currency must be specified as an attribute to the TotalAmount element.	

<b>ResponseDetails - TransactionAmount</b>		Element - E.6
<b>Value</b>	<b>Implies</b>	
Amount	Is the transaction amount.	

<b>ResponseDetails – FeeAmount (optional)</b>		Element - E.7
<b>Value</b>	<b>Implies</b>	
Amount	Contains optional fee amount added during the transaction.	

<b>ResponseDetails – VatAmount (optional)</b>		Element - E.8
<b>Value</b>	<b>Implies</b>	
Amount	Contains optional VAT amount.	

<b>ResponseDetails – CashBackAmount (optional)</b>		Element - E.9
<b>Value</b>	<b>Implies</b>	
Amount	Contains optional cash back amount.	

<b>ResponseDetails - AcquirerID (optional)</b>		Element - E.10
<b>Value</b>	<b>Implies</b>	
Text	Optional acquirer identification.	

<b>ResponseDetails - CardPan</b>		Element - E.11
<b>Value</b>	<b>Implies</b>	
Text	Truncated card pan number.	

<b>ResponseDetails - CardCircuit (optional)</b>		Element - E.12
<b>Value</b>	<b>Implies</b>	
N3	The Card Circuit is an optional 3 digit number identifying the card type – eg. 001 equals dankort.	

<b>ResponseDetails - ApprovalCode (optional)</b>		Element - E.13
<b>Value</b>	<b>Implies</b>	

AN6	The transaction approvalcode/authorization – “000000” equals success.
-----	---

ResponseDetails - TCC (optional)		Element - E.14
Value	Implies	
AN3	Contains data regarding "Card Data Entry – see appendix D.	

ResponseDetails - EReceiptToken (optional)		Element - E.16
Value	Implies	
AN50	Token to be used when implementing Electronic Receipt solutions.	

ResponseDetails - Stan (optional)		Element - E.15
Value	Implies	
Text	Unique identifier for the transaction.	

ResponseDetails - TransactionCurrency		Element - E.15
Value	Implies	
Amount	Is the transaction currency for all the amounts.	

## Example 2 - CardResponse

```
<?xml version="1.0" encoding="ISO-8859-15" standalone="yes"?>
<CardResponse RequestType="Payment" PosID="Kasse1" RequestID="1" TransacionID="201308131001"
OverallResult="Success" SubCode="" Asw1Asw2="0000">
  <ResponseDetails>
    <TerminalID>00086001</ TerminalID>
    <RefNo>001413</RefNo>
    <PSAM>5374978-0000101794</PSAM>
    <Merchant>0001978543</Merchant>
    <TotalAmount>1200.00</TotalAmount>
    <TransactionAmount>200.00</ TransactionAmount >
    <FeeAmount>0.00</FeeAmount>
    <VatAmount>40.00</VatAmount >
    <CashbackAmount>1000.00</CashBackAmount >
    <AcquirerID>343534</AcquirerID>
    <CardPAN>4539xxxxxxxx4507</CardPAN>
    <CardCircuit>003</CardCircuit>
    <ApprovalCode>0000</ApprovalCode>
    <TCC> I05</TCC>
    <EReceiptToken></EReceiptToken>
    <Stan></Stan>
    <TransactionCurrency>DKK</TransactionCurrency>
  </ResponseDetails>
</CardResponse>
```

### **9.3.3 CardRequest (Refund)**

This request type is used to transfer an amount from a merchant to an customer – customer payment card.

The request structure is the same as with Payment.

### **9.3.4 CardRequest (BalanceQuery)**

This Request type is used to retrieve the balance for at “gift card”. This request type is optional and only requerid if the payment solution should support “gift card” handling. On success the EPS application returns a receipt containing the current balance.

### **9.3.5 CardRequest (Abort)**

It is a key element in relation to the Danish certification of a payment solution, that it's possible to attempt to abort the transaction from the POS application - this at any time during the course of a transaction course of a transaction.

It is important to emphasize that in the case of an attempt to abort the transaction, the POS application must await the outcome of the transaction to see if it succeeded in aborting. Whether it's possible or not depends by an interaction between the EPS and OTRS application.

The result is always return inherrent in the CardResponse for the initial request – like CardPayment.

### **9.3.6 CardRequest (Cancellation)**

Cancellation of previous transaction.

### **9.3.7 CardRequest (Authorisation)**

Make an authorisation.

Returns a Base64 encoded token.

### **9.3.8 CardRequest (SuppAuthorisation)**

Make a supplemental authorisation.

Provide token data in TokenData tag.

Returns a new Base64 encoded token, from the original authorisation token.

### **9.3.9 CardRequest (CaptureAuthorisation)**

Captures an authorisation.

Provide token data in TokenData tag.

### **9.3.10 CardRequest (CancelAuthorisation)**

Cancellation of previous transaction.

Provide token data in TokenData tag.

### 9.3.11 CardRequest (PrepaidScanPurchase)

This Request type is used to complete a purchase on a prepaid card. Provide barcode in ScanBarcode tag.

### 9.3.12 CardRequest (PrepaidScanRefund)

This Request type is used to transfer an amount for a merchant to an customers prepaid card. Provide barcode in ScanBarcode tag.

### 9.3.13 CardRequest (PrepaidScanAuthorisation)

This Request type is used to make an authorisation on a prepaid card. Provide barcode in ScanBarcode tag.

### 9.3.14 CardRequest (PostPurchase)

This Request type is used to make a post/late purchase on a token/authorisation. Provide token data in TokenData tag.

### 9.3.15 CardRequest (PostRefund)

This Request type is used to make a post/late refund on a token/authorisation. Provide token data in TokenData tag.

### 9.3.16 SubCode table

SubCode	SubCodeMessage
1	
2	
3	Unknown ComType.
4	Terminal connect: No INI file.
5	Terminal connect: INI file read error.
6	Terminal connect: Com port init failure.
7	Terminal connect: Software not compatible.
8	Terminal connect: Terminal not responding.
9	Terminal connect: Com port already open.
10	Terminal connect: Data link error. If Yomani terminal remember MU/ECR interface.
11	Terminal connect: Function not possible.
12	Terminal connect: Timeout in communication. If Yomani terminal remember MU/ECR interface.
13	Terminal connect: No license.
14	Terminal connect: Internal error.
15	Terminal connect: System error.

16	Terminal connect: Connect init value.
17	Terminal could not be uninitialized.
18	Invalid init/uninit terminal request.
19	Terminal could not be connected.
20	Terminal could not be disconnected.
21	Invalid cardnumber.
22	Error in XML.
23	Transaction type not recognized.
24	Terminal not connected.
25	No TCP connection to POS.
26	flxTerminalProperties error.
27	Error in TPROPS_LABELS.
28	Invalid requestID.
29	Invalid transactionID.
30	Invalid VatAmount.
31	Invalid TransactionAmount.
32	Invalid CashbackAmount.
33	Cardtransaction already initiated.
34	BalanceQuery not supported on FOS5 terminals.
35	
36	
37	
38	
39	
40	
41	
42	
43	

## 9.4 DeviceRequest and DeviceResponse

When the POS application starts a CardRequest or a ServiceRequest connection to EPS client, then the client EPS immediately after must establish a DeviceRequest connection to the POS application.

The DeviceRequest connection is during the transaction used to access different devices in the POS application like display and printer. The display data can be information to be displayed or more interactive data like menus, dialogs, etc.

Each DeviceRequest from the EPS client must be answered by a DeviceResponse from the POS application. The response must contain an attribute "OverallResult" which can take the values success or failure. Failure indicates communication problems and should result in a secondary attempt to transfer the data.

The more logical elements in the reply is saved in other attributes and elements – see below.

### 9.4.1 DeviceRequest (GetCardConfirmation)

If the carddata is present during the transaction it's an option to implement the DeviceRequest – GetCardConfirmation. The purpose is to enable the POS application to:

- Evaluate if the specific transaction should be continued or be rejected based on PAN and/or CardCircuit.
- Offer an option to evaluate if the amount is beyond defined boundaries.
- Initiate a "local card transaction", where the POS backend system handles the card validation logic.

RequestType		Attribute - A.1
Value	Implies	
Text	Defines that an input is requested - GetCardConfirmation.	

PosID		Attribute - A.2
Value	Implies	
Text	Unique value for each POS workstation that in combination with the RequestID secures a unique identifier.	

RequestID		Attribute - A.4
Value	Implies	
Text	Referring to the initial CardRequest ID.	

SequenceID		Attribute - A.5
Value	Implies	
Text	Sequence number for each deviceRequest in a process started by CardRequest or ServiceRequest.	

Output - CardCircuit		Element - E.1
Value	Implies	



N3	Example "001" (Dankort) - Card type identifier like Mastercard, Diners, etc.
----	--

Output - CardNumber		Element - E.2
Value	Implies	
AN	Example "4571991234562025" or masked like "457199XXXXXX2025".	

Output - LocalCard		Element - E.3
Value	Implies	
"Y"/"N"	N = not local card   Y = local card.	

Output - AdditionalCardInfo		Element - E.4
Value	Implies	
Input	Addition information retrieved from the card - often used with local cards.	

Output - CardTracks (1-3)		Element - E.5
Value	Implies	
Input	Optional information with the content of the card track 1-3. Elements of this information is likely to be masked/criptated depending on the card type.	

### Example 3 – DeviceRequest – GetCardConfirmation

```
<?xml version="1.0" standalone="yes"?>
<DeviceRequest RequestType="GetCardConfirmation" PosID="Kasse 1" RequestID="1" SequenceID="1">
  <Output>
    <CardCircuit>001</CardCurcuit>
    <CardNumber>457199AAAAAA2025</CardNumber>
    <LocalCard>N</LocalCard>
    <AdditionalCardInfo></AdditionalCardInfo>
    <CardTrack1>Track1 information...</CardTrack1>
    <CardTrack2>Track2 information...</CardTrack2>
    <CardTrack3>Track3 information...</CardTrack3>
  </Output>
</DeviceRequest>
```

#### 9.4.2 DeviceResponse (GetCardConfirmation)

If the GetCardConfirmation is implemented in the solution, then the POS application must respond with a DeviceResponse.

Note that if the transaction amount is not specified it must be so at in this DeviceResponse.

RequestType		Attribute - A.1
Value	Implies	

Text	GetCardConfirmation.
------	----------------------

<b>PosID</b>	Attribute - A.2
<b>Value</b>	<b>Implies</b>
Text	Unique value for each POS workstation that in combination with the RequestID secures a unique identifier.

<b>RequestID</b>	Attribute - A.3
<b>Value</b>	<b>Implies</b>
Text	Referring to the initial CardRequest ID.

<b>SequenceID</b>	Attribute - A.4
<b>Value</b>	<b>Implies</b>
Text	Must be the same number as in the ServiceRequest.

<b>OverallResult</b>	Attribute - A.5
<b>Value</b>	<b>Implies</b>
"Success"/ "Failure"	Success or Failure. Success indicates succesfull exchange of request /response data – failure the opposite

<b>Input - TransactionAmount</b>	Element - E.1
<b>Value</b>	<b>Implies</b>
Amount	Example 20000 is 200,00 DKK. This amount does not include the optional FeeAmount, Discount, OverPaymentnt etc. The amount includes the optional vat amount. This amount can be changed from the one given a CardRequest start. If there is no changes the field can be left empty.

<b>Input - VatAmount</b>	Element - E.2
<b>Value</b>	<b>Implies</b>
Amount	Option to define the vat amount. If there is no changes the field can be left empty.

<b>Input - FeeAmount</b>	Element - E.3
<b>Value</b>	<b>Implies</b>
Amount	It's optional to add a fee amount. In other solutions the terminal calculates the fee amout – If any. If there is no changes the field can be left empty.

<b>Input - CashbackAmount</b>	Element - E.4
<b>Value</b>	<b>Implies</b>
Amount	Option to define the cash back amount – if any. If there is no changes the field can be left empty.

Input - CardConfirmResult		Element - E.5
Value	Implies	
Text	Y or N. Y = Card Accepted   N = Card Rejected	

#### Example 4 – DeviceResponse GetCardConfirmation

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceResponse RequestType="GetCardConfirmation" PosID="Kasse 1" RequestID="1" SequenceID="1"
OverallResult="Success">
  <Input>
    <TransactionAmount>200.00</TransactionAmount>
    <VatAmount>50</VatAmount>
    <FeeAmount>7.00</FeeAmount>
    <CashbackAmount>1000.00</CashbackAmount>
    <CardConfirmResult>Y</ CardConfirmResult >
  </Input>
</DeviceResponse>
```

Important: All amounts must be with 2 decimal places, and the decimal separator is a dot.

Please note that the Danish rules about payment transactions allow the transmission of a card charge to the cardholder. This only applies to credit cards. GetCardConfirmation allows the POS application to calculate the fee and return the amount in Device Response. In practice, fee calculation today so complex that any calculation is left to the terminal application. In this case the fee amount is returned as a part of the CardResponse. Se Fee shall be refunded in this case the same with Card Service Response terminating the transaction. Se section 9.3.2

### 9.4.3 DeviceRequest (Print)

The Print DeviceRequest is mandatory – at any moment during the transaction the POS must provide an option to print. The RequestType is “output” and the attribute “OutDeviceTarget” is a Printer.

In principle there can be defined as many printers as the POS system supports. By default the PPI protocol has 1 defined printer:

- Printer: This is the POS printer which is used for sales receipts, administrative receipts, etc.

The Output element has to more attributes:

- ImmediatePrint: 2 possible values True/False. If true then the receipt must be printed instantly, otherwise the print can be handled at the end at the transaction. The flag is true on the first receipt in a signature transaction.
- Complete: 2 possible values True/False. If When false is the receipt is not complete and additional data will be transferred in subsequent calls.

#### IMPORTANT!!

All transferred receipts from the terminal must be printed regardless of the transaction outcome.

If this is neglected the solution will not be approved by authorities.  
Receipts are printed unchanged.

### Example 5 - DeviceRequest (Print)

```
<?xml version="1.0" standalone="yes"?>
<DeviceRequest RequestType="Print" PosID="Kasse 1" RequestID="1" SequenceID="2"
TransactionID="123">
  <Output OutDeviceTarget="Printer" ImmediatePrint="True" Complete="True">
    <TextLine>PBS</TextLine>
    <TextLine>testcenter</TextLine>
    <TextLine>2750 Ballerup</TextLine>
    <TextLine>TLF. 44892299</TextLine>
    <TextLine/>
    <TextLine>2009-08-18 15:20</TextLine>
    <TextLine/>
    <TextLine>KOEB DKK 2,35</TextLine>
    <TextLine> ----</TextLine>
    <TextLine>VisaDankort PSN: 00</TextLine>
    <TextLine>XXXX XXXX XXXX 4507</TextLine>
    <TextLine>TERM: 00000040-001411</TextLine>
    <TextLine>I@5 PBS NR:0001978543</TextLine>
    <TextLine/>
    <TextLine> KORTHOLDERS SIGNATUR:</TextLine>
    <TextLine/>
    <TextLine/>
    <TextLine>.....</TextLine>
    <TextLine>ATC:00476 AED:000000</TextLine>
    <TextLine>AID: A0000000031010</TextLine>
    <TextLine>PSAM: 5374978-0000101794</TextLine>
    <TextLine>ARC:Y3</TextLine>
    <TextLine>REF:001411 AUTORISERET</TextLine>
    <TextLine/>
    <TextLine> FORRETNINGENS NOTA</TextLine>
  </Output>
</DeviceRequest>
```

## 9.4.4 DeviceResponse (Printer)

### Example 6 - DeviceResponse (Printer)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceResponse RequestType="Print" PosID="Kasse 1" RequestID="1" SequenceID="2"
OverallResult="Success" TransactionID="123">
  <Output OutDeviceTarget="Printer" OutResult="Success"/>
</DeviceResponse>
```

The attribute OverallResult indicates whether DeviceRequest was a "Success", i.e. the data was received, understood and handled.

