



Sale to POI Protocol Specifications

**Retailer Protocols
Working Group**



**Protocol Version 3.0
Document Version 3.0
3 October 2016**

© 2016 nexo AISBL All rights reserved.

This information is protected by international intellectual property laws and its use is governed by the applicable
End-User license

Revision History

Version	Date	Object
0.90	17 Nov. 2009	Draft for external consultation
1.0	10 Oct. 2010	Version 1.0
1.9	15 Feb. 2013	Draft for external consultation
2.0	02 April 2013	Version 2.0
2.9	04 May 2015	Draft for external consultation
3.0	3 October 2016	Version 3.0

Table of Contents

1 INTRODUCTION.....	23
1.1 PURPOSE OF THE DOCUMENT	23
1.2 CONTENT OF THE DOCUMENT	24
1.3 HOW TO READ THE DOCUMENT.....	28
1.4 NOTATIONS.....	29
1.4.1 <i>State Diagram Notations</i>	29
1.5 ABBREVIATIONS.....	30
1.6 REVISION HISTORY – SALES TO POI PROTOCOL SPECIFICATIONS - v 3.0.....	31
2 ARCHITECTURES AND MODELS	32
2.1 ACTORS AND ROLES OF THE RETAILER PROTOCOL	32
2.1.1 <i>Customer</i>	32
2.1.2 <i>Cardholder</i>	32
2.1.3 <i>Cashier</i>	32
2.1.4 <i>Site Manager</i>	32
2.2 ENVIRONMENTS.....	33
2.2.1 <i>Environment of the POI System</i>	33
2.2.2 <i>Retailer Environment</i>	34
2.2.3 <i>Acquirer Environment</i>	34
2.2.4 <i>TMS Environment</i>	35
2.3 IMPLEMENTATION HISTORY AND ARCHITECTURE TRENDS	36
2.3.1 <i>Stand-Beside Systems</i>	36
2.3.2 <i>Integrated Systems</i>	37
2.4 ARCHITECTURE MODELS.....	38
2.4.1 <i>Sale System Architecture</i>	38
2.4.2 <i>POI System Architecture</i>	39
2.4.2.1 Standalone POI	40
2.4.2.2 Connected POI	41
2.4.2.3 Clustered POI System	42
2.4.2.4 Distributed POI System.....	44
2.5 SYSTEM COMPONENTS IDENTIFICATION AND RELATIONSHIP	45
2.5.1 <i>Identification of Systems and Components</i>	45
2.5.2 <i>Multiple Sale or POI Systems</i>	47
2.5.3 <i>Logical Connections Between System Components</i>	48
2.5.4 <i>Logical Connections Between Terminals</i>	50
2.5.4.1 Connected POI Architecture	50
2.5.4.2 Terminal to Terminal Transport Connections.....	51
2.5.4.3 Terminal to Server Transport Connections	51
2.5.4.4 Server to Server Transport Connections	52
2.5.4.5 Server to Terminal Transport Connections	52

2.5.5	<i>Logical Connections Between Servers</i>	53
2.5.5.1	Connected POI Architecture	53
2.5.5.2	Direct Server to Server Logical Connections	54
2.5.5.3	Indirect Server to Server Logical Connections.....	56
2.6	CONFIGURATION AND EXAMPLES.....	57
2.6.1	<i>Architecture Configuration Parameters</i>	57
3	PROTOCOL MANAGEMENT	59
3.1	GENERAL ORGANISATION	59
3.2	TRANSPORT SERVICES MANAGEMENT	61
3.2.1	<i>Transport Connection Handling</i>	61
3.2.1.1	General Rules.....	61
3.2.1.2	Standard Sequence Flow.....	62
3.2.2	<i>Data Transfer</i>	64
3.2.2.1	General Rules.....	64
3.2.2.2	Standard Processing Flow.....	64
3.2.3	<i>Addressing</i>	65
3.2.4	<i>Transport Error Handling</i>	66
3.2.4.1	Sale System Unable to Establish or receive a Transport Connection (ERTR01).....	66
3.2.4.2	Transport Connection Broken (ERTR02).....	66
3.2.4.3	Unable to Send a Message (ERTR03)	67
3.2.4.4	Message Too Big (ERTR04).....	67
3.2.4.5	Late Arrival (ERTR05)	68
3.2.4.6	Max Global Number of Connections (ERTR06).....	68
3.2.4.7	Incomplete Application Message (ERTR07)	69
3.2.4.8	Other Errors	69
3.2.5	<i>State Diagrams</i>	70
3.2.5.1	Sale System.....	70
3.2.5.2	POI System.....	71
3.2.6	<i>Transport Service Configuration Parameters</i>	72
3.3	TRANSPORT PROTOCOLS.....	74
3.3.1	<i>Introduction</i>	74
3.3.2	<i>TCP Protocol</i>	75
3.3.2.1	Typical Use	75
3.3.2.2	Message Delimitation	76
3.3.2.3	Transport Connection Handling	77
3.3.2.4	Addressing.....	78
3.3.3	<i>HTTP Protocol</i>	79
3.3.3.1	Introduction	79
3.3.3.2	Transport Connection Handling	81
3.3.3.3	Application Dialogues	82
3.3.3.4	Request Start Line	88
3.3.3.5	Error Handling and Response Start Line	89
3.3.3.6	HTTP General Header Fields	90
3.3.3.7	HTTP Request Header Fields.....	91
3.3.3.8	HTTP Response Header Fields	92
3.3.3.9	HTTP Entity Header Fields	93
3.3.3.10	Addressing.....	94

3.3.4	<i>HTTPS Protocol</i>	95
3.3.4.1	Introduction	95
3.3.4.2	Application Dialogues	96
3.3.4.3	HTTPS Constraints	98
3.3.4.4	TLS Services	99
3.3.4.5	TLS Certificates	104
3.3.5	<i>Sale to POI Protocol over TLS</i>	107
3.3.5.1	Introduction	107
3.3.5.2	Transport Service Constraints	108
3.3.5.3	Application Dialogues	109
3.3.5.4	TLS Services and Certificates.....	109
3.4	MESSAGES MANAGEMENT	110
3.4.1	<i>Messages Organisation</i>	111
3.4.1.1	Message Coding and Decoding	111
3.4.1.2	Message Structure	113
3.4.1.3	Messages Typology	114
3.4.1.4	Message Header Content	116
3.4.1.5	Message Body Response	118
3.4.1.6	JSON Messages Coding/Decoding	119
3.4.2	<i>Dialogue Management</i>	122
3.4.2.1	Type of Messages Exchange	122
3.4.2.2	Type of Dialogues	124
3.4.2.3	Service Dialogue	124
3.4.2.4	Error Resolution Dialogue	128
3.4.2.5	Device Dialogue	129
3.4.2.6	Notification Dialogue	131
3.4.2.7	State Diagrams	132
3.4.3	<i>Protocol Version Management</i>	134
3.4.4	<i>Origin and Destination of Messages</i>	137
3.4.5	<i>Identification of Message Exchanges</i>	138
3.4.6	<i>Message Security</i>	140
3.4.6.1	CMS Principles	140
3.4.6.2	Cryptographic Mechanisms	141
3.4.6.3	Key Sets	141
3.4.6.4	Key Encryption	142
3.4.6.5	Data Encryption	143
3.4.6.6	MAC Protection	145
4	APPLICATION PROTOCOL SPECIFICATIONS	146
4.1	OVERVIEW	149
4.1.1	<i>Presentation of the Interface</i>	149
4.1.2	<i>Profiles</i>	152
4.2	SYSTEMS SYNCHRONISATION.....	156
4.2.1	<i>Session Management</i>	156
4.2.2	<i>Interface State Diagram</i>	158
4.2.3	<i>Login</i>	160
4.2.3.1	Login Processing	160
4.2.3.2	Presentation of the Messages	164

4.2.3.3	Login Request Layout.....	166
4.2.3.4	Login Response Layout.....	167
4.2.3.5	Error Cases.....	169
4.2.3.6	Example.....	170
4.2.4	<i>Logout</i>	172
4.2.4.1	Logout Processing	172
4.2.4.2	Presentation of the Messages	172
4.2.4.3	Logout Request Layout.....	173
4.2.4.4	Logout Response Layout.....	173
4.2.4.5	Error Cases.....	173
4.2.4.6	Example.....	175
4.2.5	<i>Enable Service Messages</i>	176
4.2.5.1	Presentation of the Messages	176
4.2.5.2	Enable Service Request Layout.....	177
4.2.5.3	Enable Service Response Layout.....	178
4.2.5.4	Enable Service Processing	179
4.2.5.5	Error Cases.....	183
4.2.5.6	Example.....	184
4.2.6	<i>Admin</i>	185
4.2.6.1	Admin Processing	185
4.2.6.2	Presentation of the Messages	187
4.2.6.3	Admin Request Layout	188
4.2.6.4	Admin Response Layout.....	188
4.2.6.5	Error Cases.....	189
4.2.6.6	Example.....	190
4.3	<i>FINANCIAL SERVICES</i>	191
4.3.1	<i>Standard Payment</i>	191
4.3.1.1	Transaction Identification	191
4.3.1.2	Presentation of the Payment Messages	193
4.3.1.3	Payment Request Layout.....	196
4.3.1.4	Payment Response Layout.....	200
4.3.1.5	Standard Payment Transaction	205
4.3.1.6	Error Cases.....	210
4.3.1.7	Examples	212
4.3.2	<i>Other Payment Services</i>	221
4.3.2.1	Cash Back	221
4.3.2.2	Split Payment.....	223
4.3.2.3	Deferred Sale.....	226
4.3.2.4	Reservation.....	229
4.3.2.5	Recurring	235
4.3.2.6	Tip.....	235
4.3.2.7	Aggregation	236
4.3.2.8	Merchant Instalment	237
4.3.2.9	Issuer Instalment	238
4.3.2.10	Voice Authorization	239
4.3.2.11	Refund	240
4.3.2.12	Cash Advance	242
4.3.2.13	Currency Conversion	243
4.3.2.14	Tokens	245
4.3.2.15	Customer Orders	249

4.3.3	<i>Card Acquisition</i>	255
4.3.3.1	Presentation of the Card Acquisition Messages	255
4.3.3.2	Card Acquisition Request Layout.....	257
4.3.3.3	Card Acquisition Response Layout.....	258
4.3.3.4	Card Acquisition and Transaction Processing	261
4.3.3.5	Error Cases.....	264
4.3.3.6	Example.....	265
4.3.4	<i>Loyalty Services</i>	268
4.3.4.1	Loyalty Services Overview.....	268
4.3.4.2	Presentation of the Loyalty Messages	271
4.3.4.3	Presentation of the Loyalty Part of the Payment Messages	273
4.3.4.4	Loyalty Request Layout	274
4.3.4.5	Loyalty Response Layout.....	276
4.3.4.6	Loyalty Processing.....	278
4.3.4.7	Error Cases.....	281
4.3.4.8	Examples	282
4.3.5	<i>Stored Value Messages</i>	289
4.3.5.1	Presentation of the Messages	289
4.3.5.2	Stored Value Request Layout.....	291
4.3.5.3	Stored Value Response Layout	293
4.3.5.4	Stored Value Processing	295
4.3.5.5	Error Cases.....	296
4.3.5.6	Example.....	297
4.3.6	<i>Reversal Messages</i>	298
4.3.6.1	Overview of Abort, Reversal and Refund.....	298
4.3.6.2	Presentation of the Messages	300
4.3.6.3	Reversal Request Layout.....	301
4.3.6.4	Reversal Response Layout.....	303
4.3.6.5	Reversal Processing	305
4.3.6.6	Error Cases.....	308
4.3.6.7	Example.....	309
4.3.7	<i>Batch Messages</i>	312
4.3.7.1	Presentation of the Batch Messages	312
4.3.7.2	Batch Request Layout	313
4.3.7.3	Batch Response Layout.....	314
4.3.7.4	Batch Processing.....	315
4.3.7.5	Example.....	319
4.3.8	<i>Payment Use Cases</i>	321
4.3.8.1	Pay at the Table.....	321
4.4	<i>ADMINISTRATIVE SERVICES</i>	325
4.4.1	<i>Reconciliation Messages</i>	325
4.4.1.1	Presentation of the Messages	325
4.4.1.2	Reconciliation Request Layout	327
4.4.1.3	Reconciliation Response Layout.....	328
4.4.1.4	Reconciliation Processing	330
4.4.1.5	Error Cases.....	337
4.4.1.6	Example.....	338
4.4.2	<i>Get Totals Messages</i>	340
4.4.2.1	Presentation of the Messages	340
4.4.2.2	GetTotals Request Layout	342