## 9.4.5 DeviceRequest (Display)

### Example 7 - DeviceRequest (Display)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceRequest RequestType="Display" PosID="Kasse 1" RequestID="1" SequenceID="3"
TransactionID="123">
  <Output OutDeviceTarget="CashierDisplay">
    <TextLine Erase="True" TimeOut="5" TextCode="6005">Transaction starting...</TextLine>
  </Output>
</DeviceRequest>
```

## 9.4.6 DeviceResponse (Display)

### Example 8 - DeviceResponse (Display)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceResponse RequestType="Display" PosID="Kasse 1" RequestID="1" SequenceID="3"
"OverallResult="Success" TransactionID="123">
  <Output OutDeviceTarget="CashierDisplay" OutResult="Success"/>
DisplayRequest
</DeviceResponse>
```

## 9.4.7 DeviceRequest (GetKeyInput)

The command "GetKeyInput" provides an option to get a text string from the POS. It will in most cases be used to obtain an authorization code. Along with the "Input" section will also be able to send text. Therefore the POS application must be able to handle this.

### Example 9 - DeviceRequest (GetKeyInput)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceRequest RequestType="GetKeyInput" PosID="Kasse 1" RequestID="1" SequenceID="4">
  <Output OutDeviceTarget="CashierDisplay">
    <TextLine Erase="True">Assign Enter Authorizations code with up to 6 digits</TextLine>
  </Output>
</DeviceRequest>
```

## 9.4.8 DeviceResponse(GetKeyInput)

### Example 10 - DeviceResponse (GetKeyInput)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceResponse RequestType="GetKeyInput" PostID="Kasse 1" RequestID="1" SequenceID="4"
OverallResult="Success">
  <Input>
    <Value>123456</Value>
  </Input>
</DeviceResponse>
```

## 9.4.9 DeviceRequest (GetConfirmation)

This DeviceRequest is typically used to allow the shop assistant to confirm/reject a given choice.

### Example 11 - DeviceRequest (GetConfirmation)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceRequest RequestType="GetConfirmation" PosID="Kasse 1" RequestID="1" SequenceID="5">
  <Output OutDeviceTarget="CashierDisplay">
    <TextLine Erase="true" TimeOut="5">Is receipt signed</TextLine>
  </Output>
</DeviceRequest>
```

## 9.4.10 DeviceResponse (GetConfirmation)

### Example 12 - DeviceResponse (GetConfirmation)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<RequestType="GetConfirmation"PosID="Kasse 1" RequestID="1" SequenceID = "5"
OverallResult="Success">
  <Input>
    <Value>Y</Value>
  </Input>
</DeviceResponse>
```

## 9.4.11 DeviceRequest (GetMenu)

This DeviceRequest enables the EPS application to send/present a list menu for the sales clerk, and the POS application then returns the selected item in the list.

Command (0) is default = do nothing.

The command list is managed via AdminFunctions.xml

```
...
<EndOfDay Description="Dagsafslutning" MenuTitle="Dagsafslutning" FctId="1" TermObjectType="1"
MenuActive="1" Country="DK" />
...
```

FctId is the Command number (Table 9.7 lists the available commands).

MenuActive is the flag that decides if the menuitem should be shown.

Country is the country code. This has to be the same as DefaultLanguage in config.xml.

### Example 13 - DeviceRequest (GetMenu)

```
<?xml version="1.0" standalone="yes"?>
<DeviceRequest RequestType="GetMenu"PosID="Kasse 1" RequestID="1" SequenceID="6">
  <Output OutDeviceTarget="CashierDisplay">
    <MenuHeader>Application Selection</MenuHeader>
    <TextLine Command="1">MasterCard</TextLine>
    <TextLine Command="2">Maestro</TextLine>
  </Output>
</DeviceRequest>
```

This DeviceRequest is different in the sense that it includes both an Output and an Input section. In the example above the EPS application sends a menu to the POS with at headline (Application Selection) and to menu items.

The value in the response equals the command from the request.

### 9.4.12 DeviceResponse (GetMenu)

#### Example 14 - DeviceResponse (GetMenu)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceResponse RequestType="GetMenu" PosID="Kasse 1" RequestID="1" SequenceID="6"
OverallResult="Success">
  <Input>
    <Value>1</Value>
  </Input>
</DeviceResponse>
```

The value in the response equals the command from the request.

### 9.4.13 DeviceRequest (AdviceFlag)

This device request is optional. It is controlled via the config.xml file. See section 9.9

#### Example 15 - DeviceRequest (AdviceFlag)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceRequest RequestType="AdviceFlag" PosID="POS3" RequestID="12" SequenceID="14">
  <Output OutDeviceTarget="CashierDisplay">
    <TextLine>Please perform end-of-day routine.</TextLine>
  </Output>
</DeviceRequest>
```

### 9.4.14 DeviceResponse (AdviceFlag)

POS must answer with the DeviceResponse if the request is sent. The Value is "Y" or "No".

#### Example 16 - DeviceResponse (AdviceFlag)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DeviceResponse RequestType="AdviceFlag" PosID="POS3" RequestID="12" SequenceID="14"
OverallResult="Success">
  <Input>
    <Value>Y</Value>
  </Input>
</DeviceResponse>
```

## 9.5 ServiceRequest and ServiceResponse

ServiceRequests can be described as calling more administrative functions - this being reconciliation, retrieving capture (Journal) data, printing of last receipt etc.

### 9.5.1 ServiceRequest (Reconciliation)

ServiceRequest (Reconciliation) finishes all transactions – including any off-line transactions. Reconciliation can only be done when the terminal is on-line.

#### Example 16 - ServiceRequest (Reconciliation)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="Reconciliation" PosID="Kasse 1" RequestID="2">
</ServiceRequest>
```

### 9.5.2 ServiceResponse (Reconciliation)

#### Example 17 - ServiceResponse (Reconciliation)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="Reconciliation" PosID="Kasse 1" RequestID="2" SubCode=""
OverallResult="Success">
</ServiceResponse>
```

OverallResult indicates success or failure for the ServiceRequest as a whole.

If OverallResult is Failure and if capture logic is implemented, then there might be pending unfinished transactions. In this case reconciliation must be attempted until OverallResult shows "success".

In case of failure the SubCode can contain a more elaborate error code - if any.

### 9.5.3 ServiceRequest (Diagnosis)

Diagnosis ServiceRequest is intended as a tool to return diagnostic data – either to POS display or printer. This tool is used in case of problems with communication or the like. Typically a diagnostic report is returned showing the current system status. The precise nature of this report is subject to the specific solution.

#### Example 21 - ServiceRequest (Diagnosis)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="Diagnosis" PosID="Kasse 1" RequestID="3">
</ServiceRequest>
```

### 9.5.4 ServiceResponse (Diagnosis)

#### Example 22 - ServiceResponse (Diagnosis)



```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="Diagnosis" PosID="Kasse1" RequestID="2" OverallResult="Success"
SubCode="" Asw1Asw2="">
  <SystemState>CONNECT_OK</SystemState>
  <Error>0</Error>
  <TerminalConnectionState>OPEN</TerminalConnectionState>
</ServiceResponse>
```

The Error value is always an integer.

Possible TerminalConnectionState values:

DISCONNECTED

CONNECTED

OPEN

OPEN\_NP

Possible SystemState values:

CONNECT\_OK

CONNECT\_NO\_INIFILE

CONNECT\_INIFILE\_READ\_ERROR

CONNECT\_COMPORTINITFAILURE

CONNECT\_SW\_NOTCOMPATIBLE

CONNECT\_TERM\_NOT\_RESP

CONNECT\_COMPORTOPEN

DATA\_LINK\_ERROR

FUNCTION\_NOT\_POSSIBLE

TIMEOUT\_IN\_COMMUNICATION

CONNECT\_NO\_LICENCE

CONNECT\_INTERNAL\_ERROR

CONNECT\_SYSTEM\_ERROR

OPEN\_NO\_RECEIPT

CONNECT\_INIT\_VALUE

### 9.5.5 ServiceRequest (Login)

If required the login ServiceRequest is used to set the terminal in a ready state, and the terminal displays the following text: "Terminal Ready". This can be done at POS application start time, or when a sales clerk logs into the system.

#### Example 23 - ServiceRequest (Login)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="Login" PosID="Kasse 1" RequestID="1">
</ServiceRequest>
```

### 9.5.6 ServiceResponse (Login)

#### Example 24 – ServiceResponse (Login)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="Login" PosID="Kasse 1" RequestID="1" SubCode=""
OverallResult="Success">
</ServiceResponse>
```

### 9.5.7 ServiceRequest (Logoff)

The Logoff ServiceRequest will set the terminal into a "Closed" state.

#### Example 25 – ServiceRequest (Logoff)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="Logoff" PosID="Kasse 1" RequestID="111">
</ServiceRequest>
```

### 9.5.8 ServiceResponse (Logoff)

#### Example 26 – ServiceResponse (Logoff)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="Logoff" WorkstationID="Kasse 1" RequestID="111" SubCode=""
OverallResult="Success">
</ServiceResponse>
```

### 9.5.9 ServiceRequest (AdminFunction)

Another option is to implement a generic ServiceRequest to call specific administrative functions. This requires a list of predined functions corresponding with different functionalities in the terminal.

#### Example 27 – ServiceRequest - AdminFunction

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="AdminFunction" PosID="Kasse 1" RequestID="12">
  <AdminCommand>34</AdminCommand>
  <Parameter1>10.0.0.131</Parameter1>
  <Parameter2>255.255.0.0</Parameter2>
  <Parameter3>10.0.10.1</Parameter3>
  <Parameter4>10.0.10.206</Parameter4>
  <Parameter5>10.0.10.10</Parameter5>
  <Parameter6>dummy</Parameter6>
</ServiceRequest>
```

Supported commands is listed in table 9.7 - Administration command table.

During the various administrative functions will occur "DeviceRequest" in the exact same manor as in the other CardRequests and ServiceRequests.

### 9.5.10 ServiceResponse (AdminFunction)

In this example the ServiceRequest AdminFunction with AdminCommand value 2 is setting the terminal time towards the acquirer host. In the ServiceResponse is included a HostTimeStamp.

See Example 28

### Example 28 – ServiceResponse (AdminFunction(2))

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType=" AdminFunction " PosID="Kasse 1" RequestID="12" SubCode=""
OverallResult="Success">
  <HostTimeStamp>2009-11-03T14:16:26-01:00</HostTimeStamp>
</ServiceResponse>
```

#### 9.5.11 ServiceRequest (SignatureOnOff)

This ServiceRequest will swap the Signature state in PPI. It gives the option the run all transactions signature based (On).

**On:** All transactions are signature based transactions.

ForcedCVM in CardRequest is overruled.

**Off:** PSAM decides CVM if it is not defined in CardRequest.

The state is switched by sending the call again.

The signature state will overrules the ForcedCVM method set in the CardRequest.

If there is not specified ForcedCVM in the CardRequest and the SignatureOnOff is Off then the terminal itself will decide which mode to choose.

### Example 29 – ServiceRequest (SignatureOnOff)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="SignatureOnOff" PosID="Kasse 1" RequestID="32">
</ServiceRequest>
```

#### 9.5.12 ServiceResponse (SignatureOnOff)

### Example 30 – ServiceResponse (SignatureOnOff)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="OfflineOnOff" PosID="Kasse 1" RequestID="32" SubCode=""
OverallResult="Success">
  <CurrentStatus>On</CurrentStatus>
</ServiceResponse>
```

CurrentStatus can be On or Off.

#### 9.5.13 ServiceRequest (OfflineOnOff)

This method gives the option to run all transactions offline.

**On:** All transactions are offline transactions.

ForcedCapability in CardRequest is overruled.

**Off:** PSAM decides online/offline if it is not defined in CardRequest.

The state is switched by sending the call again.

If there is not specified ForcedCapability in the CardRequest and the OfflineOnOff is Off then the terminal itself will decide which mode to choose.

### Example 31 – ServiceRequest (OfflineOnOff)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="OfflineOnOff" PosID="Kasse 1" RequestID="35">
</ServiceRequest>
```

## 9.5.14 ServiceResponse (OfflineOnOff)

### Example 32 – ServiceResponse (OfflineOnOff)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="OfflineOnOff" PosID="Kasse 1" RequestID="35" SubCode=""
OverallResult="Success">
  <CurrentStatus>Offline</CurrentStatus>
</ServiceResponse>
```

CurrentStatus can be On or Off.