4.4.2.3	GetTotals Response Layout	343
4.4.2.4	Get Totals Processing	344
4.4.2.5	Error Cases.....	345
4.4.2.6	Example.....	346
4.4.3	<i>Balance Inquiry Messages</i>	348
4.4.3.1	Presentation of the Messages	348
4.4.3.2	Balance Inquiry Request Layout	350
4.4.3.3	Balance Inquiry Response Layout	352
4.4.3.4	Balance Processing	355
4.4.3.5	Error Cases.....	356
4.4.3.6	Example.....	357
4.5	DEVICES SERVICES	358
4.5.1	<i>Output Content Formats</i>	358
4.5.2	<i>Display</i>	362
4.5.2.1	Processing Overview	362
4.5.2.2	Presentation of the Display Messages	364
4.5.2.3	Display Request Layout.....	365
4.5.2.4	Display Response Layout.....	366
4.5.2.5	Display Processing	367
4.5.2.6	Error Cases.....	369
4.5.2.7	Example.....	370
4.5.3	<i>Input</i>	371
4.5.3.1	Processing Overview	371
4.5.3.2	Presentation of the Input Messages	376
4.5.3.3	Input Request Layout.....	378
4.5.3.4	Input Response Layout.....	381
4.5.3.5	Input Processing	382
4.5.3.6	Error Cases.....	388
4.5.3.7	Examples	390
4.5.4	<i>Input Update</i>	394
4.5.4.1	Presentation of the Input Update Message.....	394
4.5.4.2	Input Update Layout	395
4.5.4.3	Input Update Processing.....	397
4.5.4.4	Example.....	400
4.5.5	<i>Print</i>	401
4.5.5.1	Processing Overview	401
4.5.5.2	Presentation of the Print Messages	402
4.5.5.3	Print Request Layout	403
4.5.5.4	Print Response Layout	404
4.5.5.5	Print Processing	405
4.5.5.6	Error Cases.....	408
4.5.5.7	Examples	409
4.5.6	<i>Sound</i>	410
4.5.6.1	Usage Overview	410
4.5.6.2	Presentation of the Sound Messages	411
4.5.6.3	Sound Request Layout	412
4.5.6.4	Sound Response Layout	413
4.5.6.5	Sound Processing.....	414
4.5.6.6	Error Cases.....	416
4.5.6.7	Example.....	417

4.5.7	<i>PIN</i>	419
4.5.7.1	PIN Processing	419
4.5.7.2	Presentation of the PIN Messages	420
4.5.7.3	PIN Request Layout	421
4.5.7.4	PIN Response Layout	422
4.5.7.5	Error Cases	423
4.5.7.6	Examples	425
4.5.8	<i>Card Reader</i>	426
4.5.8.1	Processing Overview	426
4.5.8.2	Presentation of the Card Reader Messages	428
4.5.8.3	Card Reader Init Request Layout	430
4.5.8.4	Card Reader Init Response Layout	431
4.5.8.5	Card Reader APDU Request Layout	432
4.5.8.6	Card Reader APDU Response Layout	432
4.5.8.7	Card Reader Power-Off Request Layout	433
4.5.8.8	Card Reader Power-Off Response Layout	434
4.5.8.9	Card Reader Processing	435
4.5.8.10	Error Cases	437
4.5.8.11	Examples	439
4.5.9	<i>Transmit</i>	443
4.5.9.1	Presentation of the Transmit Messages	443
4.5.9.2	Transmit Request Layout	444
4.5.9.3	Transmit Response Layout	444
4.5.9.4	Transmit Processing	445
4.5.9.5	Error Cases	448
4.5.9.6	Example	449
4.6	<i>ERROR REPORTING AND ERROR CASES</i>	450
4.6.1	<i>Outcome of Message Processing</i>	450
4.6.2	<i>Message Format Errors</i>	453
4.6.2.1	Message Format	453
4.6.2.1.1	General Parsing Error	453
4.6.2.1.2	Mandatory Data Absent	453
4.6.2.1.3	Unexpected Data	453
4.6.2.1.4	Invalid Number of Repetitions	453
4.6.2.1.5	Invalid Order of Repetitions	453
4.6.2.1.6	Unexpected Value	454
4.6.2.1.7	Invalid Value for the Type	454
4.6.2.1.8	Invalid Enumerated Value	454
4.6.2.1.9	Repeated Message	454
4.6.2.1.10	Data Size	454
4.6.2.1.11	Empty Cluster	454
4.6.2.1.12	Unacceptable Value Combination	455
4.6.2.1.13	Unknown Data	455
4.6.3	<i>Hardware Error</i>	456
4.6.3.1	DeviceOut Error	456
4.6.3.1.1	POI Temporary Unavailable	456
4.6.3.1.2	POI Permanently Unavailable	456
4.6.3.1.3	Device Temporary Out	456
4.6.3.1.4	Device Permanently Out	456
4.6.3.1.5	Maintenance	456
4.6.3.1.6	Security Alarm	456
4.6.3.2	UnavailableDevice Error	457
4.6.4	<i>Invalid Request</i>	458
4.6.4.1	NotAllowed Error	458
4.6.4.1.1	Forbidden Dialogue	458
4.6.4.1.2	Forbidden Combination of Service	458
4.6.4.1.3	Forbidden Message	458

4.6.4.1.4	NotAllowed Value.....	458
4.6.4.1.5	Completed Transaction.....	458
4.6.4.1.6	Invalid Payment Reservation.....	458
4.6.4.1.7	Forbidden CardReader Sequence.....	459
4.6.4.1.8	Forbidden CardReader APDU Request.....	459
4.6.4.2	LoggedOut Error	459
4.6.4.3	Unavailable Service Error	460
4.6.4.3.1	Too Old Protocol Version.....	460
4.6.4.3.2	Unavailable Service for the Card.....	460
4.6.4.3.3	Unavailable Administrative Service	460
4.6.4.3.4	Unavailable Device Service	460
4.6.4.3.5	Unavailable Display Format.....	460
4.6.4.3.6	Unavailable Input Command.....	460
4.6.4.3.7	Unavailable Printing Mode	461
4.6.4.3.8	Unavailable PIN Verification Method.....	461
4.6.4.3.9	Unavailable Sound Format.....	461
4.6.4.4	NotFound Error.....	462
4.6.4.4.1	Transaction Not Found.....	462
4.6.4.4.2	Message Not Found.....	462
4.6.4.4.3	Reconciliation Not Found	462
4.6.4.4.4	Key Reference Not Found.....	462
4.6.4.4.5	Predefined Message Reference Not Found.....	462
4.6.4.4.6	Language Not Supported	462
4.6.4.4.7	Card Removed.....	462
4.6.5	<i>Unable to Perform</i>	463
4.6.5.1	Busy Error.....	463
4.6.5.1.1	Component Unavailable.....	463
4.6.5.1.2	POI Busy	463
4.6.5.1.3	Device Busy.....	463
4.6.5.2	InProgress Error	463
4.6.5.2.1	Uncompleted Transaction	463
4.6.5.3	InsertedCard Error	463
4.6.6	<i>Stopped</i>	464
4.6.6.1	Aborted Error	464
4.6.6.2	Cancel Error.....	464
4.6.6.2.1	User Cancellation.....	464
4.6.6.2.2	System Cancellation	464
4.6.7	<i>Incorrect Card</i>	465
4.6.7.1	InvalidCard Error.....	465
4.6.7.1.1	No Card Entered.....	465
4.6.7.1.2	Unreadable Card	465
4.6.7.1.3	Unknown Card.....	465
4.6.7.1.4	Invalid Card	465
4.6.7.2	WrongPIN Error	465
4.6.8	<i>Transaction Declined</i>	466
4.6.8.1	UnreachableHost Error	466
4.6.8.1.1	Host Unreachable	466
4.6.8.1.2	No Host Answer	466
4.6.8.2	Refusal Error.....	466
4.6.8.2.1	Acquirer Decline	466
4.6.8.2.2	Local Decline	466
4.6.8.3	PaymentRestriction Error	467
4.7	<i>ERROR MANAGEMENT</i>	468
4.7.1	<i>Transaction Status Messages</i>	468
4.7.1.1	Presentation of the Messages	468
4.7.1.2	TransactionStatus Request Layout.....	469
4.7.1.3	TransactionStatus Response Layout.....	470
4.7.1.4	Transaction Status Processing	471
4.7.1.5	Error Cases.....	474
4.7.1.6	Example.....	475
4.7.2	<i>Abort Message</i>	477

4.7.2.1	Presentation of the Abort Message.....	477
4.7.2.2	Abort Request Layout	478
4.7.2.3	Abort Processing	479
4.7.2.4	Error Cases.....	482
4.7.2.5	Examples	483
4.7.3	<i>Event Notification Message</i>	485
4.7.3.1	Presentation of the Event Notification Message	485
4.7.3.2	Event Notification Layout.....	486
4.7.3.3	Event Notification Processing	487
4.7.3.4	Example.....	490
4.7.4	<i>Diagnosis Messages</i>	491
4.7.4.1	Presentation of the Messages	491
4.7.4.2	Diagnosis Request Layout.....	492
4.7.4.3	Diagnosis Response Layout.....	493
4.7.4.4	Diagnosis Processing	494
4.7.4.5	Error Cases.....	495
4.7.4.6	Example.....	496
4.7.5	<i>Error Resolution and Error Situations</i>	497
4.7.5.1	Error Resolutions Specifications on the POI System	497
4.7.5.2	Error Resolutions Specifications on the Sale System	499
4.7.5.3	Application Timers	501
4.7.5.4	Loss of Payment Response	503
4.7.5.5	Interrogation After Timeout	504
4.7.5.6	Abort on Timeout	505
4.7.5.7	Transport Connection Broken	506
4.7.5.8	POI Terminal Crash During a Session	508
4.7.5.9	User Cancellation During a Pending Device Request.....	509
5	DATA DEFINITION	510
5.1	MESSAGES DEFINITION.....	510
5.1.1	<i>Request Message</i>	510
5.1.2	<i>Response Message</i>	512
5.2	DATA DICTIONARY	513
5.2.1	<i>Introduction</i>	513
5.2.1.1	Data Names.....	514
5.2.1.2	Data Definition	514
5.2.1.3	References	514
5.2.1.4	Usage	515
5.2.1.5	Type.....	515
5.2.1.6	Format	516
5.2.1.7	CrossRef.....	516
5.2.1.8	XMLCoding	516
5.2.1.9	ASN1Coding	516
5.2.1.10	Enumeration or Cluster Labels	517
5.2.1.11	Data Structure Definition.....	518
5.2.2	<i>Data Elements and Structures</i>	519
5.2.2.1	A	519
5.2.2.2	B-C	534
5.2.2.3	D	563
5.2.2.4	E-F	572

5.2.2.5	G-I.....	586
5.2.2.6	J-L.....	605
5.2.2.7	M-N.....	621
5.2.2.8	O.....	638
5.2.2.9	P	648
5.2.2.10	Q-R.....	676
5.2.2.11	S	692
5.2.2.12	T	719
5.2.2.13	U-Z.....	737
5.2.3	<i>Data Types</i>	743
6	ANNEX A CONFIGURATION PARAMETERS	746
6.1	SALE SYSTEM PARAMETERS	746
6.2	POI SYSTEM PARAMETERS.....	751
7	ANNEX B MESSAGES EXAMPLES.....	756
7.1	TRANSPORTED MESSAGE EXAMPLES	757
7.1.1	<i>XML and JSON Coding of the Message</i>	757
7.1.2	<i>TCP Transport of the Message</i>	759
7.1.3	<i>HTTP Transport of the Message</i>	760
7.2	KEY ENCRYPTION EXAMPLE.....	763
7.3	DATA ENCRYPTION EXAMPLE	764
7.4	MAC PROTECTION EXAMPLE	766
7.5	MESSAGE EXAMPLES IN XML.....	775
7.5.1	<i>LoginRequest - LoginResponse</i>	775
7.5.2	<i>LogoutRequest LogoutResponse</i>	777
7.5.3	<i>EnableServiceRequest EnableServiceResponse</i>	777
7.5.4	<i>AdminRequest AdminResponse</i>	778
7.5.5	<i>PaymentRequest PaymentResponse</i>	779
7.5.5.1	Simple Payment.....	779
7.5.5.2	Protected Card Data and MAC	780
7.5.5.3	Payment with Products	782
7.5.5.4	Payment with a Local Card.....	783
7.5.6	<i>CardAcquisitionRequest CardAcquisitionResponse</i>	784
7.5.7	<i>Loyalty Services</i>	786
7.5.7.1	Loyalty with Payment.....	786
7.5.7.2	Payment with Rebates	788
7.5.7.3	Loyalty Award Refund.....	790
7.5.8	<i>StoredValueRequest StoredValueResponse</i>	792
7.5.9	<i>ReversalRequest ReversalResponse</i>	793
7.5.10	<i>BatchRequest BatchResponse</i>	795
7.5.11	<i>ReconciliationRequest ReconciliationResponse</i>	796
7.5.12	<i>GetTotalsRequest GetTotalsResponse</i>	797
7.5.13	<i>BalanceInquiryRequest BalanceInquiryResponse</i>	799
7.5.14	<i>DisplayRequest DisplayResponse</i>	800

7.5.15	<i>InputRequest InputResponse</i>	801
7.5.15.1	Display Confirmation	801
7.5.15.2	Information Request	802
7.5.15.3	Menu Item Selection	803
7.5.16	<i>Input Update</i>	804
7.5.17	<i>PrintRequest PrintResponse</i>	805
7.5.18	<i>SoundRequest SoundResponse</i>	806
7.5.19	<i>PINRequest PINResponse</i>	807
7.5.20	<i>CardReaderInitRequest CardReaderInitResponse</i>	808
7.5.20.1	Magstripe Card Reading	808
7.5.20.2	ICC Card Reading	809
7.5.20.3	Card Reader APDU	810
7.5.20.4	Card Reader Power-Off	811
7.5.21	<i>TransmitRequest TransmitResponse</i>	812
7.5.22	<i>TransactionStatusRequest TransactionStatusResponse</i>	813
7.5.23	<i>AbortRequest</i>	815
7.5.24	<i>EventNotification</i>	816
7.5.25	<i>DiagnosisRequest DiagnosisResponse</i>	817
8	ANNEX C EXAMPLES OF ARCHITECTURES	818
8.1	PETROL STATION	818
8.2	SUPERMARKET AND DEPARTMENT STORE	819
8.3	CAR PARK	820
8.4	VENDING MACHINE	820
8.5	E-COMMERCE	821

Figures

Figure 1: The Sale to POI Protocol	23
Figure 2: Document Organisation	24
Figure 3: State Machine Diagram.....	29
Figure 4: Environment of the POI System.....	33
Figure 5: Retailer Environment.....	34
Figure 6: Acquirer Environment.....	34
Figure 7: Intermediary Agent.....	35
Figure 8: TMS Environment.....	35
Figure 9: Stand-Beside Systems	36
Figure 10: Integrated Systems	37
Figure 11: Sharing a Common LAN	37
Figure 12: Global Sale System Architecture	38
Figure 13: Global POI System Architectures.....	39
Figure 14: Standalone POI System.....	40
Figure 15: Connected POI Terminal.....	41
Figure 16: Clustered POI System.....	42
Figure 17: Clustered POI System with a Central Site	43
Figure 18: Distributed POI System	44
Figure 19: Identification of System Components	45
Figure 20: Example of Multiple POI Systems.....	47
Figure 21: Relationships between System Components	48
Figure 22: Specific Relationships	49
Figure 23: Connected POI Terminal Architecture	50
Figure 24: Terminal to Terminal Links.....	51
Figure 25: Terminal to Server Links	51
Figure 26: Server to Server Links.....	52
Figure 27: Server to Terminal Links	52
Figure 28: Connected POI Architecture	53
Figure 29: Server to Server Link for Clustered POI	54
Figure 30: Server to Server Link, Terminal to Server case	54
Figure 31: Server to Server Link, Server to Server case	55
Figure 32: Case 2.4 – Server to Server Link, Server to Terminal case	55
Figure 33: Server to Server Link through Terminal Connections	56
Figure 34: Server to Server Link through Sale Terminal Connection	56
Figure 35: Server to Server Link through POI Terminal Connection	56
Figure 36: Protocols Organisation	59
Figure 37: Transport Adaptation Layer.....	60

Figure 38: Sale Initiated Transport Connection Standard Sequence Flow	62
Figure 39: POI Initiated Transport Connection Standard Sequence Flow	63
Figure 40: Transport Address.....	65
Figure 41: Late Arrival Error Example	68
Figure 42: Crossing of Disconnection and Connection	69
Figure 43: Sale System Transport Connection Management State Diagram	70
Figure 44: POI System Transport Connection Management State Diagram	71
Figure 45: TCP Protocol	75
Figure 46: Peer-to-peer TCP Transport Protocol	75
Figure 47: Gateway TCP Transport Protocol	75
Figure 48: Local TCP Transport Protocol.....	75
Figure 49: Message Header Length	76
Figure 50: HTTP Protocol.....	79
Figure 51: Basic HTTP Communication	79
Figure 52: HTTP Request/response Chain	79
Figure 53: HTTP Message General Format	80
Figure 54: HTTP Header Fields	80
Figure 55: Standard HTTP Sequence Flow	81
Figure 56: HTTP Service Dialogue	82
Figure 57: HTTP Service with Device Requests in Parallel	83
Figure 58: HTTP Service and Device Request without Response	84
Figure 59: HTTP Error Dialogue.....	85
Figure 60: HTTP Abort of an Unknown Request.....	86
Figure 61: HTTP Error Dialogue.....	86
Figure 62: HTTP Device Dialogue.....	87
Figure 63: HTTP Exchange for a Linked Notification Dialogue.....	87
Figure 64: HTTP Exchange for an Isolated Notification Dialogue.....	87
Figure 65: HTTP Request Start Line	88
Figure 66: HTTP Response Start Line	89
Figure 67: HTTPS Protocol	95
Figure 68: Standard HTTPS Sequence Flow	96
Figure 69: HTTPS Service Dialogue	97
Figure 70: TLS Protocol Layers.....	99
Figure 71: TLS Record Protocol	100
Figure 72: New TLS Session (Full Handshake)	101
Figure 73: Resume Previous TLS Session	102
Figure 74: Force the End of a TLS Session	102
Figure 75: X.509 Certificate Format	104
Figure 76: Certificate Path.....	105
Figure 77: PKI Topologies	105

Figure 78: Sale to POI Protocol over SSL/TLS	107
Figure 79: Standard Sequence Flow	109
Figure 80: Sale to POI Message Structure	113
Figure 81: Message Header Information	116
Figure 82: Request/Response Message Exchange	122
Figure 83: Request Only Message Exchange	122
Figure 84: Unsolicited Notification Message Exchange	123
Figure 85: Service Dialogue Sequence Flow	125
Figure 86: Service with Device Requests in Parallel	126
Figure 87: Service Dialogue with Device Request without Response	126
Figure 88: POI Sends a Device Request outside a Service Dialogue	127
Figure 89: Error Resolution Dialogue Sequence Flow	128
Figure 90: Device Dialogue Sequence Flow	129
Figure 91: Notification Dialogue Sequence Flow	131
Figure 92: Sale System Dialogues Management State Diagram	132
Figure 93: POI System Dialogues Management State Diagram	133
Figure 94: Exchange of Protocol Version	134
Figure 95: Required Exchange of Protocol Version	136
Figure 96: Service Dialogue Messages Identification	139
Figure 97: Device Dialogue Messages Identification	139
Figure 98 : CMS Data Protection in Retailer Protocol Messages	140
Figure 99 : Encryption/Decryption of a Session Key	142
Figure 100 : Data Encryption Process	143
Figure 101 : Data Decryption Process	144
Figure 102 : MAC Computation Process	145
Figure 103: Presentation of the Interface	149
Figure 104: Functional Profile	152
Figure 105: Session Between two Components (Servers or Terminals)	156
Figure 106: Sale to POI Interface State Diagram	158
Figure 107: Login Exchange	160
Figure 108: Login without Logout	161
Figure 109: Login Retry	161
Figure 110: Logout after no Login Response	162
Figure 111: Double Login Processing	162
Figure 112: Login Request/Response Information	164
Figure 113: Logout Exchange	172
Figure 114: Logout Request/Response Information	172
Figure 115: Enable Service Request/Response Information	176
Figure 116: Standard Swipe-Ahead Completion	179
Figure 117: Swipe-Ahead Abort	180

Figure 118: Swipe-Ahead Repetition.....	180
Figure 119: EnableService on a Transaction in Progress	181
Figure 120: Close a Completed Transaction	181
Figure 121: Swipe-Ahead Aborted by a Device or Service Request	182
Figure 122: Admin Standard Process	185
Figure 123: Admin Request/Response Information	187
Figure 124: Daily Payment Report Example	190
Figure 125: Transaction Identifiers	191
Figure 126: Sale and POI Transactions	192
Figure 127: Payment Request Information.....	194
Figure 128: Payment Response Information.....	195
Figure 129: Standard Payment Exchange	207
Figure 130: Cash Back Data Elements	221
Figure 131: Split Payment Data Elements	223
Figure 132: Typical Sequence of Split Payments	225
Figure 133: Deferred Sale Data Elements	226
Figure 134: Successful Deferred Sale Sequence Flow.....	227
Figure 135: Unsuccessful Deferred Sale Sequence Flow.....	227
Figure 136: First Reservation Data Elements	229
Figure 137: Update Reservation Data Elements.....	230
Figure 138: Payment after Reservation Data Elements	231
Figure 139: Reservation Transaction State Diagram	232
Figure 140: Cancelled Reservation Sequence Flow	233
Figure 141: Tip Data Elements	235
Figure 142: Issuer Instalment.....	238
Figure 143: Payment Refund Data Elements.....	240
Figure 144: Cash Advance Data Elements	242
Figure 145: Currency Conversion Data Elements.....	244
Figure 146: Issuer Token.....	245
Figure 147: Payment Token Data Elements	246
Figure 148: Real-Time Parking Using Token	247
Figure 149: Click and Collect Steps	249
Figure 150: Customer Order Data Elements	251
Figure 151: Click and Collect Process Flow Example.....	253
Figure 152: CardAcquisition Information	255
Figure 153: CardAcquisition Link Data Elements.....	261
Figure 154: Card Acquisition Sequence Flow Example	261
Figure 155: Card Acquisition Sequence Flow Example	262
Figure 156: Sale and POI Systems Loyalty Services Management.....	268
Figure 157: POI Identification of a Loyalty Account	269