## 9.5.15 ServiceRequest (AdminMenu)

It's a mandatory requirement that the POS solution support some basic administrative functions in the terminal. Like retrieving last receipt, retrieve a terminal report and so forth. One of doing this is to do a ServiceRequest with the command type "AdminMenu"

### Example 33 – ServiceRequest - AdminMenu

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="AdminMenu" PosID="Kasse 1" RequestID="49">
</ServiceRequest>
```

In response to the the Request the DeviceClient returns a GetMenu DeviceRequest like the one explained in section 9.4.11. See example 34 below.

### Example 34 - DeviceRequest (GetMenu)

```
<?xml version="1.0" standalone="yes"?>
<DeviceRequest RequestType="GetMenu" PosID="Kasse 1" RequestID="49" SequenceID="3">
  <Output OutDeviceTarget="CashierDisplay">
    <MenuHeader>Select admin function</MenuHeader>
    <TextLine>Do diagnosis</TextLine>
    <TextLine>Retrieve version number</TextLine>
    <TextLine>Set clock </TextLine>
    <TextLine>Do reconciliation</TextLine>
    <TextLine>Get last receipt</TextLine>
  </Output>
</DeviceRequest>
```

In example 35 is shown a reply example – in this case to get last receipt from the terminal.

### Example 35 - DeviceResponse (GetMenu)

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<RequestType="GetMenu" PosID="Kasse 1" RequestID="49" SequenceID ="3" OverallResult="Success">
  <Input>
    <Value>4</Value>
  </Input>
</DeviceResponse>
```

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<RequestType="GetMenu" PosID="Kasse 1" RequestID="49" SequenceID ="3" OverallResult="Success">
  <Input>
    <Value>34</Value>
    <Parameter1>10.0.0.131</Parameter1>
    <Parameter2>255.255.0.0</Parameter2>
    <Parameter3>10.0.10.1</Parameter3>
    <Parameter4>10.0.10.206</Parameter4>
    <Parameter5>10.0.10.10</Parameter5>
    <Parameter6>dummy</Parameter6>
  </Input>
</DeviceResponse>
```

**Table 9.7:** Administration command table PSAM

Command (Value)	Description	Flexdriver call	Supported	Parameter
0	Default TCP response	None	Yes	No
1	Endofday (balancing/clearing) routine	ADMIN_ENDOFDAY	Yes	No
2	Endofday routing	ADMIN_ENDOFDAYLOG	Yes	No
3	Terminal report	ADMIN_REPORT_TERMINALREPORT	Yes	No
4	Transaction total report	ADMIN_REPORT_TOTALS	Yes	No
5	Transaction log report	ADMIN_REPORT_LOG	Yes	No
6	Old transaction log report	ADMIN_REPORT_OLDLOG	Yes	No
7	Get last receipt	ADMIN_LASTRECEIPT	Yes	No
8	Unlock receipt - Deprecated	ADMIN_UNLOCK_RECEIPT	Yes	No
9	Sync the clock against PBS/Nets	ADMIN_CLOCKSYNCPBS	Yes	No
10	Sync the clock against Point/Verifone	ADMIN_CLOCKSYNCPPOINT	Yes	No
11	Send log to Verifone	ADMIN_SENDLOG	Yes	No

12	Clear datastore	ADMIN_CLEARDATASTORE	Yes	No
13	Download program	ADMIN_DOWNLOADPROGRAM	Yes	No
14	Download param	ADMIN_DOWNLOADPARAM	Yes	No
15	Download PAN	ADMIN_DOWNLOADPAN	Yes	No
16	Download TLCMDB	ADMIN_DOWNLOADTLCMDB	Yes	No
17	Restore TLCMDB	ADMIN_RESTORETLCMDB	Yes	No
18	Terminal contrast up	ADMIN_CONTRASTUP	Yes	No
19	Terminal contrast down	ADMIN_CONTRASTDOWN	Yes	No
20	Restart terminal	ADMIN_RESTARTTERMINAL	Yes	No
21	Force an eject of the card	ADMIN_EJECTCARD	Yes	No
22	Retrieve card range table	ADMIN_MSC	Yes	No
23	Terminal backlight on	ADMIN_BACKLIGHT_ON	Yes	No
24	Terminal backlight off	ADMIN_BACKLIGHT_OFF	Yes	No
25	Network report	ADMIN_NETWORK_REPORT	Yes	No
26	DCC rates report	ADMIN_RATES_REPORT	Yes	No
27	Gratuity receipt	ADMIN GRATUITY_RECEIPT	Yes	No
28	Remove/delete defect advice file	ADMIN_EXCLUDE_DATASTORE_RECORD_WITH_STAN	Yes	1: STAN 2: File ID 3: File type 4: PSAM sub (slot)
29	Emptying of the terminals transactions database against the host (Nets)	ADMIN_ADVICEFORWARDING	Yes	No
30	File 5 status report	ADMIN_REPORT_FILE5STATUS	Yes	No
31	Deprecated	ADMIN_RESERVED_FOR_OCX	Yes	No
32	Report PCT	ADMIN_REPORT_PCT	Yes	No
33	Download DCC rates	ADMIN_DOWNLOADDCCRATES	Yes	No
34	Update PSAM	ADMIN_UPDATEPSAM	Yes	No
35	Update fee table	ADMIN_UPDATEFEETABLE	Yes	No

36	Update salt value for Ekvittering	ADMIN_UPDATESALT	Yes	No
37	Reconciliation with parameter	ADMIN_GETADVICERECON	Yes	1: End-of-days
38	Set batch number	ADMIN_SET_BATCH_NUMBERS	Yes	1: Currency 2: Batch Number
39	TCS report	ADMIN_REPORT_TCS	Yes	No
40	Get and save the terminal properties used in the FlexDriver to control Extended Issuer Envelope (EIE).	ADMIN_REPORT_TPROPS_CVS	Yes	No
41	Print the PCI eventlog report	ADMIN_EVENTLOG_PRINT	Yes	No
42	Send the PCI eventlog to a syslog server, with IP address that you provide.	ADMIN_EVENTLOG_SEND	Yes	1: IP
43	Delete the PCI eventlog	ADMIN_EVENTLOG_DELETE	Yes	No
44	Get IP settings	ADMIN_GETIP_SETTING	Yes	No
45	Set IP settings	ADMIN_SETIP_SETTING	Yes	1: IP/ DHCP 2: Subnet 3: Gateway 4: DNS1 5: DNS2 6: Domain
46	Download images to the terminal	ADMIN_DOWNLOAD_IMAGES	No	No
47	Check if the card is inserted in the card reader	ADMIN_CHECK_CARD	No	No

**Table 9.9:** Administration command table FOS5

Command (Value)	Description	Supported	Parameter
0	Default TCP response	Yes	No
3030	Print stored report, if any		
3031	Test communication to host		

3032	Start copy of last reconciliation		
3130	Start reconciliation		
3135	Start balance inquiry for a card		
3136	Print X-report		
3137	Print Z-report		
3138	Start sending offline transactions to host, if any		
3139	Print turnover report		
3830	Start copy of last receipt		
3831	Print errorlog		
3832	Print parameter list		
3833	Print program information		
3834	Print active cards on terminal		
3835	Print stored offline transaction information		
3836	Print communication status		
3837	Print ECR status		
3930	Send logs		
3931	Start software loading		
3932	Connect to helpdesk		
3933	Trans status, only after advice		
3934	Card status, only unattended		
3935	Prepare purchase transaction, requires special software in terminal		
3936	Prepare other transaction, requires special software in terminal		
3937	End transaction, requires special software in terminal		
3938	Start sending of paypoint.log		

### 9.5.16 ServiceResponse (AdminMenu)

It's a mandatory requirement that the POS solution support some basic administrative functions in the terminal. Like retrieving last receipt, retrieve a terminal report and so forth.

#### Example 36 – ServiceResponse - AdminMenu

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="AdminMenu" RequestID="49" SubCode="" OverallResult="Success">
</ServiceResponse>
```

### 9.5.17 ServiceRequest (GetPreviousResult)

If an TransactionsID is provided at initiation of an transaction, then the system can provide a copy of the receipts and the results - if data exists. The intention is to provide an option to retrieve the data after a kind of breakdown, and in the same time it's possible to retrieve all transactions from



the current date. Older transaction data is deleted.

The data returned will be as display messages (DeviceRequest Display) and receipts (DeviceRequest Print) The CardResponse data will come in the format of a CardResponse - but wrapped as a service response (see example).

### Example 37 – ServiceRequest - GetPreviousResult

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="GetPreviousResult" PosID="Kasse 1" TransactionID="201308131001"
RequestID="49">
</ServiceRequest>
```

### 9.5.18 ServiceResponse (GetPreviousResult)

If no data is present for the requested TransactionID then the OverallResult would be "NoData".

### Example 38 – ServiceResponse - GetPreviousResult

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="GetPreviousResult" PosID="Kasse1" RequestID="21" TransactionID="999"
OverallResult="Success" SubCode="" Asw1Asw2="">
  <ResponseDetails>
    <CardResponse RequestType="Payment" PosID="Kasse1" RequestID="4" TransactionID="999"
OverallResult="Success" SubCode="0" SubCodeMessage="" Asw1Asw2="0">
      <TerminalID>990790</TerminalID>
      <RefNo>4065516056</RefNo>
      <PSAM>104855</PSAM>
      <Merchant>0001978543</Merchant>
      <TotalAmount>8384</TotalAmount>
      <TransactionAmount>8384</TransactionAmount>
      <TransactionCurrency>DKK</TransactionCurrency>
      <FeeAmount>0</FeeAmount>
      <VatAmount>0</VatAmount>
      <CashbackAmount>0</CashbackAmount>
      <AcquirerID></AcquirerID>
      <CardPAN>541333AAAAAA1612</CardPAN>
      <CardCircuit>3</CardCircuit>
      <ApprovalCode>0</ApprovalCode>
      <TCC>DA1</TCC>
      <EReciptToken></EReciptToken>
      <Stan>2189</Stan>
    </CardResponse>
  </ResponseDetails>
</ServiceResponse>
```

## 9.6 Configuration and deployment

Point Payment Interface (PPI) is a integration interface and an application. The interface is defined in the previous sections. The application behind the interface and the configuration of the interface is the subject of this section.

The PPI application can be used in single instance mode on as workstation (POS), and in this case the application will run as service under "local system" account. This is the topic of section 9.6.1 - Single instance PPI.

If required the application can be used in server mode, and in this case the application can handle a number of POS → Terminal relations. The application will also here run as a service.

The overall structure and logic in PPI is the same for the 2 types of execution modes, but there are differences when it comes to initialization and termination of POS → Terminal relations.

### 9.6.1 Single instance mode

If the setting Server is set to False in the Config.xml file, then the application will start as in "single instance mode". In this case the application will read the configuration parameters regarding integration type and other integration parameters and will attempt to connect to the terminal.

If the setting AutoLogin is set to true, then the application will do a Open() request to the terminal, and the terminal will end in "Terminal Ready" state. Otherwise the terminal will be in "Welcome" state until a Login() request is initiated from the POS system.

After entering "Terminal Ready" state the POS can do transactions and administrative request until Logout() request is initiated and the terminal returns to the "Welcome" state.

Which IP-addresses and port numbers used by the application for Card-, Service or Device request is defined in the Config.XML file.

The different settings in the Config.xml file is explained in further detail in section 9.6.3.

### 9.6.2 Server mode

If the setting Standalone is set to No(N) in the Config.xml file, then the application will start as in "Server instance mode". In this case the application will always initiate a server instance on port 20000 and wait for a POS to initiate an POS → Terminal relation.

TerminalPort is optional.

#### Example 39 – ServiceRequest - GetInitializedTerminal()

```
<?xml version="1.0" encoding="ISO-8859-15"?>
```

```
<InitTermRequest RequestType="GetInitializedTerminal" PosID=" Kasse123456">
  <IPAddress>127.0.0.1</IPAddress>
  <ComPortOrIPAddress>10.0.0.6</ComPortOrIPAddress>
  <TerminalNumber>00990790</TerminalNumber>
  <TerminalPort>2000</TerminalPort>
</InitTermRequest>
```

#### Example 40 – ServiceResponse - GetInitializedTerminal()

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType=" GetInitializedTerminal " PosID="Kasse123456" OverallResult="Success">
  <RequestPort>20001</RequestPort>
  <DevicePort>20002</DevicePort>
</ServiceResponse>
```

### 9.6.3 Reload config file

There are different use cases of which it is relevant to be able to reload the configuration file. This can be done using the following request.

#### Example – ServiceRequest - ReloadConfig

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<InitTermRequest RequestType="ReloadConfig" PosID="Kasse1"></InitTermRequest>
```

#### Example – ServiceResponse - ReloadConfig

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<InitTermResponse RequestType="ReloadConfig" PosID=" Kasse1"
OverallResult="Success"></InitTermResponse>
```

## 9.7 Running PPI as a Windows Service

To run PPI as a service the PPIService application is used.

### 9.7.1 Before installation of PPIService

In the folder:

- C:\PPI-EPS

the following files have to be placed:

Files:

- AdminFunctions.xml
- config.xml
- flxdrv.dll
- InstallPPIService.bat
- PPI.exe
- PPIService.exe
- UninstallPPIService.bat

### 9.7.2 Install PPIService

Guide:

1. Open cmd.exe as Administrator
2. Go to C:\PPI-EPS
3. Run InstallPPIService.bat

PPIServices will not start automatically after a restart.

Instead of restarting your computer then open the (Windows) Services application. Find PPIServices and start it manually.

### 9.7.3 Uninstall PPIService

Guide:

1. Open cmd.exe as Administrator
2. Go to C:\PPI-EPS
3. Run UninstallPPIService.bat

If the output from the bat file says something different from:

The uninstall has completed.

Then close the (Windows) Services application and try uninstalling again. Otherwise restart Windows and try uninstalling again.

## 9.8 Default error messages

Error messages comes in the format:

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<Message>Invalid XML request</Message>
```

All requests to PPI can result in an error message.

Message	Description
Invalid XML request	The XML send to PPI is invalid in the sense that it either does not contain a valid XML header or it does not have a valid root element. At the moment nothing else is checked.

## 9.9 Configuration (config.xml)

The config.xml file contains the config info.

The meaning of each element is described in the following table:

Element name	Description	Value description
IntegrationType	Decides the integration type	"PSAM" or "FOS5"
IPAddress	The IP address that PPI listens on for new terminal initialization. Only relevant when Standalone = N	IP address
DataPortInit	The IP port that PPI listens on for new terminal initialization. Only relevant when Standalone = N	IP Port
DefaultLanguage	Describes the default language	2 character language code
Standalone	Describes if PPI should only handle one terminal and one POS (Y) or PPI should handle several POS and terminals (N).	"Y" or "N"
NetworkTestOnly	Mode to test network without a physical terminal. Only login/logoff and transactions are supported. With display messages and prints.	"Y" or "N"

SendAdviceFlag	Decides if a AdviceFlag request should be sent from PPI.	"Y" or "N"
SendGetCardConfirmation	Decides if a GetCardConfirmation request should be sent from PPI.	"Y" or "N"
DailyLog	Log events on daily basis.	"Y" or "N"
LogCleanUpAfterDays	Delete log after specified number of days.	Number of days
EnableSNMP	Enable SNMP traps if terminal is disconnected. Currently only supported in standalone mode.	"Y" or "N"
AskForAuthCode	This decides if the merchant should type in a authorization code (Y) or the default authorization code should be used.	"Y" or "N"
TerminalTraceLevel	This sets the terminal trace level. Default is 2.	PSAM:  1 = FLX_CONF_TRACE 2 = FLX_CONF_EXTTRACE 3 = FLX_CONF_EXTTRACE_PLUS
TimeOutSettings → Request-Port → ReceiveTimeout	Describes how long PPI should wait receiving data before a timeout i made in receive mode.	Timeout in ms. 0 is infinite.
TimeOutSettings → Request-Port → SendTimeout	Describes how long PPI should wait trying to send data before a timeout i made.	Timeout in ms. 0 is infinite.
TimeOutSettings → Device-Port → ReceiveTimeout	Describes how long PPI should wait receiving data before a timeout i made in receive mode.	Timeout in ms. 0 is infinite.
TimeOutSettings → Device-Port → SendTimeout	Describes how long PPI should wait trying to send data before a timeout i made.	Timeout in ms. 0 is infinite.
CardResponseSettings → SubCodeInCardResponse	Sub code in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → Asw1Asw2InCardResponse	Asw1Asw2 code in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → TerminalIDInCardResponse	Terminal ID in CardResponse. Yes or No.	"Y" or "N"

CardResponseSettings → PSAMInCardResponse	PSAM number in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → MerchantInCardResponse	Merchant ID in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → FeeAmountInCardResponse	Fee amount in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → VatAmountInCardResponse	Vat amount in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → CashBackAmountInCardResponse	Cashback amount in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → AcquirerIDInCardResponse	Acquirer ID in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → CardCircuitInCardResponse	Card Circuit in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → ApprovalCodeInCardResponse	Approval code in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → TCCInCardResponse	TCC in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → EReceiptTokenInCardResponse	E receipt token in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → StanInCardResponse	Stan in CardResponse. Yes or No.	"Y" or "N"
CardResponseSettings → TokenDataInCardResponse	TokenData in CardResponse. Yes or No.	"Y" or "N"
SNMPSettings → SNMPIP	IP address of the SNMP manager	IP address
SNMPSettings → SNMPOid	Object ID for the the given POS. Value of traps is "Terminal disconnected from {PosID}"	Valid Oid, eg. 1.6.1.1.8.5.2.1.0
StandaloneProperties → AutoLogin	Descides whether a Login request should be sent before terminal is connected and opened.	"Y" or "N"
StandaloneProperties → PosID	Descides the communication type	"IP" or "RS232"

StandaloneProperties ComPortOrIPAddress →	If the ComType is IP then this describes the terminals IP address. If the ComType is RS232 then this describes the COM port.	IP address or COM port
StandaloneProperties BAUDRATEorTCPIPPORT →	If the ComType is IP then this is the port the terminal is listening on. If the ComType is RS232 then this describes the BAUD rate	IP port or BAUD rate. Consult Verifone for supported BAUD rates.
StandaloneProperties → RequestPort	Port for sending CardRequest/ServiceRequests	IP port
StandaloneProperties → DevicePort	Port for receiving DeviceRequests	IP port
ClientServerProperties TalkToLocalhost →	Talk to 127.0.0.1 or the machines IP address	"Y" or "N"
ClientServerProperties ReuseInitializedTerminals →	Option to reuse the flexdriver instance for the given terminal	"Y" or "N"
ClientServerProperties DynamicPortAllocationStart →	If the POSId isn't present under "FixedPOSSystems", then it will start the port allocation from this value	Port number
ClientServerProperties FixedPOSSystems →	List of POS systems with fixed tcp ports	Ex:  <POSSystem> <POSID>Kasse1</POSID> <RequestPort>20001</RequestPort> <DevicePort>20002</DevicePort> </POSSystem>

#### Example of the config.xml file:

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<config>
  <IntegrationType>PSAM</IntegrationType>
  <IPAddress>127.0.0.1</IPAddress>
  <DataPortInit>20000</DataPortInit>
  <DefaultLanguage>DK</DefaultLanguage>
  <Standalone>Y</Standalone>
  <NetworkTestOnly>N</NetworkTestOnly>
  <SendAdviceFlag>Y</SendAdviceFlag>
  <SendGetCardConfirmation>Y</SendGetCardConfirmation>
  <DailyLog>Y</DailyLog>
```



```

<LogCleanUpAfterDays>5</LogCleanUpAfterDays>
<EnableSNMP>Y</EnableSNMP>
<AskForAuthCode>Y</AskForAuthCode>
<TerminalTraceLevel>2</TerminalTraceLevel>
<TimeOutSettings>
  <RequestPort>
    <ReceiveTimeout>15000</ReceiveTimeout>
    <SendTimeout>10000</SendTimeout>
  </RequestPort>
  <DevicePort>
    <ReceiveTimeout>15000</ReceiveTimeout>
    <SendTimeout>10000</SendTimeout>
  </DevicePort>
</TimeOutSettings>
<CardResponseSettings>
  <SubCodeInCardResponse>Y</SubCodeInCardResponse>
  <Asw1Asw2InCardResponse>Y</Asw1Asw2InCardResponse>
  <TerminalIDInCardResponse>Y</TerminalIDInCardResponse>
  <PSAMInCardResponse>Y</PSAMInCardResponse>
  <MerchantInCardResponse>Y</MerchantInCardResponse>
  <FeeAmountInCardResponse>Y</FeeAmountInCardResponse>
  <VatAmountInCardResponse>Y</VatAmountInCardResponse>
  <CashBackAmountInCardResponse>Y</CashBackAmountInCardResponse>
  <AcquirerIDInCardResponse>Y</AcquirerIDInCardResponse>
  <CardCircuitInCardResponse>Y</CardCircuitInCardResponse>
  <ApprovalCodeInCardResponse>Y</ApprovalCodeInCardResponse>
  <TCCInCardResponse>Y</TCCInCardResponse>
  <EReciptTokenInCardResponse>Y</EReciptTokenInCardResponse>
  <StanInCardResponse>Y</StanInCardResponse>
  <TokenDataInCardResponse>Y</TokenDataInCardResponse>
</CardResponseSettings>
<SNMPSettings>
  <SNMPIP>10.0.0.5</SNMPIP>
  <SNMPOid>1.6.1.1.8.5.2.1.0</SNMPOid>
</SNMPSettings>
<StandaloneProperties>
  <AutoLogin>N</AutoLogin>
  <PosID>Kasse1234</PosID>
  <ComType>IP</ComType>
  <ComPortOrIPAddress>10.0.0.131</ComPortOrIPAddress>
  <BAUDRATEorTCPIPSPORT>2000</BAUDRATEorTCPIPSPORT>
  <RequestPort>20001</RequestPort>
  <DevicePort>20002</DevicePort>
</StandaloneProperties>
<ClientServerProperties>
  <TalkToLocalhost>N</TalkToLocalhost>
  <ReuseInitializedTerminals>Y</ReuseInitializedTerminals>
  <DynamicPortAllocationStart>20001</DynamicPortAllocationStart>
  <FixedPOSSystems>
    <POSSystem>
      <POSID>kasse1</POSID>
      <RequestPort>20009</RequestPort>
      <DevicePort>20010</DevicePort>
    </POSSystem>
  </FixedPOSSystems>
</ClientServerProperties>

```

```

    <POSSystem>
      <POSID>Kasse2</POSID>
      <RequestPort>20011</RequestPort>
      <DevicePort>20012</DevicePort>
    </POSSystem>
  </ClientServerProperties>
</config>

```

## 9.10 Register DLL (FOS5)

Before being able to use a FOS5 terminal the paypointAPI.dll has to be registered on the machine that runs PPI.

On windows 32 bit the path to the regsvr32 application is:

```
%systemroot%\System32\
```

On Windows 7 64-bit the 32bit folder is:

```
%systemroot%\SysWoW64\
```

Open a command prompt as administrator and go to the System32 folder. Then run the command:

```
regsvr32 C:\PPI-EPS\ paypointAPI.dll
```

## 9.11 Update Logic

This section describes the update communication between PPI and PPIService.

This option can only be used in standalone setups for now. The request is sent to the PPI admin port (19999) as a service request.

The flow is:

1. A client sends a SetUpdateFlag request to PPI.
2. PPIService checks for updates using the GetUpdateFlag request once every hour.
3. If the update flag is set to "Y" then PPIService uses the GetUpdateFile request to retrieve the new PPI.exe file.
4. The client can test whether the update has happened with the GetUpdateFlag request where the DoUpdate flag is reset to "N" when the update has happened.

### ServiceRequest - SetUpdateFlag()

PPI uses IntegrationType from config.xml to decide if the DLLFile is flxdrv.dll for PSAM terminals or paypointAPI.dll for FOS5 terminals.

```

<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="SetUpdateFlag" PosID="Kasse123456">
  <DoUpdate>Y</DoUpdate>

```

```

<PPIFile></PPIFile>
<PPIHash></PPIHash>
<DLLFile></DLLFile>
<DLLHash></DLLHash>
<UpdateIntervalStart>2013-10-10 08:00:00</UpdateIntervalStart>
<UpdateIntervalEnd>2013-10-10 18:00:00</UpdateIntervalEnd>
</ServiceRequest>

```

Element	Description	Data
DoUpdate	Describes whether an update should be done or not. Can be used to cancel updates by sending "N"	"Y" or "N"
PPIFile	Base64 encoded string of bytes read from the PPI.exe file.	Base64 encoded string. Empty if no update.
PPIHash	Hash (X2) of the PPIFile byte string.	Hash (String). Empty if no update.
DLLFile	Base64 encoded string of bytes read from the DLL file (flxdrv.dll or paypointAPI.dll)	Base64 encoded string. Empty if no update.
DLLHash	Hash (X2) of the DLL byte string.	Hash (String). Empty if no update.
UpdateIntervalStart	Time when the update interval should start.	DateTime format.
UpdateIntervalEnd	Time when the update interval should end.	DateTime format.

When sending DoUpdate: N all other elements are irrelevant.

### ServiceResponse - SetUpdateFlag()

```

<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType=" SetUpdateFlag " PosID="Kasse123456" OverallResult="Success">
</ServiceResponse>

```

To see pending updates:

### ServiceRequest - GetUpdateFlag()

```

<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="GetUpdateFlag" PosID="Kasse123456">
</ServiceRequest>

```

### ServiceResponse - GetUpdateFlag()

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="GetUpdateFlag" PosID="Kasse123456">
  <DoUpdate>Y</DoUpdate>
  <UpdateIntervalStart>2013-10-10 08:00:00</UpdateIntervalStart>
  <UpdateIntervalEnd>2013-10-10 18:00:00</UpdateIntervalEnd>
</ServiceRequest>
```

To download the new PPI.exe file:

### ServiceRequest - GetUpdateFile()

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceRequest RequestType="GetUpdateFile" PosID="Kasse123456">
</ServiceRequest>
```

### ServiceResponse - GetUpdateFile()

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<ServiceResponse RequestType="GetUpdateFile" PosID="Kasse123456">
  <PPIFile></PPIFile>
  <PPIHash></PPIHash >
  <DLLFile></DLLFile>
  <DLLHash></DLLHash>
</ServiceRequest>
```

## 9.12 Log files

When using PPI, a number of log files is used to log events. This is usefull in case of an error happening.

PPI logs:

C:\PPI-EPS\log folder

For each day a new log in made in the C:\PPI-EPS\log\YYYY-MM-DD folder. The logs contains all XML communication and relevant values for PPI.

The logging can be turned on and off using the config file (DailyLog).

Logs can be deleted automatically by defining the number of days a log should be kept fore using the config file. Recommended number is 5 days.

PPIAdmin log:

C:\PPI-EPS\log\PPIAdminLog.txt

This log contains all XML communication and relevant values for PPIAdmin.  
The log is deleted automatically when the file size reaches 20 MB.

PPIService log:

C:\PPI-EPS\log\PPIServiceLog.txt

This log contains all XML communication and relevant values for PPIService.  
The log is deleted automatically when the file size reaches 20 MB.

POSTestClient.log:

C:\PPI-EPS\log\POSTestClientLog.txt

This log contains all XML communication and relevant values for POSTestClient.  
The log is deleted automatically when the file size reaches 20 MB.

Terminal logs:

**PSAM:**

Point Flexdriver Trace file: C:\PPI-EPS\PtFxDrTr@XXXX.txt

FlexDriver Communication Trace file: C:\PPI-EPS\flxComTrace.txt

**FOS5**

PayPoint device log: C:\PPI-EPS\paypoint.log

PayPoint transaction log: C:\PPI-EPS\paypointtrans.log

## 10 | NFC

The NFC reader on the Yomani and Yomani 2 terminals is able to read both MIFARE Ultralight and MIFARE DESFire cards. The usage of both cards is – at the moment – limited to local cards and the customers own PAN range, i.e. PAN numbers starting with 9.

### 10.1 MIFARE Ultralight

To use a MIFARE Ultralight Verifone Denmark needs to read an ISO track 2 from the internal memory. In order to satisfy most customers, we support a number of different options on how we read the track data. The supported options are listed below.

1. We read a complete ISO track 2,
2. We read only a PAN followed by an '=' sign and append the cards unique id (UID).

The PAN must conform to the ISO/IEC 7812-1 standard, i.e. can be at most 19 digits long. Additionally the PAN must be at least 8 digits long so we can identify a customer correctly. In the case that you have bought a PAN-subrange from Verifone Denmark, the PAN must be at least 12 digits long.

All data must be written starting at page 4 on the chip. The first byte is reserved for the number of characters ( $n$ ) (written in hex) Verifone Denmark should read, followed by  $n$  characters. The track data must be written as binary-coded decimal (BCD), hence  $n$  should represent the number of nibbles, **not** the number of bytes. The PAN **must** always end with an '=' sign normally written as 0x3D (ASCII), this becomes 0xD in BCD format.

## Full track 2 example

Table 10.1 shows an example of Verifone Denmark 1.

Page	Data			
	0x00	0x01	0x02	0x03
0x04	0x1F	0x92	0x08	0x61
0x05	0x61	0x00	0x00	0x05
0x06	0x3D	0x99	0x12	0x70
0x07	0x10	0x00	0x00	0x04
0x08	0x9F	0xFF	0xFF	0xFF

**Table 10.1:** Example with track 2 “9208 6161 0000 053 = 9912 7010 0000 049”

Table 10.1 explained:

Page (0x04, 0x00)	0x1F = 31 characters
Page (0x04, 0x01) to (0x06, 0x00)	BCD encoded PAN ending with 0xD (=)
Page (0x06, 0x01) to (0x08, 0x00)	0xDF: expiration date, service code and discretionary data
Page (0x08, 0x00) to (0x08, 0x03)	0xDF: last digit of discretionary data followed by garbage

## Only PAN example

Table 10.2 shows an example of Verifone Denmark 2.