Figure 158: Example of POI Loyalty Services Management.....	269
Figure 159: Loyalty Request Information	271
Figure 160: Loyalty Response Information.....	272
Figure 161: Loyalty Components of the Payment Messages.....	273
Figure 162: Stored Value Request Information	289
Figure 163: Stored Value Response Information	290
Figure 164: Sale System Payment Cancellation	298
Figure 165: POI System Payment Cancellation	298
Figure 166: Sale Payment Annulation.....	299
Figure 167: Reversal Request/Response Information	300
Figure 168: Partial Reversal Example	306
Figure 169: Reversal without Identification	307
Figure 170: Batch Information	312
Figure 171: Retrieval of Transactions Performed without Sale System.....	315
Figure 172: Sending of Transactions to be Performed Later	315
Figure 173: Batch Request of Transaction Processing	316
Figure 174: Batch Request of Transaction Results.....	316
Figure 175: Pay at the Table Steps.....	321
Figure 176: Transaction Selection to Pay at the Table	322
Figure 177: Invalid Transaction Selection to Pay at the Table	322
Figure 178: Login After Sale Wake-up	323
Figure 179: Use Case Pay-at-the-table	323
Figure 180: Reconciliation Request Information	325
Figure 181: Reconciliation Response Information	326
Figure 182: Type of Reconciliation Requests.....	330
Figure 183: Reconciliation Periods Overlapping	331
Figure 184: Reconciliation between Sale System and POI System	332
Figure 185: Reconciliation between Sale System and POI System	333
Figure 186: Overlapping of Reconciliation Periods	333
Figure 187: Get Totals Request Information	340
Figure 188: Get Totals Response Information	341
Figure 189: Balance Inquiry Request Information	348
Figure 190: Balance Inquiry Response Information	349
Figure 191: Format of Information to Display or Print	358
Figure 192: Barcode Display Example	361
Figure 193: Display Devices and Information Qualify	362
Figure 194: Display Request/Response Information.....	364
Figure 195: Display Commands Relationships	367
Figure 196: Minimum Display Time Management.....	368
Figure 197: Input Messages Mechanisms.....	371

Figure 198: Display/Input Devices and Information Qualify	373
Figure 199: Other Types of Customer Interface Environments	375
Figure 200: Input Request/Response Information	376
Figure 201: GetAnyKey Input Command Responses	382
Figure 202: Length Constraint for TString, DigitString and DecimalString	383
Figure 203: StringMask for Formatted String Input	383
Figure 204: MaxInputTime Limitation	384
Figure 205: Inserted Card Notification.....	385
Figure 206: Menu Display Example	386
Figure 207: Input Update Information.....	394
Figure 208: Input Update Sequence Flow	397
Figure 209: Too Late Input Update	397
Figure 210: Incorrect Input Message Reference	398
Figure 211: Language Input Update.....	399
Figure 212: Printer Location	401
Figure 213: Print Request/Response Information	402
Figure 214: Print Response Message Modes	405
Figure 215: Print Processing Sequence Example	406
Figure 216: Sound Usage.....	410
Figure 217: Sound Request/Response Information	411
Figure 218: Sound Response Mode.....	414
Figure 219: PIN Exchange Overview	419
Figure 220: PIN Request/Response Information	420
Figure 221: Card Reader Exchanges Overview	426
Figure 222: Card Reader Init Request/Response Information	428
Figure 223: Card Reader APDU Request/Response Information.....	429
Figure 224: Card Reader Power-Off Request/Response Information	429
Figure 225: Card Reader Device State Diagram.....	435
Figure 226: Transmit Information	443
Figure 227: Transmit Exchange in a MPos Architecture	445
Figure 228: Outcome of Message Processing	450
Figure 229: ErrorCondition - AdditionalResponse	452
Figure 230: TransactionStatus Request/Response Information	468
Figure 231: Transaction Status Typical Exchange.....	471
Figure 232: Duplicate Payment Receipt with Transaction Status	473
Figure 233: Abort Request Information	477
Figure 234: Standard Processing of an Abort	480
Figure 235: Abort Processing Impossible.....	480
Figure 236: Abort of a Terminated Processing.....	481
Figure 237: Abort Response Message Crossing	481

Figure 238: Event Notification Information	485
Figure 239: StopAssistance Event	488
Figure 240: Abort Event.....	488
Figure 241: Diagnosis Request/Response Information.....	491
Figure 242: Error Occurring on a POI System	497
Figure 243: Service Request Processing Error Situation.....	497
Figure 244: Device Request Sent to the Sale System Error Situations	498
Figure 245: Error Situation outside a Service Request	498
Figure 246: Error Occurring on a Sale System	499
Figure 247: Rearming the Timer on Device requests	501
Figure 248: Loss of Payment Response	503
Figure 249: Timeout on Waiting Payment Response.....	504
Figure 250: Abort after Payment Response Timeout.....	505
Figure 251: Transport Connection Broken before Payment Response	506
Figure 252: POI Terminal Crash During a Session.....	508
Figure 253: User Cancellation during a Pending Device Request.....	509
Figure 254: Data Definition Attributes	513
Figure 255: Enumeration or Cluster Labels.....	517
Figure 256: Data Structure Definition	518
Figure 257: Logical Architecture of a Petrol Station	818
Figure 258: Distributed POI in Petrol Station	819
Figure 259: Car Park Architecture	820
Figure 260: e-Commerce Architecture	821
Figure 261: Cardholder Centralised Architecture	821

Tables

Table 1: Transport Connection Sale System States	70
Table 2: Transport Connection Sale System Events	70
Table 3: Transport Connection POI System States	71
Table 4: Transport Connection POI System Events	71
Table 5: Message Header Data Elements	116
Table 6: Message Header Presence Condition for a Request.....	117
Table 7: Message Header Presence and Value Condition for a Response.....	117
Table 8: Response Data Element	118
Table 9: Response Presence Condition.....	118
Table 10: EPAS to JSON Data Type Conversion	120
Table 11: Dialogues Management Sale System States	132
Table 12: Dialogues Management Sale System Events	132
Table 13: Dialogues Management POI System States	133
Table 14: Dialogues Management POI System Events	133
Table 15: Generic and Services Profiles per Messages	154
Table 16: Reconciliation Transaction Counting	334
Table 17: Parameters Validity per Input Command	382
Table 18: Response Presence Condition.....	450

Configuration Parameters

Configuration 1: Sale and POI Systems Identification	57
Configuration 2: Terminals Identification and Logical Connections	57
Configuration 3: Servers Identification and Logical Connections.....	58
Configuration 4: Transport Service.....	72
Configuration 5: Sale Transport Service	72
Configuration 6: Transport Connection	73
Configuration 7: Sale and POI Protocol Version	135
Configuration 8: Sale and POI Terminal Profiles.....	155
Configuration 9: Server Login.....	163
Configuration 10: Time Synchronisation	163
Configuration 11: Reconciliation Details.....	336
Configuration 12: Event Notification Outside Sessions	489

1 Introduction

1.1 Purpose of the Document

This document contains the specifications of the protocol between a *Sale System*¹ and a *POI System*, one of the Retailer's Protocols also called *Sale to POI* protocol.



Figure 1: The Sale to POI Protocol

This is the first activated interface during a payment, or more generally any kind of electronic transaction using a card or other account identifier. The protocol allows a *Sale System* requesting a *POI System* to process the payment of purchases made by a customer.

Purpose is to present the detailed specifications of the protocol on several aspects:

- The modelisation of the various architectures that could be supported by the protocol.
- The way to use the transport protocol located below the application protocol that is specified.
- The management of the messages, including the dynamic behaviour of the protocol with the error management.
- The specification of the services offered by the protocol, and the specification of how to use these services, by the exchange of sequence of messages.
- The static part of the protocol, definition of the messages and their components.

To implement the protocol, these specifications have to be completed by:

- The definitions needed for the coding of the messages:
 - The XML/Schema definition of the messages which are presented in the four following files:

EpasSaleToPOIMessages.xsd	messages of the protocol
EpasCMSTypes.xsd	data structures used for the security
EpasCommonTypes.xsd	types common to the EPAS protocols
 - The ASN.1 modules which defined the protocol:

EpasSaleToPOIMessages.asn	messages of the protocol (PDU)
EpasCMSTypes.asn	data structures used for the security
- If the implementation covers some protection of data at the application level, the security mechanisms and their data definition are presented in a dedicated data dictionary, completed by a document regarding test data samples.

¹ Also called Selling System

1.2 Content of the Document

The document is separated into four main parts:

- (a) *Modelling* (chapter 2 Architectures and Models),
- (b) *Communication level management* (chapter 3 Protocol Management),
- (c) *Application protocol* (chapter 4 Application Protocol Specifications),
- (d) *Definition of messages* (chapter 5 Data Definition).

They are completed by an introduction (*chapter 1 Introduction*) and three annexes (*Annex A Configuration Parameters*, *Annex B Messages Examples* and *Annex C Examples of Architectures*).

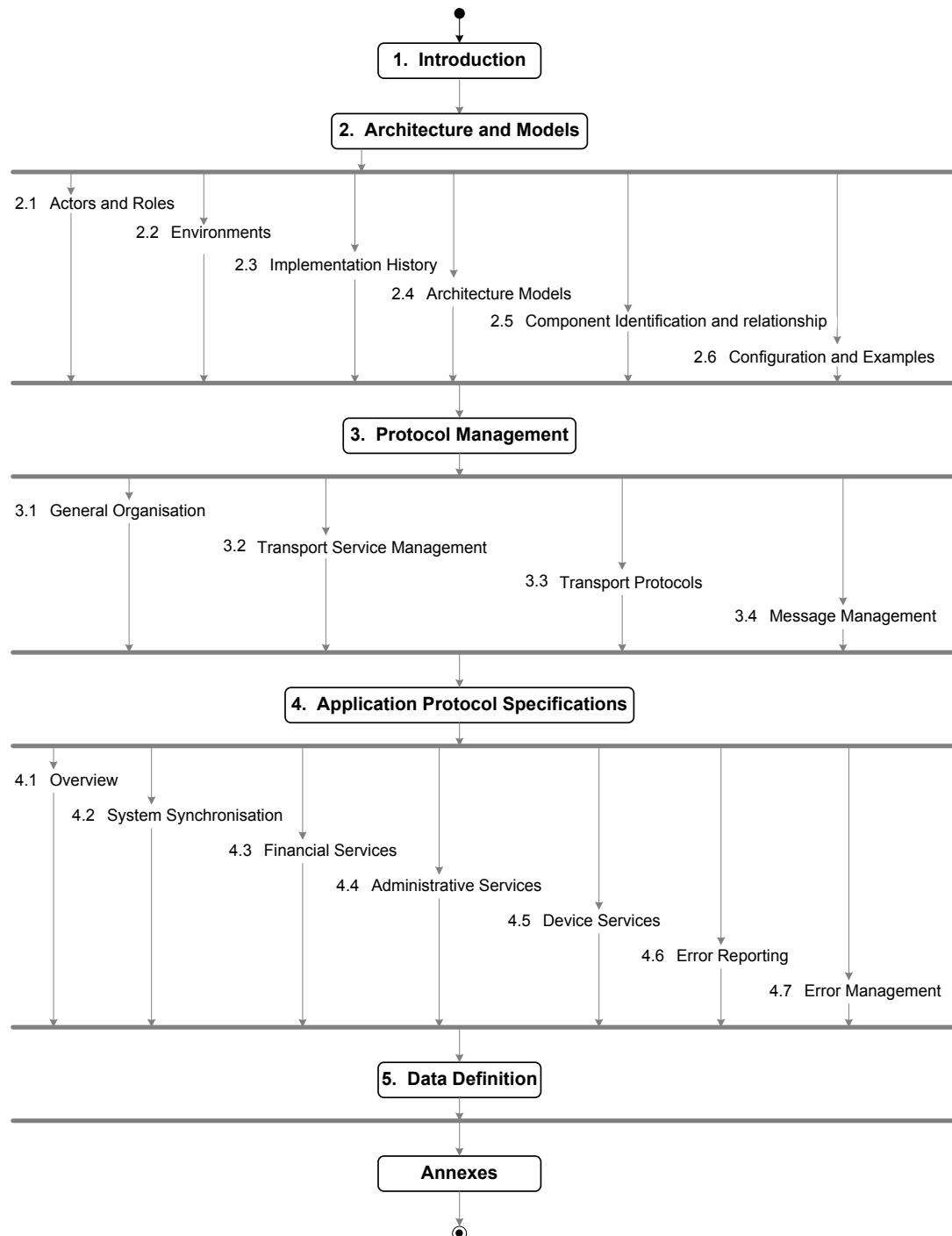


Figure 2: Document Organisation

Chapter 2, *Architectures and Models*, starts by a quick presentation of the main actors around the protocol.

The section 2.2, *Environments*, explains the environments of the systems involved directly or not, by the protocol. The section helps to understand the scope of the protocol.

The section 2.3, *Implementation History and Architecture Trends*, introduces a brief history of implementations of cashless payment in the retail market. This history is presented around the two main trends for the implementation. The section helps to understand the modeling explained in the following section.

The section 2.4, *Architecture Models*, presents the four models of architecture considered for implementing the Sale To POI protocol.

The last section of the chapter, section 2.5 *System Components Identification and Relationship*, describes on a practical way how the protocol uses these models of architecture. It first defines how the protocol identifies components of the systems. Then it presents the possible relationships (logical connections) between these components to dialog between systems and implement the communication aspect of the protocol. The section ends by the configuration parameters used to implement these models.

Chapter 3, *Protocol Management*, is devoted to the communication level management.

It starts by reminding the protocol organization in levels, focused on the application protocol this document is specifying.

The section 3.2, *Transport Services Management*, contains the transport services required by the application protocol, and specifies how to use them, including the error management of the transport service level. The section is completed by the state diagrams of the two systems, and their configuration parameters.

The section 3.3, *Transport Protocols*, lists the transport protocols which are selected to support the Sale to POI application protocol, and define their use in order to provide the transport services required by the application level. The transport protocols that can be used include the TCP protocol , the HTTP protocol, the HTTPS protocol with HTTP over SLL/TLS, and SLL/TLS supporting directly the application level .

The section 3.4, *Messages Management*, specifies the management of the messages, with the information located in the Message Header:

- The section starts with the section 3.4.1, *Messages Organisation* which presents the structures of messages, their coding and decoding, the types of messages, and describes the content of the Message Header and the common information in response message reporting the result of the request processing. The end of the section defines the rules for the JSON coding decoding.
- Then the section 3.4.2, *Dialogue Management* presents all the type of exchanges and dialogs depending on type of messages and initiator of the dialog. It also includes general state diagrams to specify the behavior of each system.
- The section is completed by the specification of:
 - The management of protocol versions,
 - The identification of the origin and the destination of the messages,
 - The identification of message exchanges.
 - The security principles and mechanisms to protect the messages

Chapter 4, *Application Protocol Specifications*, starts with a presentation which provides a summary of the interface in terms of messages and functionalities, including implementation profiles.

The section 4.2, *Systems Synchronisation*, specifies the part of administrative services related to the synchronization of systems. It describes the management of session between systems, specifies the synchronization messages Login, Logout and Enable Service, completed by a global state diagram of the synchronization, and ends with and Admin.

The section 4.3, *Financial Services*, specifies all the financial services:

- The section starts with the section 4.3.1, *Standard Payment* which presents the payment messages, and specifies the standard payment service. It also specifies the identification of transactions.
- The section 4.3.2, *Other Payment Services* presents all the extensions of service for a payment transaction: cash-back, split payment, deferred sale, reservation, installment, recurring, tip, referral, refund and cash-advance. For each of the services, the additional data of the payment messages required by the service are presented, followed by the specification of the service.
- The section 4.3.4, *Loyalty Services* presents the loyalty part of the financial service. Loyalty service could be provided either inside a payment message for synchronization with the payment transaction, or with a dedicated loyalty message. The section presents an overview of the loyalty services, the loyalty message, the loyalty data of the payment messages, and the specification of the loyalty.
- The other sections present:
 - The card acquisition messages, allowing the reading and the analysis of the card used by the customer, before the process of the transaction.
 - The stored value messages to activate, load or reload stored value cards.
 - The reversal messages to process a manual reversal of a financial transaction.
 - The batch messages to process transactions without real time contact with the Sale system, and to get the result of these transactions.

The section 4.4, *Administrative Services*, specifies the part of administrative services related to the financial services. The section describes:

- The reconciliation messages to exchange the totals of the transactions during a certain period,
- The balance inquiry of an account attached to a payment, loyalty or stored value card;
- The get totals messages, to exchange the actual totals of the transactions since the beginning of a period.

The section 4.5, *Devices Services*, specifies the low level services. First the section describes the format of information to display or print, which is used by most of the device services. Then the section specifies the protocol for each of the following devices services: card reader display, input, input update, print, PIN, sound and transmit. Most of these services defines and specifies first the involved logical devices, presents the services, describe the related messages and specify the processing of the various related services.

The section 4.6, *Error Reporting and Error Cases*, presents all the error cases of the protocol by category, and the standard framework to report the details of errors or warnings.

The section 4.7, *Error Management*, presents first the messages of the protocol that are used for error resolution: transaction status, abort, event notification and diagnosis. Then, the section 4.7.5, *Error Resolution* and Error Situations, presents first the general specification of error resolution on the two sides of the protocol. This section is completed by a set of typical error situations with the way to resolve these errors.

Chapter 5, *Data Definition*, provides a detailed description of the messages of the Sale to POI protocol.

It starts by presenting the global format of the Sale to POI protocol messages.

The section 5.2, *Data Dictionary*, contains the data dictionary of the protocol, which provides exhaustive definitions of all the messages and their components. An introduction presents the organization of the data dictionary entries and the used notation.

All the entries are listed by their name in alphabetical order, and most of the references to the name of an entry contain a link to a corresponding entry in the data dictionary.

At the end of the document, annexes contain:

- The configuration parameters defined in the specifications,
- Examples of messages:
 - An example of message carried out by the transport protocol below the Sale to POI application protocol,
 - Examples of secured messages with the allowed cryptographic algorithms, including test data with intermediate results,
 - All the examples of messages presented in the specifications, are encoded in XML.
- Examples of architectures for the Sale and the POI systems.

1.3 How to Read the Document

The document may have several levels of reading, depending on needs of the reader. Some suggestions are proposed in this section.

Quick Overview

The section 4.1.1, *Presentation of the Interface*, is sufficient to have a first overview of the protocol, summarised in a few pages.

More Details on the Application Protocol

A first approach which provides more details on the application protocol in the chapter 4 could be the reading of:

- Introduction of sections, or
- The sub-section explaining the processing of a service requested by a message.

The complete reading of the chapter 4, or the section related to the set of messages that need to be implemented, provides all the details of the application protocol.

The reading of the chapter 4 could benefit from the review of some entries of the data dictionary, using the direct links associated to the data name in the chapter 4.

It could also be preceded by some part of the chapter 2, *Architectures and Models*, at least the section 2.4, *Architecture Models*, to better understand the wording used by the specifications.

Design and Implementation of the Protocol

To design an implementation of the protocol, it is required to:

- Start with the chapter 2, *Architectures and Models*,
- Continue with the chapter 3, *Protocol Management*, at least with the section 3.4, *Messages Management*,
- Study the chapter 4, *Application Protocol Specifications*, with the help of the chapter 5, *Data Definition*.

Communication Implementation

The low level communication protocol implementation requires the reading of:

- The section 2.5 *System Components Identification and Relationship*,
- The section 3.3, *Transport Protocols*, and 3.2, *Transport Services Management*,

Error Management

A particular effort has been provided to define error cases, error reporting, error resolution procedures, which are developed in the section 4.6, *Error Reporting and Error Cases*, and 4.7, *Error Management*.

Examples of Messages

Some help can be provided to the messages coding or security with:

- The section 7.5 *Message Examples in XML* for the XML/Schema data coding,
- For the security, the section 7.2, *Key Encryption Example*, to 7.4, *MAC Protection Example*,

1.4 Notations

1.4.1 State Diagram Notations

Inside the specifications, many behaviours are specified with the help of state diagrams using the UML 2 notation. A State Diagram drawing includes:

- States showing the behavior of the Initiator and the Recipient. States include at least one initial state and one final state.
- Transitions between states, which provide:
 - The particular event (or list of events) which occurs,
 - The condition to change state when this event arrives, and
 - The list of actions that are executed during the change of state.

The notation used is: `event [condition]/action`, only the `event` is mandatory.

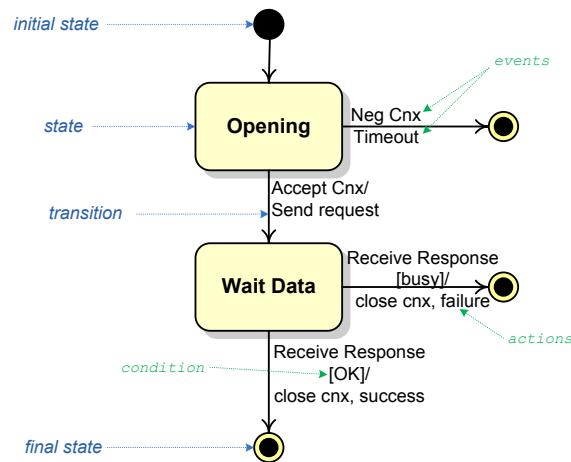


Figure 3: State Machine Diagram

A state diagram drawing is completed by the explanation of the various states, the events. When the drawing is a simplified representation of the behavior to stay understandable, a state table is provided.

A State Table is a table where the states are the columns and the events the lines. The cells contain the processing to achieve on the column state when the line event occurs, possibly depending on conditions, with the state to go to after the processing.

1.5 Abbreviations

APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation 1
CAT	Cardholder activated Terminal
ECR	Electronic Cash Register
EMV	EuroPay, MasterCard, Visa (originator of EMV specifications)
EPAS	Electronic Protocol Application Software
FIPS	Federal Information Processing Standard
HSM	Hardware or Host Security Module
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ICC	Integrated Circuit Card
IP	Internet Protocol
MAC	Message Authentication Code
PAN	Primary Account Number
PCI	Payment Card Industry (PCI PIN, PCI PTS PIN Transaction Security, PCI DSS Data Security Standard)
PDU	Protocol Data Unit
PED	PIN Entry Device
PIN	Personal Identification Number
PKI	Public Key Infrastructure
POI	Point of Interaction used to process a card payment in order to perform a financial transaction (sometimes called an EFT-POS, or POS, or payment terminal/system)
POS	Point Of Sales or Point Of Services
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TMS	Terminal Management System
WP	Work Package
XML	Extensible Mark-up Language
XHTML	XML HTML

1.6 Revision History – Sales to POI Protocol Specifications - v 3.0

Transport Service Management:

- Addition of the POI Initiated Transport Connection Sequence Flow.
- Addition of HTTP and HTTPS.
- Support of TLS protocols.

New messages:

- Batch to preload sales transactions and get outcome of a list of sales transactions.
- Sound device services.
- Input Update device services.
- Transmit device services.

New services and Use Cases:

- Tokenization.
- Instalments.
- Customer Order.
- Pay at the table.

Protocol improvement and maintenance.

2 Architectures and Models

Architecture is fundamental to be successful on the design of any hardware or software product. But the design of a protocol between two systems needs to be as far as possible independent of any implementation of the systems that use it.

The purpose of this chapter is to clearly define the environment of the retailer protocol and to provide model of architectures on each side of the Retailer protocol.

2.1 Actors and Roles of the Retailer Protocol

2.1.1 Customer

A *Customer* is someone who acquires products or services provided by the Sale System.

2.1.2 Cardholder

A *Cardholder* refers to a customer using a Card to pay or more generally to perform electronic transactions.

2.1.3 Cashier

A *Cashier* refers to the attendant of the merchant who performs electronic transactions using the Sale and the POI systems. The *Cashier* is accountable for the money in the cash drawer. This role is not mandatory, for instance in case of unattended sales operation. A *Cashier* is also called *Operator* in the following pages of the document.