Page	Data			
	0x00	0x01	0x02	0x03
0x04	0x0D	0x92	0x08	0x61
0x05	0x61	0x99	0x99	0xDF

**Table 10.2:** Example with PAN sub-range “9208 6161 9999”

Table 10.2 explained:

Page (0x04, 0x00)	0x0D = 13 characters
Page (0x04, 0x01) to (0x05, 0x02)	BCD encoded PAN
Page (0x05, 0x03)	0xDF: ‘=’ followed by garbage

# 11 | Connection Service Provider

## 11.1 Internetbetingelser

### 11.1.1 Verifone Denmark's internetbetingelser

1. Internettet skal være tilgængeligt f.eks. via en router.
2. Terminalen leveres med Ethernetkort og 3m kabel (dette gælder ikke for Yomani terminaler), hvilket kræver et ledigt RJ45E Ethernet stik i netværket.
3. Terminalen er ikke designet til at køre direkte på internettet. Der skal derfor være en router/-firewall mellem internettet og terminalen.
4. Det net, som terminalen kobles til, skal kunne agere som DHCP server, idet terminalen 'spørger' efter netværksadresser, når den starter op.
5. Hvis punkt 4 ikke er mulig, kan terminalen køre med faste IP adresser. Verifone Denmark skal da have oplyst følgende i forbindelse med bestilling af terminalen: Domæne navn, DNS1, DNS2, IP-adresse, Subnet Maske og Default Gateway. Det er til enhver tid kundens ansvar, at disse oplysninger er korrekte.
6. Kommunikation til Nets og Verifone Denmark sker på følgende porte:  
13, 5214, 10760, 10960, 19000, 22000 og 24000, hvorfor disse skal være åbne fra terminalen og ud mod internettet plus for DNS-opslag på port 53 (UDP adresse).
7. Der skal åbnes for DNS forespørgsler på domænet point-ts.dk, bec.dk og danskebank.dk, dvs. følgende DNS forespørgsler skal kunne besvares af DNS-serverne. Sekundære opslag understøttes ikke.

pbs1.point-ts.dk port 19000

pbs2.point-ts.dk port 19000

param.point-ts.dk port 24000

time.point-ts.dk port 13

rtl.point-ts.dk port 5214

test.point-ts.dk port 22000

test2.point-ts.dk port 22000

dcc.point-ts.dk port 1003

ekvittering.point-ts.dk port 80

storebox.point-ts.dk port 80

lp.point-ts.dk port 10760

pplp.point-ts.dk port 10960

cbp.point-ts.dk port 9000

tnss.point-ts.dk port 443

mobilepaypos.danskebank.dk port 443

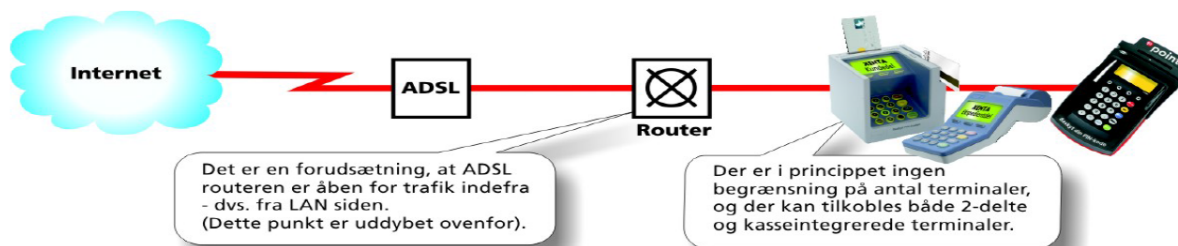
mobilepaypos2.danskebank.dk port 443

Disse domænenavne og IP-adresser kan blive ændret som følge af ny funktionalitet. Verifone Denmark tager ikke ansvaret for, at eventuelle opdateringer på kundens net gennem-



føres.

- Verifone Denmark anbefaler at lave en host(A) record med de IP-adresser, som terminalen bruger på jeres lokale DNS-server, hvis der findes regler i firewall omkring trafik til Verifone Denmark og Nets.
- Verifone Denmark anbefaler, at terminalen sidder i eget WLAN, således at terminalen beskyttes mod broadcasts og RIP request, da disse pakketyper kan påvirke terminalens svartider.



Gratis program (\*.exe fil) til test af forbindelsen fra kundens net til Verifone Denmark og Nets kan rekvireres fra Verifone Denmark.

**Verifone Denmark påtager sig intet ansvar for kundernes internet installation og/eller stabiliteten af denne, men kan dog anviser en samarbejdspartner, som kan være behjælpelig med installationen.**

## 11.2 Verifone Denmark's Internet Requirements

1. The Internet must be accessible, through a router, for example.
2. The terminal comes with an Ethernet card and a 3m cable (this does not apply for the Yomani terminal), which requires a free RJ45 Ethernet connector to the network.
3. The terminal is not designed to run directly on the Internet. Therefore, there must be a router/firewall between the Internet and the terminal.
4. The network the terminal is connected to must be able to act as a DHCP server, as the terminal "asks" for a network address when it starts up.
5. If item 4 is not possible, the terminal may function with a fixed IP address. Verifone Denmark must be notified of the following when ordering the terminal: Domain name, DNS1, DNS2, IP address, Subnet Mask and Default Gateway. It is your responsibility to ensure this information is correct.
6. Communication to Nets and Verifone Denmark is through the following ports: 13, 5214, 10760, 10960, 19000, 22000 and 24000, which means these must be open from the terminal and onto the Internet, plus DNS lookup on port 53 (UDP address).
7. DNS queries must be opened on the domain point-ts.dk, bec.dk and mobilepay.dk, i.e., the following DNS queries must be answered by the DNS servers: (Secondary lookup is not supported).

pbs1.point-ts.dk port 19000

pbs2.point-ts.dk port 19000

param.point-ts.dk port 24000

time.point-ts.dk port 13

rtl.point-ts.dk port 5214

test.point-ts.dk port 22000

test2.point-ts.dk port 22000

dcc.point-ts.dk port 1003

ekvittering.point-ts.dk port 80

storebox.point-ts.dk port 80

lp.point-ts.dk port 10760

pplp.point-ts.dk port 10960

cbp.point-ts.dk port 9000

tnss.point-ts.dk port 443

mobilepaypos.danskebank.dk port 443

mobilepaypos2.danskebank.dk port 443

These domain names and IP addresses can be changed as a result of new functionality.

Verifone Denmark accepts no responsibility for any updates implemented in the customer's network.

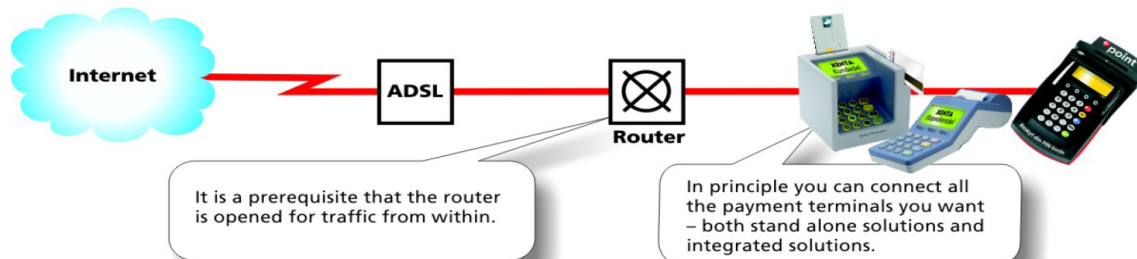
- Verifone Denmark recommends creating a host(A) record with the IP addresses the terminal uses on your local DNS server, if there are rules in the firewall concerning traffic to Verifone Denmark and Nets.
- Verifone Denmark recommends that the terminal is in its own WLAN, so that the terminal is protected from broadcasts and RIP requests, as these package types may affect the terminal response times.

Verifone Denmark A/S

Kundeservice, Knapholm 7, 2730 Herlev, Tlf.: 44 53 75 00, [kundeservice@verifone.com](mailto:kundeservice@verifone.com), CVR nr. 15 40

12 81

Se vilkår på [www.verifone.dk](http://www.verifone.dk)



A free program (\*.exe file) to test the connection from the customer's network to Verifone Denmark and Nets can be obtained from Verifone Denmark.

**Verifone Denmark accepts no responsibility for customers Internet installation and/or its stability, but may assign a partner who can help with installation.**

# 12 | Implementation Guide for Integrators

## 12.1 Objective

The objective of this guide is to describe all the steps needed in order to create an Electronic Cash Register Integration (Merchant Application) to a Verifone Denmark payment terminal. It will guide you to make the best choice of integration for your needs including how to fill in and sign the Development Agreement. Additionally it will provide an overview of the PCI rules and a general FAQ. Verifone Denmark will also provide technical manuals with examples for all the various integration techniques, which will offer an overview of the different approaches.

## 12.2 Completing the Integration – Step by Step

### Step 1 – Select an Integration Technique

Firstly you must choose which technique you will use to build your Merchant Application. Basically this choice is a question of how much you want to design and program yourself or how much you want to reuse the applications made by Verifone Denmark. The technical development guides in the appendixes will help you choose the right technique for your Merchant Application.

Here are the different techniques supported by Verifone Denmark:

1. PointTerminalDLL (UserControl): a fast way of integrating a POS running on Windows 32/64 bit (2003 to Windows 10).  
This DLL is compiled for .NET 4.0 to ensure compatibility with POSReady and ReadyPOS (Windows XP).
2. FlexDriver: a DLL which will take longer to develop but will provide you with more flexibility regarding your own layout of dialogs. Currently 32/64 bit Windows 2003 to Windows 10 and Linux supported.
3. Local Payment Protocol (LPP): for the most experienced programmer who is able to handle COM and TCP/IP communication, packet coding/decoding, dialogs and printers. This technique is mainly used by proprietary systems, ROM based or non Windows/Linux based POS.

Typical time consumption for each development process using the different techniques:

UserControl – development takes from 1 week to several weeks.

DLL – development takes from 1 month to several months.

LPP – development takes 6 months or more.

Please feel free to contact Verifone Denmark's development department at [development.hrv@verifone.com](mailto:development.hrv@verifone.com) if you have any questions regarding the different integration techniques.

### Step 2 – Signing the Development Agreement

When you have decided which integration technique you want to use, you will have to fill in and sign a Development Agreement. Please contact Verifone Denmark's sales department at

[sales.hrv@verifone.com](mailto:sales.hrv@verifone.com), and they will help you with the Development Agreement.

### **Step 3 – PCI DSS & PA DSS Requirements**

PCI DSS (Payment Card Industry Data Security Standard) is a set of global requirements created to ensure a high level of account data protection.

The standards are published on [www.pcisecuritystandards.org](http://www.pcisecuritystandards.org) – among other web pages.

Account Data (Cardholder Data and Sensitive Authentication Data) is sensitive and Verifone Denmark payment terminals do not store this data in the terminal, and furthermore all such data that is sent from the terminal is masked. Verifone Denmark payment terminals are PTS approved.

NB - If a Merchant Application allows input of Cardholder Data it will affect the need for PA DSS approval.

NB - If the Merchant Application is based on an existing solution that used to store Cardholder Data, the integrator must remove all critical data in order to be PCI DSS compliant.

The decision whether or not a Merchant Application requires a PCI DSS approval rests with the Acquirer.

### **Step 4 – Technical Development**

You are now ready to develop a solution based on 'PointWare Ekspedient', 'UserControl', 'DLL' or 'LPP'. See the appendixes for the Technical Guides and programming examples.

Verifone Denmark's development department ([development.hrv@verifone.com](mailto:development.hrv@verifone.com)) will help you with all the questions you may experience during the development. You can also see the FAQ in this document – it will answer the most common questions.

### **Step 5 – Certification**

When the solution is complete it must be certified to make sure it meets all requirements for Card Data security and that it functions correctly. Nets Certification is handled by Nets. Verifone Denmark can assist you with the right contact to Nets.

### **Step 6 – Additional information required by PCI-SSC**

The RTL system used for the authenticated remote software distribution should be evaluated by a QSA as part of a PCI DSS assessment. The terminal comes with certain audit trails enabled. The log is automatically sent to Verifone Denmark. The logs are available by calling the customer support. If you are using wireless network within your business network you must make sure that firewalls are installed that deny or control (if such traffic is necessary for business purposes) any traffic from the wireless environment into the rest of the network environment.

In case you are using a wireless network you must also make sure that:

- Encryption keys were changed from vendor defaults at installation
- Encryption keys are changed anytime someone with knowledge of the keys leaves the company or changes position
- Default SNMP community strings on wireless devices are changed
- Firmware on wireless devices is updated to support strong encryption, WPA/WPA2. Please note that WEP must not be used for new installations and is not allowed after June 30, 2010

- Other security related vendor defaults are changed

Your Verifone Denmark terminal allows transmission over public networks, e.g. Internet. To protect sensitive data your Verifone Denmark terminal uses the PSAM chip provided by Nets. This chip uses triple DES encryption with a unique key per transaction. To connect your Verifone Denmark terminal to public networks you do not need to take any further action regarding encryption.

Before exchanging or updating the Nets PSAM, in order to remove any historical data stored by previous versions of the PSAM it is absolutely necessary for PCI DSS compliance to remove the historical data. This can be performed by going into the Menu, Option (4) Admin, Option(10) Slet Datastore, Option (11) Flyt/Slet Advice.

It is very important for PCI-DSS compliance that all accounts allowing access to any PCs, servers, and databases with payment applications and cardholder data must be unique. You must not use generic or shared user accounts with unsecure stored passwords. The Verifone Denmark solution does not hinder nor affects these requirements in any way.

*Procedure to facilitate Centralized Log Management (PCI-DSS Requirement 10.5.3)*

DK-8111 facilitates centralized log management to a Syslog compatible centralized log management server. All the logs can be offloaded to such a server by accessing the Menu, Option (4) Admin, Option (14) Send Event Log. Then you will be prompted for the IP address of the Syslog server.

The following payment terminals are supported:

**Hardware PIN Entry Devices terminals:**

Atos Worldline Banksys XENTA

Hardware #: 90640000xx 90640000xx REV\_L, PTS 1.x approval 4-30001

Atos Worldline Banksys XENTA

Hardware #: 90640100xx Rev 0, 90640100xx rev. A, PTS 2.x approval 4-30051

Atos Worldline Banksys Xentissimo

Hardware #: 9066000xx rev A <BR> 90660000xx rev F 90660000xx rev G, PTS 1.x approval 4-30007

Atos Worldline Banksys YOMANI

Hardware #: 90670000xx Rev: 1 90670000xx Rev: A, PTS 2.x approval 4-30046

Atos Worldline Banksys YOMANI XR/ML

Hardware #: 90700x00x Rev: Ax, PTS 3.x approval 4-30092

Atos Worldline Banksys YOMANI XR/ML

Hardware #: 90700x00x Rev: Ax, 90700x00x Rev: Bx, PTS 3.x approval 4-30094

VeriFone Inc. Vx520

Hardware #: M252-6xx-xx-xxn-2, M252-7xx-xx-xxn-2, (online only) M252-1xx-xx-xxn-2, (SP version) M252-8xx-xx-xxn-2, PTS 2.x approval 4-30050

VeriFone Inc. Vx520 (VOS)

Hardware #: M254-6xx-xx-xxx-3, M254-7xx-xx-xxx-3, OP support M254-x5x-xx-xxx-3, M254-x7x-xx-xxx-3, PTS 3.x approval 4-10125

VeriFone Inc. Vx520, VX520 3G

Hardware #: M252-6xx-xx-xxx-3, M252-7xx-xx-xxx-3, M252-x5x-xx-xxx-3, M252-x7x-xx-xxx-3, M252-65x-Cx-xxxx-3, M252-69x-Cx-xxx-3, M252-79xCx-xxx-3, M252-69x-Gx-xxx-3, M252-79x-Gx-xxx-

3, PTS 3.x approval 4-300052

Verifone Inc. Vx680

Hardware #: M268-7xx-xx-xxn-2, PTS 2.x approval 4-20146

VeriFone Inc. Vx680, Vx680-E1

Hardware #: M268-70x-xx-xxn-3, M268-73x-xx-xxn-3, M268-74x-xx-xxn-3, M268-76x-xx-xxn-3, M268-77x-xx-xxn-3, M268-78-x-xx-xxn-3, M268-79x-xx-xxn-3, PTS 3.x 4-30053

VeriFone Inc. Vx820

Hardware #: M282-X0X-XX-XXX-R-3, M282-X0X-XX-XXX-2, PTS 2.x approval 4-40053

### **Step 7 – Using PA DSS approved PSAMs**

The PSAM card used in terminals is delivered by Nets. The version used in the terminals should be PA DSS approved to ensure compliance.

### **Step 8 – Terminal logging**

In order to be PA DSS 2.0 compliant, the terminal will log various information related to events and actions on the terminal. It is possible to print this log information and it is also possible to send it as syslog format to an IP address specified.

## **12.3 Other Documents**

### **FAQ(s)**

Frequently Asked Questions

### **Technical Guides**

FlexDriver DLL

Local Payment Protocol LPP

# 13 | Development Agreement

## 13.1 Scope of Agreement

The following development agreement is contracted between \_\_\_\_\_ henceforth 'the Client', and Verifone Denmark A/S, henceforth 'Verifone'. The development agreement relates to the integration of Verifone Credit Card Machines (henceforth 'Terminals') with the Clients' cash till.

## 13.2 Subject Matter

The agreement includes Verifone Terminals, software, and support along with the Clients' products, when these are an integral part of the solution.

## 13.3 Verifone's Obligations

Verifone Denmark is obliged to deliver a *flexdriver* with appurtenant documentation to the Client. A *flexdriver*, which is preapproved by Nets, is delivered, unless something else is agreed upon in writing, which can occur when new versions of software and *flexdriver* are introduced.

The functionality of the *flexdriver* is as described in the appurtenant documentation (the version number can be seen in appendix B).

Verifone is responsible for corrections in the current version, only when Verifone acknowledges a problem as an error. New or changed functions can solely be implemented in connection with the introduction of new versions of the *flexdriver*.

Documentation as well as support (maximum 5 hours) will be provided by Verifone free of charge in connection with the development of the solution. When support beyond the 5 hour maximum is needed, the Client must pay the going hourly rate. Verifone will provide one additional hour for the presentation of the course of development. This representation will take place at Verifone, and will be free of charge.

Requests concerning support must be addressed to [development.hrv@verifone.com](mailto:development.hrv@verifone.com). The Client can expect an answer within one workday. The Client cannot expect the above mentioned response time, if the timetable, as described in appendix C, is not followed.

## 13.4 The Client's Obligations

It is the Client's responsibility to make all the necessary agreements with Nets, just as it is the Client's responsibility to comply with requirements set by Nets. In order to see a directory of the requirements go to Nets website.

The Client must participate in a meeting with Verifone's development department to discuss the planning and the start-up phase. This meeting will take place before the development of the



solution is launched. This meeting is cost-free.

The Client must report all possible errors in the software made by Verifone to:

`development.hrv@verifone.com`.

The Client must defray all expenses to Nets in connection with the certification of the solution.

The Client must submit the detail specification to Verifone before the certification at Nets.

The Client must submit a copy of the test report to Verifone as soon as possible after the approval by Nets, so that Verifone is able to record the integration.

It is the Clients responsibility, that the selected/available way of communication is workable with the terminal. Cf. Firewalls, line access etc.

If the Client wish to report an error, this must be documented via trace or the like, in order for Verifone to recreate the error. An error will not be recognized as such until the above mentioned procedure is followed.

Once the project is completed future inquiries must be made to Verifone's customer service, provided that the Client has signed a service agreement.

## **13.5 Service**

The Client will provide the service and help-desk function to the end-user in relation to the actual integration of the cash till.

## **13.6 Independent Parties**

Both parties must act as independent and autonomous business partners. None of the parties are entitled to use the other party's name or logo unless it is done in connection with the sale of the other party's products and on product sheets. Both parties are obligated to act with loyalty towards the other party. This loyalty covers, but is not limited to, all situations, which may be of interest of the other party, or situations, which could influence the collaboration.

## **13.7 Rights of Third Parties**

Verifone will warrant that everything, which have been delivered does not violate the rights of others including patents and/or copyrights.

In order for the warrant to be legally valid, the Client must notify Verifone in writing immediately if the Client realizes possible violations of rights. Furthermore the Client must assist Verifone during the case to a sufficient extent.

## **13.8 Liability**

Verifone will make all reasonable efforts and to the greatest extent be attentive during the execution of the specified services (outlined in the development agreement). But Verifone is no case liable for any loss or damage occurring as a result of any delay, action or omission of service or the completion hereof with the exception of severe negligence.

## **13.9 Commencement of Agreement**

The agreement will become effective when signed by both parties.

## **13.10 Termination of Agreement**

Both parties can terminate the agreement if the other party is guilty of violating the agreement. The parties can also terminate the agreement if one of them does not contribute in a positive manner to the agreed timetable for the development of the solution, for which reason the development of the solution can not be fulfilled. The agreement can not, however, be annulled, if the other party is not responsible for the violation.

The agreement can be annulled, if one of the parties is declared bankrupt, their payments are suspended, if they have agreed on a compulsory composition with their creditors, or a similar settlement of debts.

## **13.11 Transference of Agreement**

Both parties are able to transfer the agreement to another group related company. However, they have to inform the other party in writing well in advance. In addition to this, the agreement can only be transferred with a preceding written consent from the other party.

## **13.12 Written Agreements**

This agreement will replace all other previously signed agreements, both verbal- and written agreements, and future changes can only be made in writing and with the approval from both parties.

## **13.13 Confidential Information**

Each party is obligated to ensure that confidential information about the other party obtained directly or indirectly through the collaboration, is not brought to the knowledge of a third party.

In addition to the conditions of the agreement, confidential information must be regarded as any notification or information about technical, commercial or of a similar character. Including information which directly or indirectly concern production processes, technical practices, all manner of planning and other rights; among these are immaterial rights, concepts and projects, combination of products, market - and economic information.

Confidential information, which is handed over to the other party before the entering of the Development Agreement, is included in the confidentiality commitment.

The confidentiality commitment continues after the termination of this agreement.



## 13.17 Signature

With the signature below both parties are accepting the provisions in this agreement. The parties will at the same time accept the fact that the appendixes can be changed through the duration of the agreement, though this requires a written accept from both parties.

Dato:	Dato:
Kunden:	Verifone:
	Verifone Denmark A/S Knapholm 7 2730 Herlev Phone: +45 44 53 16 10 Fax: +45 44 53 46 20 CVR nr.: 15 40 12 81

## 13.18 Appendix A – Contact

### Contact information at Verifone Denmark

Function	Name	E-mail	Phone
Sales		<a href="mailto:sales.hrv@verifone.com">sales.hrv@verifone.com</a>	+45 44 50 16 52
Customer Service		<a href="mailto:kundeservice@verifone.com">kundeservice@verifone.com</a>	+45 44 53 75 00
Development		<a href="mailto:development.hrv@verifone.com">development.hrv@verifone.com</a>	+45 44 50 16 51

### Contact information at the Client

Function	Name	E-mail	Phone
Sales			
Ecomony			
Customer Service			

Date tables were filled in: \_\_\_\_\_

## 13.19 Appendix B – Development Agreement

This appendix will outline a specific project plan and time line for the development project, and in this connection describe the preconditions and requirements, which must be fulfilled.

### Project organization

<b>Product Manager at Verifone Denmark</b>	
Name	
Phone	
E-mail	

<b>Product Manager at the Client</b>	
Name	
Phone	
Mobile Phone Number	
E-mail	

<b>Contact Person at the End Customer</b>	
Name	
Phone	
Mobil Phone Number	
E-mail	

<b>Contact Person when new versions of terminal/flexdriver</b>	
Name	
Phone	
Mobil Phone Number	
E-mail	

<b>Access Type</b>			
<input type="checkbox"/> Wlan	<input type="checkbox"/> USB	<input type="checkbox"/> Ethernet	<input type="checkbox"/> GSM
Comment:			

<b>Basic Information about Cash till Integration</b>	
Cash till Application:	

<b>Cash till Type</b>		
<input type="checkbox"/> PC	<input type="checkbox"/> Prompt	<input type="checkbox"/> Andet
Comment:		

<b>Operating System</b>		
<input type="checkbox"/> Windows	<input type="checkbox"/> Linux	<input type="checkbox"/> Other
Comment:		

<b>EPJ Platform</b>
(e.g. C5, Visma, Navision)

<b>Description of Cash till equipment (standard configurations)</b>	
Cash till	
Printer	

Development tools (incl. Version number)				
<input type="checkbox"/> Access Version: _____	<input type="checkbox"/> Visual Basic Version: _____	<input type="checkbox"/> C Version: _____	<input type="checkbox"/> C++ Version: _____	<input type="checkbox"/> Delphi Version: _____
<input type="checkbox"/> Other Version: _____				
Comment:				

Integration Type	
<input type="checkbox"/> TCP/IP	<input type="checkbox"/> RS232
Comment:	

Integration via				
<input type="checkbox"/> DLL	<input type="checkbox"/> UserControl	<input type="checkbox"/> LIB	<input type="checkbox"/> LPP	<input type="checkbox"/> SpinConnect
<input type="checkbox"/> Point Ware Ekspedient (PWE)				
Comment:				

Terminal Type		
<input type="checkbox"/> Yomani	<input type="checkbox"/> VX820	<input type="checkbox"/> VX680/VX690
Comment:		

Date:

\_\_\_\_\_  
The Client

Date:

\_\_\_\_\_  
Verifone Denmark A/S



## 13.20 Appendix C – Timetable

The Timetable will be outlined when signing the agreement, and the Cash till integration will, as a general rule, be completed within a period of two – three months from the defined starting point and to the approval at Nets/Verifone. The below outlined part-activities reflect Verifone's present experiences from completed integration projects. All part-activities are as a general rule obligatory, but they can be conducted at the same date, if the circumstances require it.

Activity	Description	Date/Time	Completed
Agreement signed			<input type="checkbox"/>
Project start			<input type="checkbox"/>
Flex-driver handed over			<input type="checkbox"/>
Development Terminal	Is the development terminal delivered?		<input type="checkbox"/>
Pre-certification	Test of the solution at Verifone against Test host. <i>Verifone can in return for a fee perform the Pre-certification.</i>		<input type="checkbox"/>
Nets certification	Test at Nets		<input type="checkbox"/>
Project completed			<input type="checkbox"/>

Date:

Date:

\_\_\_\_\_  
The Client

\_\_\_\_\_  
Verifone Denmark A/S

The Client's expectations to number of sold terminals: \_\_\_\_\_

# 14 | Udviklingsaftale

## 14.1 Aftalens omfang

Mellem \_\_\_\_\_ herefter kaldet Kunden og Verifone Denmark A/S, herefter kaldet Verifone, er indgået følgende udviklingsaftale.

Udviklingsaftalen vedrører integration af Verifone betalingsterminaler med kundens kasseløsning.

## 14.2 Aftalens genstand

Aftalen omfatter Verifone flexterminaler, programmel og support og Kundens produkter, når disse indgår i en samlet løsning.

## 14.3 Verifones forpligtelser

Verifone forpligter sig til at levere flexdriver med tilhørende dokumentation til kunden.

Med mindre andet er skriftligt aftalt, hvilket kan forekomme i forbindelse med introduktion af nye versioner af terminalprogrammel og flexdriver, leveres en af Nets på forhånd godkendt flexdriver. Flexdriverens funktionalitet er som beskrevet i den tilhørende dokumentation (versionsnummer fremgår af aftalens bilag B).

Først når Verifone har anerkendt et problem som en fejl, er Verifone ansvarlig for fejlrettelser i den gældende version, hvorimod nye eller ændrede funktioner kun kan indføres i forbindelse med introduktion af nye versioner af flexdriveren.

I forbindelse med udvikling af løsningen stiller Verifone dokumentation samt support (maks. 5 timer) til rådighed for kunden uden beregning. Ved behov for support ud over 5 timer betales den til enhver tid gældende timepris herfor. Verifone afsætter yderligere én time til introduktion og præsentation af udviklingsforløb. Denne foregår hos Verifone og er uden beregning.

Henvendelser om support rettes til udviklingsafdelingen på mailadressen:

development.hrv@verifone.com.

Man kan forvente svar inden for én arbejdsdag. Såfremt tidsplanen i bilag ikke er overholdt, kan Kunden ikke forvente ovenstående svartid.

## 14.4 Kundens forpligtelser

Det er kundens ansvar at indgå de nødvendige aftaler med Nets, ligesom det er kundens ansvar, at de af Nets stillede krav overholdes. For en oversigt over disse henvises der til Nets website.

Kunden skal før udviklingen påbegyndes deltage i et opstarts-/ planlægningsmøde med Verifone's udviklingsafdeling. Dette er vederlagsfrit for Kunden.

Kunden skal rapportere eventuelle fejl i Verifones programmel til mailadressen:

development.hrv@verifone.com

Kunden afholder selv omkostninger til Nets for certificering af løsningen.

Kunden skal fremsende detailspecifikationen til Verifone inden certificering hos Nets.

Kunden skal indsende en kopi af testrapporten til Verifone snarest efter godkendelse hos Nets, således Verifone kan registrere integrationen.

Det er kundens ansvar, at den valgte/tilgængelige kommunikationsform er funktionel med terminalen. Jvf. firewalls, lineadgang etc.

Såfremt Kunden ønsker at rapportere en fejl, skal denne dokumenteres via trace eller lignende, således at Verifone kan genskabe fejlen. Eventuelle fejl vil ikke blive anerkendt, før denne procedure er overholdt.

Når projektet er afsluttet, skal fremtidig henvendelse ske til Verifone Kundeservice, såfremt man har tegnet en serviceaftale.

## **14.5 Service**

Kunden varetager selv servicering og help-desk funktion over for slutbrugeren i relation til selve kasseintegrationen.

## **14.6 Uafhængige Parter**

Begge parter optræder som selvstændige og uafhængige forretningspartnere.

Ingen af parterne er berettiget til at anvende den anden parts navn og/eller logo undtagen i forbindelse med salg af hinandens produkter og på produktblade.

Begge parter er forpligtet til at optræde loyalt over for den anden part. Denne loyalitet omfatter, men er ikke begrænset til, oplysning om ethvert forhold, der er af interesse for den anden part, eller som må antages at være af betydning for samarbejdet.