2.1.4 Site Manager

The *Site Manager* is a special *Cashier* who is authorized to perform certain function that a standard *Cashier* cannot execute.

2.2 Environments

This section presents the environments of the systems involved directly or not, by the protocol. The section introduces briefly and defines these systems, to delimit the scope of the protocol.

2.2.1 Environment of the POI System

The POI System, which is responsible for the processing of electronic transactions², is surrounded by three other systems:

- The *Sale System*, which has in charge the management of a purchase made by a customer, which might involve a secure electronic transaction.
- The *Acquirer System*, which processes the transaction according to the rules of the card used by the cardholder for the transaction.
- The *TMS System*, which manages the hardware and software of the POI for the organisation in control of installed base of POI, maintains and downloads the configurations parameters for the processing of transactions and usable cards.

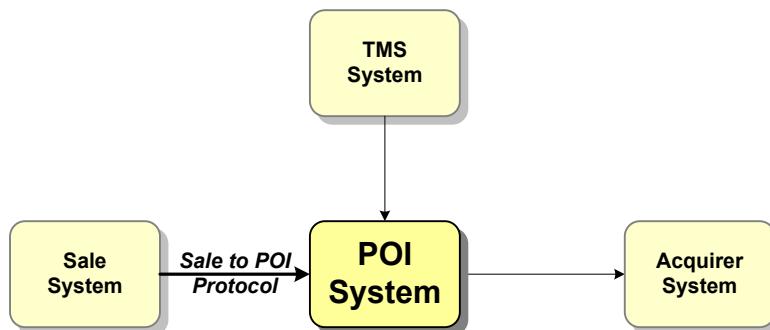


Figure 4: Environment of the POI System

In order to accept cards and realise transactions, the POI has to be initialised by the TMS System.

Processing of transactions at the Sale System, might involve processing of electronic transactions by the POI. Each electronic transaction at the POI might involve processing of a transaction on the Acquirer Host using the EPAS Acquirer Protocol, or other Application Hosts using other protocols.

During the life cycle of the POI, update or removal of hardware, software and configuration parameters are supervised by the TMS System.

² The POI system is not necessary responsible of processing all electronic transaction (e.g. local fleet card).

2.2.2 Retailer Environment

In the Store, the retailer's environment includes three systems:

- The *Sale System* is the entity in charge of the store, which provides all the standard functionalities of a point of sale, including the interface with the Delivery and the POI System. It can contain particular functions or services specific to the specialty or the context of the shop (for instance handling of the checkout in a supermarket, of the forecourt in a gas station, of the toll in a car park, of a vending machine...).
- The *Delivery System* has responsibility of delivering goods or services sold to the customer. It is always controlled by or included in the Sale System. This is also the case if the POI and the Delivery Systems are integrated in the same device, as in a pump or a vending machine.
- The *POI System* which, at Sale System's request, handles secure electronic transaction accomplished by a cardholder with an account identifier like a card. It has no direct relationship with the Delivery System.

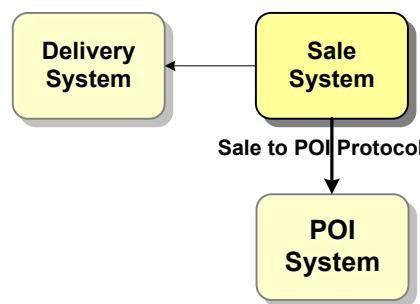


Figure 5: Retailer Environment

2.2.3 Acquirer Environment

The Acquirer is in relation with the two following systems:

- The *POI System* which performs the accepting part of electronic transactions inside the Acceptor System, which processes the sale transactions.
- The *Issuer System*, through Card Scheme Networks, to process transaction authorization and clearing. In some cases, the Acquirer and the Issuer may belong to the same entity and managed by the same host.

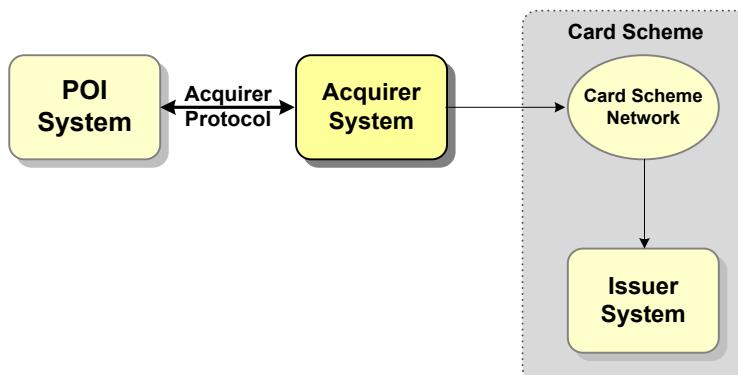


Figure 6: Acquirer Environment

Some Intermediary Agents may take place between the POI System and the Acquirer. They deliver some added value services, and can be a member located on the Acceptor domain, the Acquirer domain or both, depending on the services and responsibilities.

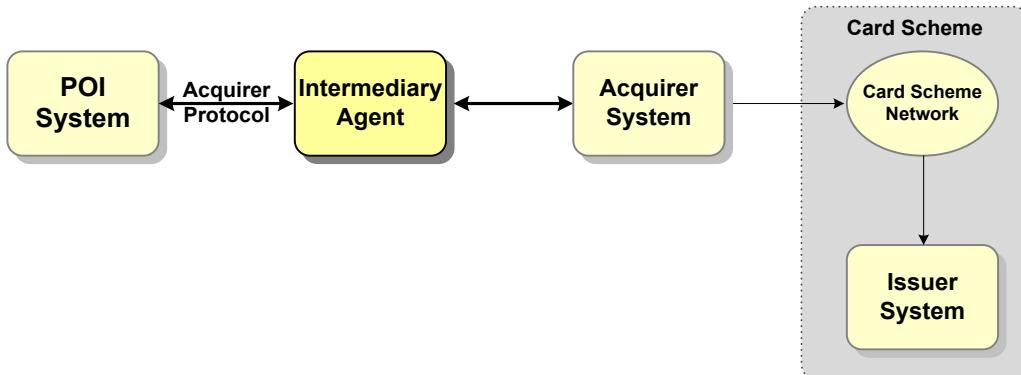


Figure 7: Intermediary Agent

2.2.4 TMS Environment

TMS services are controlled by a unique Master Terminal Manager, which has the total control of the POI System as regards to the security, applicative and maintenance services needed to manage the POI. The Master Terminal Manager can delegate part of the TMS services to one or several other Terminal Managers. Terminal Managers, including the Master can be various actors (e.g. a Manufacturer, an Acceptor, an Acquirer or a third party).

TMS services are provided by the following systems:

- A *Software Management System*, which manages the hardware related firmware, operating system and application software of the POI on behalf of the responsible entity in control of the installed base of POIs.
- A *Parameter Centre*, which stores, maintains and downloads the configuration parameters and cryptographic keys for the processing of transactions and usable cards.
- A *Maintenance and Device Management System*, for diagnosis of the POI component functionalities with limited access rights (e.g. read software versions only).

POI Systems might get the software and parameters from different Terminal Management Systems.

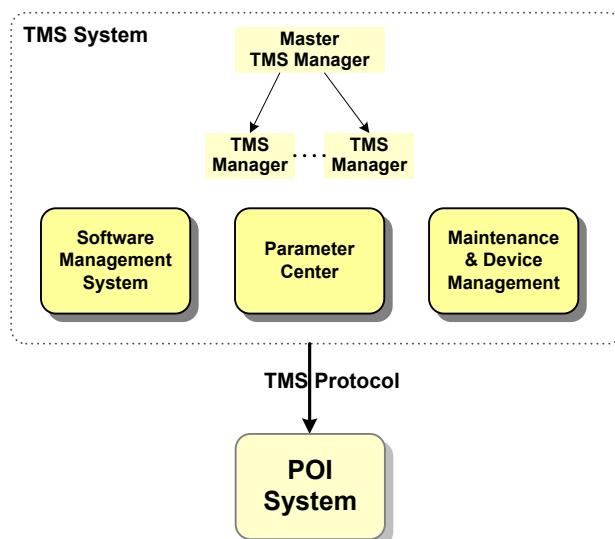


Figure 8: TMS Environment

2.3 Implementation History and Architecture Trends

Payment architectures in the retail/petrol activity are strongly correlated to the market of the sales products. Users, payment solution providers, and financial institution take care of this domain only quite recently.

Payment is only one function among others provided by a Sale software product, and whatever the architecture of the Sale System vis-à-vis POI System, payment still stays an important function of the Sale product for the management of sales amount, payment means, taxes, rebates ...

2.3.1 Stand-Beside Systems

The simplest architecture to imagine is to install a standalone payment terminal in the store to be used by one or several cashiers beside the cash registers (or ECR or POS). This was the first and the most primitive POI solution used in the multilane retail, inherited from the standalone POI architecture (see Figure 9: Stand-Beside Systems-a).

To avoid re-entering of the amount and get back some information about the payment, an elementary protocol was designed to link the payment terminal to the cash register (Figure 9: Stand-Beside Systems-b).

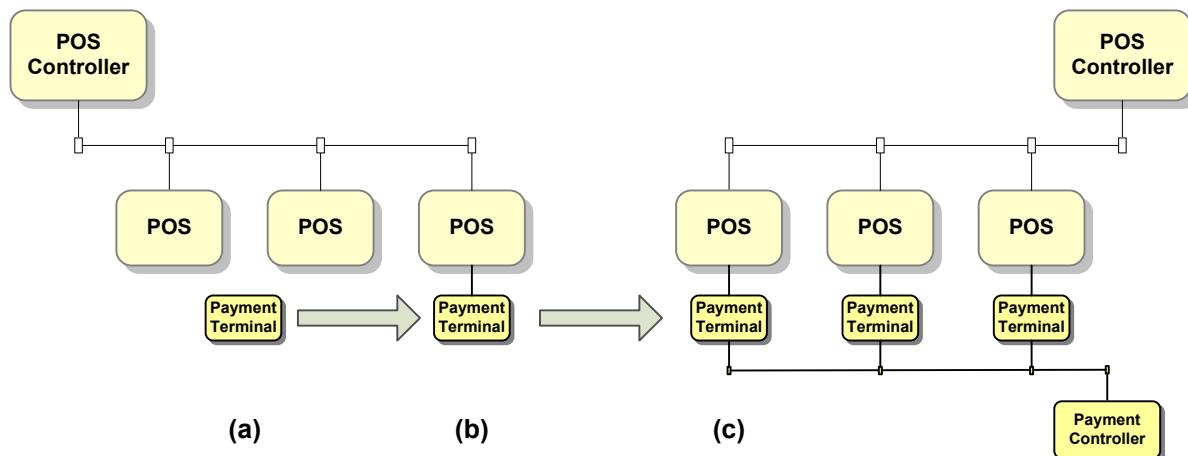


Figure 9: Stand-Beside Systems

Later the market proposed cluster architectures, made up of payment terminals linked by a primitive LAN to a gateway or a payment controller, for the communication with the Acquirer Host. Sometimes, a particular payment terminal plays the role of terminal master to contain shared resources (see Figure 9: Stand-Beside Systems-c).

To optimise payment terminal cost, receipt printer was removed and payment receipt printed by the cash register.

All these kinds of architectures belong to the Clustered POI System architecture model³.

³ The POS controller is the Sale Server, the POS are the Sale Terminals, the Payment Controller is the POI Server, and the Payment Terminals are the POI Terminals.

2.3.2 Integrated Systems

Shortly after the first stand beside payment solutions, Sale software products have included electronic payment software and a magnetic stripe card reader, device of the Sale Terminal. So the POS was completely autonomous to make payment initiated by magnetic cards, with the help of the POS controller which integrates the record of payment transactions, as the record of the sale transaction, and the communication software modules with the Acquirer Hosts (see Figure 10: Integrated Systems-a).