## **14.7 Tredje mands rettigheder**

Verifone indestår for, at det leverede ikke krænker andres rettigheder, herunder patenter eller ophavsrettigheder.

Indeståelsen forudsætter dog, at Kunden straks giver Verifone skriftlig meddelelse, når Kunden bliver opmærksom på eventuelle rettighedskrænkelser, og at Kunden bistår Verifone under sagen i fornødent omfang.

## **14.8 Ansvar**

Verifone vil udfolde alle rimelige anstrengelser og udvise størst mulig påpasselighed under udførelsen af de Udviklingsaftalens anførte ydelser, men er i intet tilfælde, undtaget grov uagtsomhed, ansvarlig for noget tab eller nogen skade, der er en følge af nogen forsinkelse, handling eller undladelse ved servicearbejdet eller gennemførelsen heraf.

## **14.9 Aftalens ikrafttræden**

Aftalen træder i kraft, når den er underskrevet af begge parter.

## **14.10 Ophævelse af aftalen**

Begge parter kan hæve aftalen, hvis den anden part gør sig skyldig i væsentlig misligholdelse af denne, herunder at én af parterne ikke bidrager positivt til at de aftalte tidsplaner for udviklingen af Kundens løsning overholdes, og de dermed ikke kan indfries. Aftalen kan dog ikke hæves, hvis den anden part ikke kan gøres ansvarlig for misligholdelsen.

Endelig kan aftalen hæves, hvis én af parterne erklæres konkurs eller kommer under betalingsstandsning, tvangsakkord eller lignende gældsordning.

## **14.11 Overdragelse af aftalen**

Begge parter kan ved skriftligt at orientere den anden part overdrage aftalen til et andet koncernforbundet selskab. Herudover kan aftalen kun overdrages med den anden parts forudgående skriftlige samtykke.

## **14.12 Skriftlige aftaler**

Nærværende aftale erstatter alle evt. tidligere indgåede aftaler, såvel mundtlige som skriftlige, og fremtidige ændringer kan kun ske skriftligt og med godkendelse af begge parter.

## **14.13 Fortrolighed**

Parterne forpligter sig til at sikre, at fortrolige oplysninger erhvervet om den anden part, direkte eller indirekte gennem samarbejdet, ikke kommer til tredjemands kendskab.

Ved fortrolige oplysninger forstås, ud over vilkårene i Udviklingsaftalen, enhver oplysning eller information – uanset form – af teknisk, kommerciel eller tilsvarende karakter om hinanden, herunder oplysninger som direkte eller indirekte vedrører produktionsprocesser, tekniske fremgangsmåder, alle former for planlægning og andre rettigheder, herunder immaterielle rettigheder, koncepter og projekter, produktsammensætninger, markedsoplysninger og økonomiske oplysninger.

Fortrolige informationer, som er overgivet til den anden part forud for tidspunktet for Udviklingsaftalens indgåelse, er omfattet af fortrolighedsforpligtelsen.

Fortrolighedsforpligtelsen fortsætter efter aftalens ophør.

## **14.14 Force majeure**

Såfremt der indtræder uventede og ekstraordinære forhold uden for parternes kontrol, og som parterne ikke ved aftalens underskrift burde have taget i betragtning (herunder strejker) og ej heller burde have undgået eller overvundet, bevirker dette, at aftalens rettigheder og pligter bliver

suspenderet for begge parter i det tidsrum forholdet vedrører. Hver af parterne er forpligtiget til at gøre sit yderste for at overvinde sådanne hindringer, således at tab begrænses mest muligt. Såfremt det omtalte forhold hindrer Verifone i overholdelse af aftalen i et tidsrum, der overstiger 3 måneder, kan aftalen annulleres af begge parter med 14 dages varsel.

## 14.15 Værneting/Lovvalg

Retsforholdet i følge denne aftale og bilagene hertil, samt disses fortolkning afgøres efter dansk ret. Aftalens bestemmelser går forud for lovgivningens bestemmelser, i det omfang man kan fravige fra disse.

Såfremt der måtte opstå uoverensstemmelse mellem parterne om denne aftale, bilagene her- til eller disses fortolkning og/eller udfyldning, skal parterne med en positiv, samarbejdende og ansvarlig holdning indlede forhandlinger på højt plan i parternes organisationer med henblik på at løse tvisten. Disse forhandlinger kan maksimalt have en tidsmæssig udstrækning på 15 arbejds- dage.

Såfremt der ved forhandling ikke opnås nogen løsning, skal parterne inden 10 arbejdsdage i fællesskab udpege - eller anmode Dansk IT om at udpege - en uafhængig og sagkyndig mægler, der kan mægle og komme med ikke-bindende forslag til tvistens løsning.

Såfremt der ved mægling heller ikke opnås nogen løsning, er hver af parterne berettiget til at kræve uoverensstemmelserne afgjort ved Sø- og Handelsretten i København.

## 14.16 Bemærkninger

Særlige bemærkninger til aftalen:

## 14.17 Underskrift

Ved nedenstående underskrift accepterer begge parterne betingelserne i denne aftale. Samtidig accepterer partnerne, at bilagene kan ændres i aftalens løbetid. Dette forudsætter dog skriftlig accept fra begge parter på det pågældende bilag.

Dato:	Dato:
Kunden:	Verifone:
	Verifone Denmark A/S Knapholm 7 2730 Herlev Phone: +45 44 53 16 10 Fax: +45 44 53 46 20 CVR nr.: 15 40 12 81

## 14.18 Bilag A - Kontrakt

### Kontaktinformationer hos Verifone

Funktion	Navn	E-mail	Telefon
Salg		<a href="mailto:sales.hrv@verifone.com">sales.hrv@verifone.com</a>	44 50 16 52
Kundeservice		<a href="mailto:kundeservice@verifone.com">kundeservice@verifone.com</a>	44 53 75 00
Udvikling		<a href="mailto:development.hrv@verifone.com">development.hrv@verifone.com</a>	44 50 16 51

### Kontaktinformationer hos Kunden

Funktion	Navn	E-mail	Telefon
Salg			
Økonomi			
Kundeservice			

Udfyldt dato: \_\_\_\_\_

## 14.19 Bilag B - Udviklingsaftale

Dette bilag opstiller en konkret projekt- og tidsplan for udviklingsprojektet, og vil i den sammenhæng beskrive de forudsætninger og krav, som skal være opfyldt.

### Projekt organisation

Produktansvarlig hos Verifone	
Navn	
Telefonnummer	
E-mail	

Produktansvarlig hos Kunden	
Navn	
Telefonnummer(direkte)	
Mobiltelefonnummer	
E-mail	

Kontaktperson hos slutkunde	
Navn	
Telefonnummer(direkte)	
Mobiltelefonnummer	
E-mail	

Kontaktperson ved nye versioner af terminal/flexdriver	
Navn	
Telefonnummer(direkte)	
Mobiltelefonnummer	
E-mail	



<b>Opkoblingstype</b>			
<input type="checkbox"/> Wlan	<input type="checkbox"/> USB	<input type="checkbox"/> Ethernet	<input type="checkbox"/> GSM
Kommentar:			

<b>Basisinformation om kasseintegration</b>	
Kasseapplikation:	

<b>Kasstype</b>		
<input type="checkbox"/> PC	<input type="checkbox"/> Prompt	<input type="checkbox"/> Andet
Kommentar:		

<b>Operativ system</b>		
<input type="checkbox"/> Windows	<input type="checkbox"/> Linux	<input type="checkbox"/> Andet
Kommentar:		

<b>EPJ platform</b>
(f.eks. C5, Visma, Navision)

<b>Beskrivelse af kasseudstyr (standard konfigurationer)</b>	
Kasse	
Printer	

<b>Udviklingsværktøj (inkl. versionsnummer)</b>				
<input type="checkbox"/> Access Version: _____	<input type="checkbox"/> Visual Basic Version: _____	<input type="checkbox"/> C Version: _____	<input type="checkbox"/> C++ Version: _____	<input type="checkbox"/> Delphi Version: _____
<input type="checkbox"/> Andet Version: _____				
Kommentar:				

<b>Integrationstype</b>	
<input type="checkbox"/> TCP/IP	<input type="checkbox"/> RS232
Kommentar:	

<b>Integration via</b>				
<input type="checkbox"/> DLL	<input type="checkbox"/> UserControl	<input type="checkbox"/> LIB	<input type="checkbox"/> LPP	<input type="checkbox"/> SpinConnect
<input type="checkbox"/> Point Ware Ekspedient (PWE)				
Kommentar:				

<b>Terminal type</b>		
<input type="checkbox"/> Yomani	<input type="checkbox"/> VX820	<input type="checkbox"/> VX680/VX690
Kommentar:		

Dato:

Kunden

Dato:

Verifone Denmark A/S

## 14.20 Bilag C - Tidsplan

Tidsplanen opstilles ved indgåelse af aftale, og som udgangspunkt forventes en kasseintegration afsluttet inden for en periode af 2-3 måneder fra defineret starttidspunkt til godkendelse hos Nets/Verifone. De nedenfor opstillede delaktiviteter afspejler Verifones nuværende erfaringer fra gennemførte integrationsprojekter. Som udgangspunkt er alle delaktiviteter obligatoriske, men man kan sagtens forestille sig, at delaktiviteter afholdes samme dato – hvis situationen lægger op til det.

Aktivitet	Beskrivelse	Dato/Tid	Afsluttet
Aftale indgået			<input type="checkbox"/>
Projekt start	Dato defineret som startdato for projekt.		<input type="checkbox"/>
Flex-driver udleveret			<input type="checkbox"/>
Udviklingsterminal	Er udviklingsterminal leveret?		<input type="checkbox"/>
Præcertificering	Test hos Verifone af løsningen, hvor dette gøres mod Test host. <i>Verifone kan mod vederlag udføre Præcertificering for Kunden.</i>		<input type="checkbox"/>
Nets certificering	Test hos Nets		<input type="checkbox"/>
Projekt afsluttet			<input type="checkbox"/>

Dato:

Dato:

\_\_\_\_\_

Kunden

\_\_\_\_\_

Verifone Denmark A/S

Kundens forventninger til antal solgte terminaler: \_\_\_\_\_

# 15 | FAQ

## 15.1 Basic Information

When contacting [development.hrv@verifone.com](mailto:development.hrv@verifone.com) we need to know the following basic information in order for us to help you the best way possible:

- Problem is found on terminal number.
- Events leading to the question/problem.
- How often do you see the problem?
- Can this problem be reproduced or was it a single event.
- Was the UserControl or DLL used?
- Version of UserControl/DLL, as found under properties.
- If possible - tracefiles containing the problem, with some before/after data.
- Time of the problem.

## 15.2 How is Certification Handled?

Nets have to approve the results of a test performed with your ECR application and flexterminal before the solution may be used in real life. Verifone Denmark can help with a pre certification before the certification at Nets, pointing out missing functionality in your application to be corrected before Nets certification.

Contact [qa.hrv@verifone.com](mailto:qa.hrv@verifone.com) for pre certification.

Contact [it-verification@nets.eu](mailto:it-verification@nets.eu) for the final certification.

Further information on certification can be found on Nets' homepage.

### 15.2.1 Important before a Certification

Verifone need to know any special setup used by your ECR integration including:

- Number of test terminal used for the Certification
- The name of the Integrator
- The name of the ECR solution
- Version of the ECR solution
- Integration via LPP/DLL/UserControl or PWE
- If connecting via RS232/USB the baudrate
- ReceiptType
- TokenBasedpayment
- etc.

Upon receiving this information ([development.hrv@verifone.com](mailto:development.hrv@verifone.com)) Verifone Denmark will update our database, the configuration, and setting your test terminal to use this setup.

After you performed a param download on the test terminal, it will be running with this setup, and you are ready to go to Pre Certification and Certification at Nets.

After you have been Certified by Nets, you can order terminals with this exact setup, by giving the unique:

- Name of the Integrator
- Name of the ECR solution
- Verson of the ECR solution

This ensures that the terminals will be as the one Certified.

### **15.3 Description of Best Practice**

A description of best practice can be found in TRG OTRS v. 3 book 1.pdf section 9. Best practice gives a good introduction to the development project you are going to make. The OTRS can be found on Nets' website.

## 15.4 Configuration of the Test Terminal

To ensure correct setup of the test terminal, Verifone Denmark needs to know the terminal number of the terminal you will use during development.

Test terminals have a 99XXXX number.

After changes of setup parameters in Verifone's internal database system, data is ready to be downloaded into the terminal. This download is always initiated from the terminal.

Functions to be used if you have an Operator Unit (OPU):

Menu – 4 – 3 Download parameters

Menu – 4 – 4 Download program

You will be asked for a password to be allowed to perform these tasks, it's the 4 digit number you selected upon buying the test terminal.

Your final ECR (Electronic Cash Register) application should also be able to perform download of parameters, programs etc.

### 15.4.1 Special Function Configuration

Some of the more advanced functions in the terminal requires a special parameter setup. Likewise the functions must be enabled in the UserControl/DLL for correct function.

Some examples:

- Token
- Gratuity
- IP Routing
- Receipt type
- Fee calculation
- Local card check
- DCC (Direct Currency Conversion)

## 15.5 Connect a Terminal to ECR

The terminals can be connected in two ways to the ECR application.

- Via RS232
  - A special serial cable is needed to connect the terminal to the PC running the ECR application. This cable is delivered with the test terminal – if the cable is missing a new can be ordered at [sales.hrv@verifone.com](mailto:sales.hrv@verifone.com).
- Via Ethernet
  - An ordinary Ethernet cable connected to a hub/switch with Ethernet connection to the PC running the ECR application.



## 15.6 How to Connect the Terminal

### Via RS232:

- LINE – Ethernet connection, for connection to Nets and Verifone
- Electronic Cash Register (ECR), serial cable to PC running the ECR application
- Power

### Via Ethernet:

- LINE – Ethernet connection, for connection to Nets/Verifone Denmark and PC running the ECR application
- Power

### Via USB (only some terminal types):

- USB – delivers power from PC and also the serial connection to the PC
- Ethernet – Ethernet connection, for connection to Nets and Verifone Denmark

NB! The terminal must be rebooted if you will use the OPU (Operator Unit), after connection the OPU. Likewise the terminal must be rebooted after removing the OPU from the setup.

Normally OPU is used in a startup process to operate the terminal, so it's ready with latest parameters/software etc. without the need to have a running ECR application.

### 15.6.1 Ethernet Demands

To get the optimal usage of flex terminal its network connection should be isolated from surrounding network traffic. The reason for this demand:

Often networks experience broadcast storms and the flex terminals can't cope with the extra data load.

- The terminal has limited processing power and compared with a PC it's very limited.
- Different network traffic has different demands to decode of incoming packets.
- All packets have to be decoded according to type and first at the appropriate ISO level it can be accepted or rejected. The higher the ISO level, the more processing power has been used.