Some Sale products were designed to facilitate customisation of locally dependent functions, like tax or payment. So logically, because of complexity of the payment, raise of investment, and difficulty to be on time with the market, payment modules were more and more developed separately.

On the same time, to offer a PIN code verification or encryption, a pin-pad replace the magnetic card reader, connected to the POS with a protocol specific to the manufacturer and a module for this interface.

The POI is then composed of a pin-pad, a payment module in the POS, which handles the pin-pad, process the payment, and dialogue with a payment module in the POS controller (see Figure 10: Integrated Systems-b). To be independent of the Sale product, a payment server can be a substitute for the payment module in the ECR controller (see Figure 10: Integrated Systems-c).

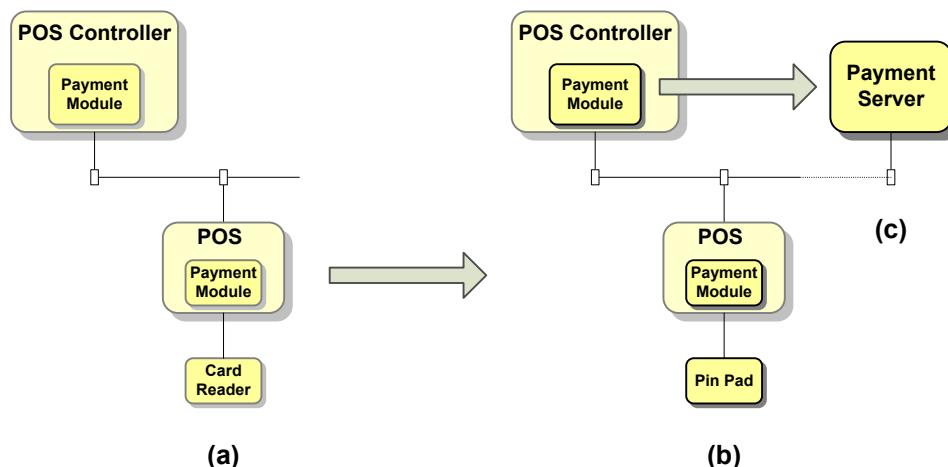


Figure 10: Integrated Systems

These architectures are a mix of Clustered POI and Distributed POI Systems models⁴.

The introduction of Ethernet LAN on which Sale System and POI System can both be connected, brings in Distributed POI configurations.

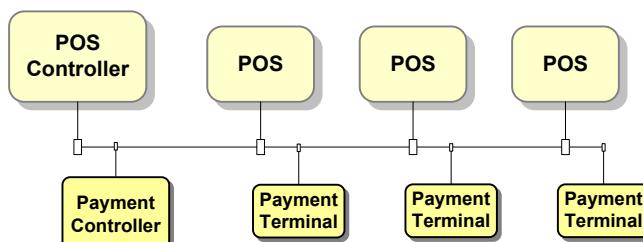


Figure 11: Sharing a Common LAN

⁴ The Sale Server is the POS controller, the Sale Terminals are the POS, the POI Server is the payment module in the POS Controller or the Payment Server, and the POI Terminals are the payment module in the POS and the Card Reader or the PIN pad, or the Payment Terminals

2.4 Architecture Models

This section describes the four models of architecture considered to implement the Sale To POI protocol.

2.4.1 Sale System Architecture

The Sale System, as seen by the Retailer Protocol's interface includes⁵:

- A set of *Sale Terminals*, which could be physical terminals (Electronic Cash Register), or logical terminals if there is no cashier (unattended transaction). The Sale Terminal manages the whole sale transaction, choice and identification of the purchased items, operations necessary to complete the transaction (payment, loyalty, and other secure operations), the interface with all the participants to transaction.
- A *Sale Server*, which processes all the global sale functions, in particular the recording of the transaction and the database of the system.

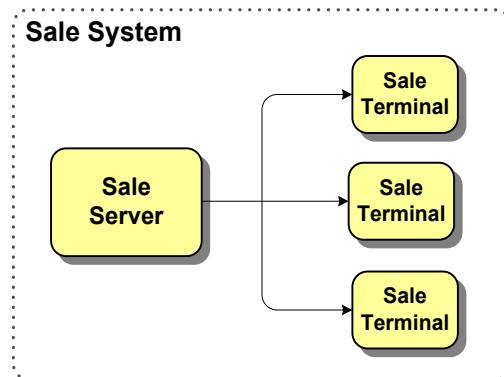


Figure 12: Global Sale System Architecture

The Sale Server and the Sale Terminals are usually joined together with a LAN, allowing communication between a terminal and the server. Traditionally, there are no communications between Sale Terminals.

The Sale Terminal handles all the physical devices required during the processing of the transaction, and delegates to the Sale Server, all the functions consuming substantial resources or functions common to the whole system.

In some contexts, i.e. if there is only one Sale Terminal, the Sale Server can be located in a single Sale Terminal.

⁵ Some examples of real and typical architecture of Sale Systems are described in the Annex C *Examples of Architectures*.

2.4.2 POI System Architecture

The POI System, as seen by the Retailer Protocol's interface includes⁶:

- A set of *POI Terminals*, a physical or logical terminal, or a pin-pad which contains at the minimum the security services, the low level services, and the user interfaces. It can also include application to process part of the transaction in cooperation with the POI Server, and some synchronisation tasks with the Sale System. In some cases, POI Terminal uses a component of the Sale System as a user interface.
- A *POI Server*, which groups or contains the global functions, as the recording of the transaction or the communication with the Acquirer Hosts or the TMS Server.

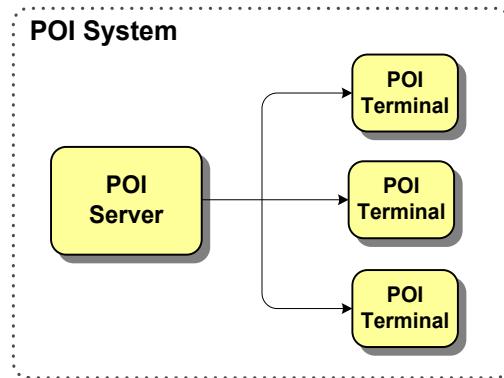


Figure 13: Global POI System Architectures

We can distinguish four classes of POI Systems, which characterize the various POI architectures that are implemented in real systems:

The *Standalone POI*, where the hardware and software components of the POI are concentrated in a single unit (the POI Terminal), without any correlation with the Sale System other than a user interface for the cashier.

The *Connected POI*, where the hardware and software components of the POI are concentrated in a single unit, connected to the Sale System through a retailer's application protocol.

The *Clustered POI System*, where the hardware and software components of the POI are split among several units (POI Terminals and POI Server), and are connected to the Sale System as a device of the Sale Terminal.

The *Distributed POI System*, where the hardware and software components of the POI are distributed among several units (POI Terminals and POI Server), and form a Payment Solution separated from the Sale System (Sale Terminal, Sale Server), with clear interface and responsibilities.

⁶ Some examples of real and typical architectures of POI Systems are described in the Annex C Examples of Architectures, and can be read before this section.

2.4.2.1 Standalone POI

The Standalone POI architecture, which is reduced to a single POI Terminal, possesses the following characteristics:

- The terminal encloses all the hardware and software components required to process card transactions and manage the POI Terminal.
- The terminal provides user interface to the cardholder for the transactions, and user interface to the cashier for the transactions and the administration of the terminal. There is no communication with the Sale System.
- The terminal manages the communication protocols with the Acquirer Hosts and the TMS Servers.

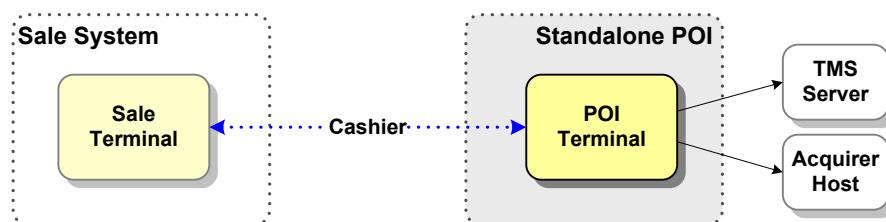


Figure 14: Standalone POI System

By definition, this type of architecture does not use any retailer's application protocol with the Sale System, which is usually composed of a unique Sale Terminal.

2.4.2.2 Connected POI

The Connected POI is a simple configuration where a POI Terminal is connected to a Sale Terminal, without any relation between POI Terminals.

The Connected POI architecture, which can also be reduced to a single POI Terminal, assumes the following characteristics:

- The terminal encloses all the hardware and software components required to process card transactions and manage the POI Terminal.
- The terminal provides user interface to the cardholder for the transactions, and user interface to the cashier for the transactions and the administration of the terminal.
- The communication between the Sale System and the POI Terminal allows performing transactions, printing receipt and status display on the Sale Terminal.
- The terminal manages the communication protocols with the Acquirer Hosts and the TMS Server.

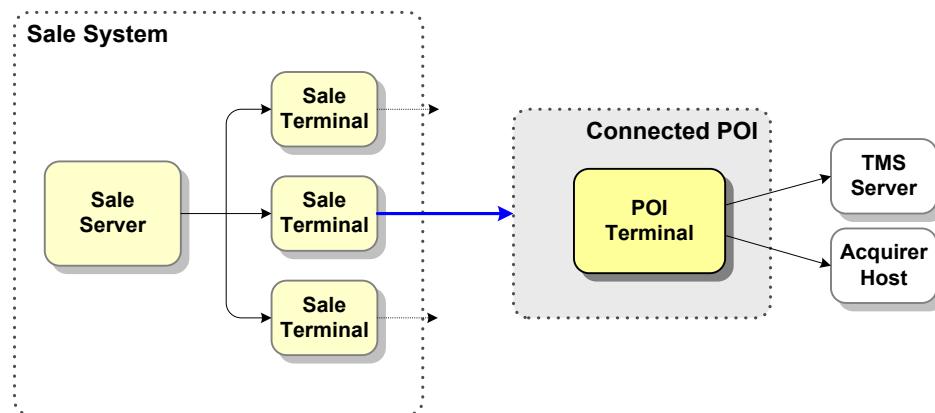


Figure 15: Connected POI Terminal

The Connected POI architecture can use the Sale to POI Retailer Protocol as the application protocol between the Sale Terminal and the corresponding POI Terminal. From the view of the POI System by the Sale System, there are as many POI Systems as POI Terminals, since each POI Terminal does not know the presence of other POI Terminals.

In the following specifications, this architecture model will be associated to the following model, the Clustered POI System, with the specific case of only one POI Terminal.

2.4.2.3 Clustered POI System

The Clustered POI architecture is characterised by the following features:

- Each Sale Terminal is physically linked to its respective POI terminal, involving an implicit addressing between the two Terminals. The Sale Terminal dialogues with an interface dedicated to the device or a POI software component implemented in the Sale Terminal to interface this POI Terminal.
- Some devices of the Sale Terminal (printer, display, scanner...) are used by the POI System during the transaction.
- Some devices of the POI Terminal (card reader, display) can be used by the Sale System to achieve sale function.
- Global functions as transaction recording and communication management with the Acquirer Hosts or the TMS Server are carried out by the POI Server inside a common platform. These global functions can be implemented inside the Sale Server, a particular Sale Terminal, or a dedicated platform.
- Viewed from the Sale system, each POI Terminal is considered as a peripheral device of the Sale Terminal, in the same respect as a magnetic stripe card reader, a scanner or a printer.