### 15.6.2 Ethernet – How the Terminal can obtain an IP Address

IP addresses for the terminal can be set in different ways:

- Default DHCP is used, and the terminal receives its IP address via this method.
- Verifone Denmark enters a IP address for the terminals Ethernet in Verifone Denmark's internal database system, and this data is downloaded to the terminal by a call to download parameters from the ECR application or by installing the Operator Unit (OPU) and calling menu 4 3
- Using the OPU – you have access to menu 6 3 IP addresses

Using menu 6 3 to enable DHCP after entering password use this sequence to all questions:

0 → Corr ("Slet" yellow button) → OK ("Godkend" green button) to all questions about IP etc.

### 15.6.3 Ethernet – Network Required Addresses/Names

To connect to Nets and Verifone Denmark, the terminal must be able to reach both through the network, using DNS-lookup on specific names, and in the same way specific ports must be open.

Default DNS-names, IP-address, and port:

pbs1.point-ts.dk	Port 19000
pbs2.point-ts.dk	Port 19000
test.point-ts.dk	Port 22000
test2.point-ts.dk	Port 22000
time.point-ts.dk	Port 13
rtl.point-ts.dk	Port 5214
param.point-ts.dk	Port 24000
ekvittering.point-ts.dk	Port 80
storebox.point-ts.dk	Port 80
lp.point-ts.dk	port 10760
pplp.point-ts.dk	port 10960
cbp.point-ts.dk	port 9000
api.swipp.bec.dk	port 443

Verifone Denmark has developed a test application to verify access to the above DNS-names, and to check the ports are available.

The ECR admin function 'network report' makes this test on the terminal.

If OPU connected, you could run menu-7-6 (menu-test-network).

Network Address Translation (NAT) is possible; with this setup it is possible to use other addresses internally on the network. Addresses must be routed to the correct ones at some Verifone Denmark in your network.

This test doesn't guarantee the terminal will function correctly, the terminals IP interface only accept DNS lookup if data found directly on the DNS configured, PC's IP interface use another algorithm for DNS lookup, including lookup on neighbor DNS servers. In other cases the PC might be known on a firewall in the network, where the terminal is unknown.

If you experience network problems with the terminal in a network with a firewall, try moving the terminal to the outside of the firewall, if it works here a network expert must configure your firewall to allow terminal traffic.

### 15.6.4 IP and Ports used during Certification

Nets	KOPI	test.point-ts.dk	port 22000
Nets	KOPI	test2.point-ts.dk	port 22000

Nets uses a white-list for the above mentioned services and your external Internet address must be known before contact attempts will be answered. Make sure your own firewall is open as well. Contact: [it-verification@nets.eu](mailto:it-verification@nets.eu) with information of your external Internet address, to be put on this white-list.

### 15.6.5 No connection to terminal - Disable Cisco Skinny Call Control Protocol

Verifone Denmark use port 2000 to communicate with terminal and on some network equipment this is seen as the "Cisco Skinny Call Control Protocol", it might help to disable this protocol in the equipment.

### 15.6.6 Guide to Ethernet trace.

This guide is "as is", Verifone don't deliver network support of any kind. Use a network expert to analyze your network, to advice you on network setup, traffic analyses and configuration. Below is a setup used by Verifone to analyze network traffic to and from Verifone terminals.

Requirements:

- Program to make an Ethernet trace.
- HUB or Switch, to allow sniffing the Ethernet traffic.

Switches with port mirror (portspejling)

Netgear GS105E

Netgear GS108E

HP 1810-8G

Cisco SF302-08P

Switches have not been tested by Verifone, but we have read about others having done Ethernet traces with success.

- Network expert to analyze data.

One Ethernet trace program to be used is WireShark, the program can be found on [www.wireshark.org](http://www.wireshark.org) Make sure the HUB is a real HUB, where you will be able to see all traffic on all ports.

Setup:

- Always power down equipment before fiddling with cables. In this case HUB, Verifone terminal and Ethernet trace PC.
- Insert the HUB between the terminal and the network connection used by the Verifone terminal.
- Connect the Ethernet trace PC to the HUB.
- Power up HUB.
- Power up Ethernet trace PC and start capture with no filters.
- Power up Verifone Denmark terminal.

#### 15.6.6.1 What to Capture

All traffic on the network reaching the Verifone terminal.

Analyze the captured data:

Let the network expert analyze the data.

This requires knowledge of the network setup/layout, equipment used, programs using the network, baseline traffic loads etc.

#### 15.6.6.2 What to look for

- Errors of any kind.
- Warnings of any kind.
- Unknown traffic.
- Traffic loads.
- Where time is spend, and is the response times normal. DNS lookup, routing etc.
- Broadcast storms.

Let the network expert come up with solutions for problems found.

The impact a given solution will have on the network and Your Company.

## **15.7 What is needed to make a Test Transaction**

From Nets you must have a development agreement (udviklingsaftale) and buy a test PSAM for your test terminal and some test cards to be used for test transactions.

You can read more about TEST PSAM's and TEST cards on Nets website.

## 15.8 'Ingen kvittering' – 'No Receipt'

After a param download to the terminal, some old data might reside in the terminal. If you try to solve this problem using one of our demo programs you get an error 65539 No receipt or "Ingen kvittering" – "No Receipt", unable to be unlocked by calling function UNLOCK\_LASTRECEIPT, LAST\_RECEIPT. Try to run Administration functions:

- Get last receipt from terminal and unlock terminal.

### 15.8.1 Using ADMIN and 'Pasord'

1. Reboot terminal.
2. Wait for the text ADMIN? In lower right corner of display and within 3 seconds, press right-most up-arrow. (Yomani: Right button, of the three just below display)
3. Enter 'pasord' 746578. Now a menu is shown.

With arrow in lower left corner, select menu 18 'FLERE FUNKTIONER' and then '1 – LÅS INGEN KVIT.OP' press green button ("GODKEND") to approve.  
After this you must reboot the terminal.

NB! Operator Unit (OPU) must NOT be connected during this operation.

## 15.9 Important Info about ECR Application

Developers often have administrative rights on the machine used for test and development. To allow a normal user to use the application as intended, with limited rights it's a good idea to check the application on another PC, with this limited rights or by logging on as an ordinary user.

If you are using the flxdrv.dll it's not allowed to rename it. The flxdrv.dll name is used to lookup the position to place tracefiles and to get the version information for this dll.

If you are using the PointTerminal.dll it's not allowed to rename it.

Tracefiles for the UserControl are placed in the directory defined by SetConfiguration - printer options - param2.

PointTerminalLog.txt contains UserControl specific trace information.

PointWareExpedientLog.txt contains PointWareExpedient specific trace information.

flxComTrace.txt contain information about the communication between ECR and Terminal.

PtFxDTr@XXXX.txt Low level trace information.

Avoid Virus scanning of files used by the DLL/UserControl, if the Virus scanner locks a file during a transaction, the transaction might be rejected, take longer or cause other hard to find errors.

Receipt text received by the ECR must be printed without delay. Customer is expected to receive a receipt at specific times in the transaction flow and this must be printed at once. The flow of the receipt text is:

- Saved in the ECR database
- Print
- ECR reply is send to the terminal indicating receipt saved and printed OK or not.

Make sure your application is none blocking. This can be done with threading.

## 15.10 Error Codes – Nice to Know

If the terminal encounters an error according to the OTRS specification, this error is passed through to your application.

Verifone Denmark has added some extra codes starting at 0x10000 = 65536.

### **0x1632 – Bitmap error**

If this error occurs during PSAM installation it indicates, that the PSAM type is wrong. A FTD PSAM is used instead of a KOPI PSAM or the reverse. To correct, either change the PSAM or request an update of the terminals TLCMDB for the PSAM used. If Nets has service on the KOPI environment this error could be the result as well.

**0xFFFF2 – Handler timeout**, on a new terminal the problem might be caused by an incorrect TLCMDB entry. Contact Verifone Denmark with terminal number 99XXXX and let Verifone Denmark verify, that the setup is correct. In Verifone Denmark's terminal DBS, Communication must be set to NET.

If communication is set to NET-NetsIFS and must be so, let Nets check the setting of integrators public IP with Nets positive list for IFS.

**0xFFFF3 – Handler error**, a general error, but during start-up of a new terminal it is often: PSAM not installed.

**0xFFFF6 – Handler insufficient**. Terminal can't reach Nets during PSAM installation, check network setup. PSAM must be in slot 2. Make a terminal report using OPU menu 5 3 and check the PSAM ADRESSE:, if it says 3 it's normally enough to switch power for 10 seconds and then try menu 5 3 once more, otherwise move the PSAM to another slot.

**0xFF24 – No card present**, the PSAM is not installed.

**0xFFFF – General error**. Given this error during PSAM installation indicate the PSAM type is wrong. A FTD PSAM is used instead of a KOPI PSAM or the reverse. To correct, either change the PSAM or request an update of the terminals TLCMDB for the PSAM used. If Nets has service on the KOPI environment this error could be the result as well. If the PSAM has not been used for a long time, Nets might have closed the PSAM, contact Nets TC-vagten to reopen PSAM.



## **15.11 Interpretation of other Errors**

Other errors must be found in the individual documentation OTRS, TAPA, EMV, FlexDriver, Point-Terminal etc.

## 15.12 'Systemfejl' – Errors not listed in OTRS

- 5000 - Money left on terminal
- 5001 - Unable to read batch data
- 5002 - Unable to generate new OTRSLOG\_LOGFILE
- 5003 - Nets - rejected an advice
- 5004 - Failed to remove a token from the terminal.

## **15.13 Handling of Different Transaction Types**

The following paragraphs will describe how to handle different transaction types.

### **15.13.1 PIN Purchase Transactions**

PIN based transactions can be initiated from either the terminal by inserting/swiping a card, or from the ECR, by sending purchase information to the terminal. If initiated from the terminal, the user must enter the PIN code and wait for the amount from the ECR. If initiated from the ECR, the user is requested to insert/swipe a card, enter a PIN code and press 'Godkend' (OK).

### **15.13.2 Signature Purchase Transactions**

Signature based transactions can be initiated from either the terminal by inserting/swiping a card, or from the ECR, by sending purchase information to the terminal. After initiating the transaction the user is requested to insert/swipe a card and approve the amount. This transaction type generates two receipts, unless host rejects the transaction or a communication error occurs. In this case only 1 receipt is generated. In case of successful host communication and verification, one receipt is generated, by the terminal and immediately sent to the ECR. The receipt must be printed, the user must sign and the operator must verify the signature. The second receipt is then generated based on the operator's decision (signature accepted/rejected).

### **15.13.3 Offline Purchase Transactions**

If contact to Nets' host system is not possible, some card issuers allow an offline transaction to be made. Contact Nets in order to get an authorization code, to be entered as part of the start up of the offline transaction. Otherwise an offline transaction may be handled like signature purchase – see above or as an offline pin purchase.

### **15.13.4 Refund Transactions**

After ECR initiating the transaction the user is requested to insert/swipe a card (if not already done) and then to approve the amount. This transaction type generates two receipts, unless the host rejects the transaction or a communication error occurs. In this case only 1 receipt is generated. In case of successful host communication and verification, the two receipts are generated by the terminal and immediately send to the ECR. Both receipts must be printed, and the operator must sign the customer receipt.

## 15.14 Terminal Messages – in Danish and English

Afbryder	Disconnecting
Afstemning	End of day
Afventer kort	Waiting for card
Afventer PIN/Beløb	Waiting for PIN/Amount
Afvist	Rejected
Arbejder	Working
Beløb for højt	Amount too high
Beløb	Amount
Check underskrift	Check the signature
Extra	Gratuity
Forkert pin	Wrong pin
Forretning	Merchant
Godkendt	Approved
Indlæs kort	Swipe card
Ingen kvittering	No receipt
Kan ikke anvendes	The card cannot be used
Kommunikationsfejl	Communication error
Kortholder	Cardholder
Kort isat korrekt	Card inserted correct
Kvittering udskrives	Receipt is being printed
Købet er afbrudt	The purchase is aborted
Lukket	Closed
Modtager	Receiving
Prøver igen	Reconnecting
Retur	Refund
Ring	Phone
Sender	Sending
Slet Alt	Delete All
Slet	Delete
Spærret inddrag	Blocked card – Withdraw card
Systemfejl	System error
Tast PIN	Enter PIN
Teknisk fejl	Technical error
Terminalen er klar	The terminal is ready
Udfør afstemning nu	Make an EndOfDay now
Ugyldig valuta	Invalid currency code
Ukendt kort	Unknown card
Vent	Wait

## **15.15 Common ECR Problems in Certification**

### **15.15.1 Handling of Multipart Receipts**

Multipart receipt means:

- Receipt for the merchant to keep.
- Receipt for the customer to keep.
- Receipt with error code to indicate progress.

An example of this is the signature transaction receipt.

1. A receipt is printed for the customer to sign. For the shop to keep.
2. The operator when have to validate the signature and accept or decline the signature.
3. A second receipt is printed, with the result of the operator's decision. For the customer to keep.

The refund transaction is another example giving multipart receipts.

### **15.15.2 Handling of Receipt Reprint**

A copy of a receipt must be marked with a text to indicate it is a copy of an original.

### **15.15.3 Reprint of Receipts in a given Period**

Reprint of receipts in a given period, either via ECR or BackOffice application must be possible. This is done to ensure documentation in case of terminal breakdown.

### **15.15.4 Password Protection of some Admin Functions**

This is to ensure the daily user of the ECR is unable to erase terminal transaction data by accident.

### **15.15.5 Large Transactions**

During the certification transactions are made with large amounts; ensure that it is possible to do this on the ECR. Normally 2 transactions are made to give a sum above 1.000.000,00 DKK.

## 15.16 Traces – in Case of Problems

Verifone expects your ECR application to be able to change the tracelevel, without recompilation. For the PointTerminal.dll, set tracelevel to 4 and verify you can find the:

- PointTerminalLog.txt, containing UserControl specific trace information
- And the two files mentioned below

For the FlexDriver DLL, set tracelevel to CONF\_EXTTRACE\_PLUS and verify you can find the:

- flxComTrace.txt containing information about the communication between ECR and Terminal.
- PtFxDrTr@XXXX.txt containing FlexDriver Low level trace information.

If you are using TCP/IP to communicate with the terminal, Verifone expects you know how to trace/sniff the Ethernet by using HUB/WireShark or similar equipment. The network used must be stable and without dropouts.

In the ADMIN? menu, see 15.8.1, you can enable trace "ECR TRACE" in the terminal, the trace data is sent with the next "SEND LOG" you make. Observe the trace file has limited size and trace stops if limit reached. After a "SEND LOG" the "ECR TRACE" has to be enabled again.

Inform [development.hrv@verifone.com](mailto:development.hrv@verifone.com) with terminal number and time/details about the problem you want us to investigate, after making the "SEND LOG".