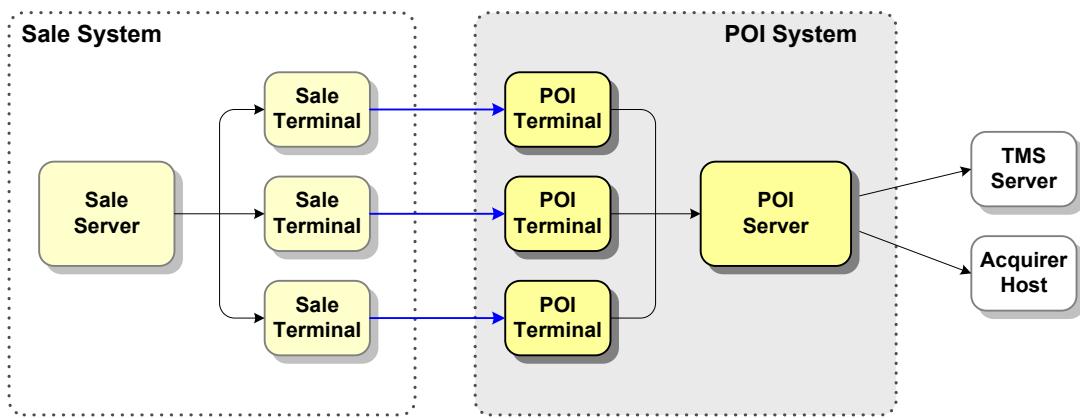


Figure 16: Clustered POI System

Implementation of the POI Server (which can only handle communication device interfaces, or manage global POI functions and located in the Sale System), depends on the configuration of the shop, number of Sale Terminals, volume of transaction, and the level of integration inside the Sale System.

In this architecture, all the global services of the Sale to POI Protocol could go through the link between the Sale Terminal and the corresponding POI Terminal, or between the Sale Server and the POI Server.

For the Clustered POI System, an alternative to the POI Server, as described in the previous figure, is the deportation of the POI Server in a central location in order to keep in the store only the POI Terminals and a communication gateway.

This POI Server manages on a *Central Site* several POI Systems where the POI Terminals are on *Local Sites* in the shops.

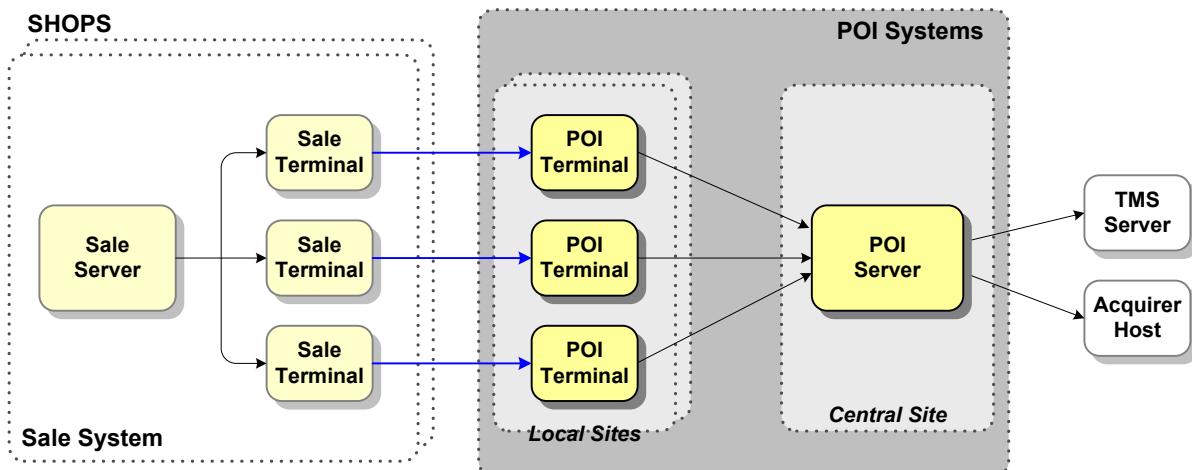


Figure 17: Clustered POI System with a Central Site

2.4.2.4 Distributed POI System

The Distributed POI architecture is characterised by the following features:

- The Sale System and the Distributed POI Systems dialogue with a standard interface in a system-to-system relationship.
- Each Sale Terminal is logically linked to its respective POI terminal, and dialogue with the Distributed POI System via the POI server. There is no implicit addressing between the Terminals, or a POI software component in the Sale Terminal to implement an interface with the POI.
- As the Clustered POI, some devices of the Sale Terminal (printer, display, scanner...) are used by the POI System during the transaction. This data flow carrying out lower level service of the standard interface is clearly identified with specific rules.
- A specific data flow of the standard interface allows the use of some devices of the POI Terminal (card reader, display) by the Sale System to achieve sale function.
- This architecture allows some flexibility for the distribution of functions between the POI Server and the POI Terminal, according to the requirements defined in this document (global functions have to be carried out by the POI Server, secure and stable functions has to be located in the POI Terminal...).

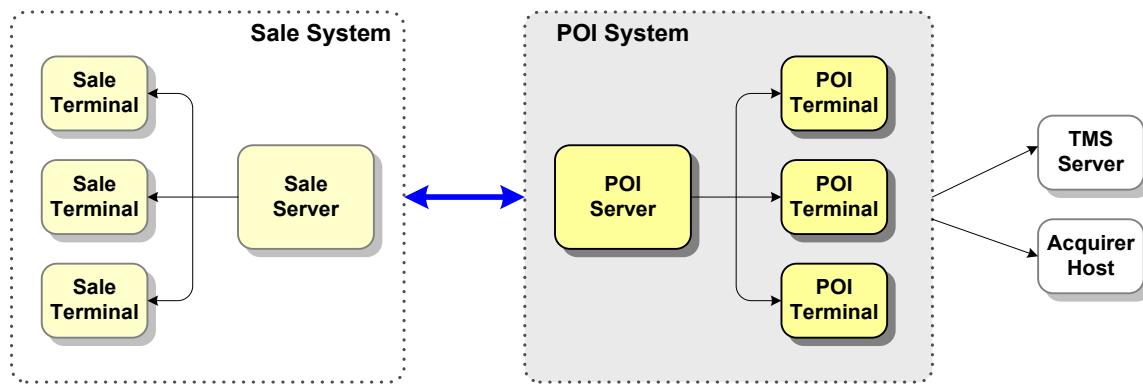


Figure 18: Distributed POI System

The standard interface is governed by some rules implied by the architecture of the POI System:

- Whatever the requester and the provider are (Server or Terminal), this is always a system-to-system interface. This is essentially a matter of implementation of the interface or the protocol.
- Visibility of the POI System: the Sale System can address a particular POI Terminal or the POI Server (i.e. the POI System) if no POI Terminal is involved in the requested service.
- Visibility of the Sale System: the POI System can address a particular Sale Terminal or the Sale Server (i.e. the Sale System) if no Sale Terminal is involved in the requested service.

2.5 System Components Identification and Relationship

The section describes in a practical way how the protocol uses the models of architecture defined in the previous section.

It first defines how the protocol identifies components of the systems.

Then it presents the possible relationships (logical connections) between these components to dialog between systems and implement the communication aspect of the protocol.

The section ends by the configuration parameters used to implement these models.

2.5.1 Identification of Systems and Components

The Sale to POI protocol is designed to interface a unique Sale System to a unique POI System.

In some particular contexts, architecture can include several POI Systems or even several Sale Systems. In this case, the various systems are considered independents, and the protocol allows the management of different systems by their system identification.

As we have seen in the various models of Sale System and POI System, on the protocol point of view, a system is composed of:

1. A *Server*, which covers hardware or software module, and assures the functions global to the System.
2. One or several *Terminals* which takes in charge part of the processing of a transaction, either because the transaction involves some hardware device, either because several transactions have to be processed in parallel and the Terminal stands for the transaction context.
3. One or several *Devices* managed and attached to each *Terminal*. These Devices are used either with a logical identification (e.g. for a Display request), either implicitly by a particular device command (e.g. CardRead request).

As already mentioned, all these system components are a logical view of the Sale and the POI Systems through the Sale to POI protocol interface. The concrete POI Server could be in a wide range of implementations, from a dedicated Point of Sale terminal in the store, to a specific software module on a Sale Terminal.

All Terminals of a given Sale or POI System could have different sets of Devices. The list of available Devices on a Sale Terminal and on a POI Terminal is exchanged at the login step between these two terminals.

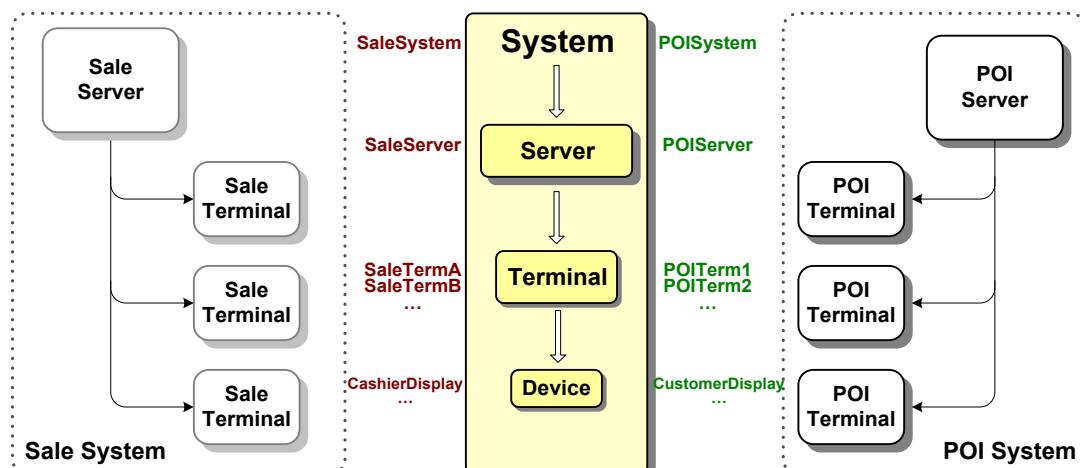


Figure 19: Identification of System Components

These components of a System are identified inside the protocol in a hierarchical way:

1. The System has an identifier, root of its components. This identifier prefixes all components identifiers of the System, and allows the recognition of a System when several Systems are managed in parallel.
2. The Server has a unique identifier, which is prefixed by the System identifier. This identifier could be used in the protocol when a message does not involve a Terminal, or is a dialogue from System to System.
3. Every Terminal has a unique logical identifier, also prefixed by the System identifier. This identifier is used in the protocol for Terminal to Terminal messages.
4. Every Device on a Terminal is identified by a logical identifier in the Terminal. This is a logical device name used in the protocol for Terminal to Terminal messages.

For the remainder of the document, when components' identifiers are mentioned or inside examples of exchanges, the following identifiers values are used:

1. The identifier of the Sale System is "*SaleSystem*", and the identifier of the POI System is "*POISystem*",
2. The identifier of the Sale Server is "*SaleServer*" (or just "*Srv*"), and the identifier of the POI Server is "*POIServer*" (or just "*srv*"),
3. Identifiers of the Sale Terminals are "*SaleTermA*", "*SaleTermB*",... (or simply "A", "B",...), and the identifier of the POI Terminals is "*POITerm1*", "*POITerm2*",... (or simply "1", "2",...),
4. Identifier of the Device's is defined by the label of the data element *Device* page 566 (e.g. "*CashierDisplay*"⁷)

⁷ "*CashierDisplay*" identifies the logical device where information is displayed for the Cashier. This logical device is generally managed by the Sale System, but could also be managed by the POI System.

2.5.2 Multiple Sale or POI Systems

In some particular context, architectures can include several POI Systems or even several Sale Systems. For instance, in a petrol station, the Sale System can manage one POI System for payment indoor and another POI System for the outdoor payment.

In this case:

- Either the various POI (or Sale) Systems are independents or cannot share any information, or cannot use common communication channels, and there are two Sale to POI protocols management. These two protocols cannot mix messages because of the different identification of each POI (or Sale) System.
- Either the various POI (or Sale) Systems can share the same application protocol by any possible implementation, there is only one POI (or Sale) System using the same system identification.

To follow the example of a petrol station, with two different POI Systems for indoor and outdoor payment, the figure below shows the case of two independent POI Systems with two different assigned system identifications (*IndoorPOISystem* and *OutdoorPOISystem*).

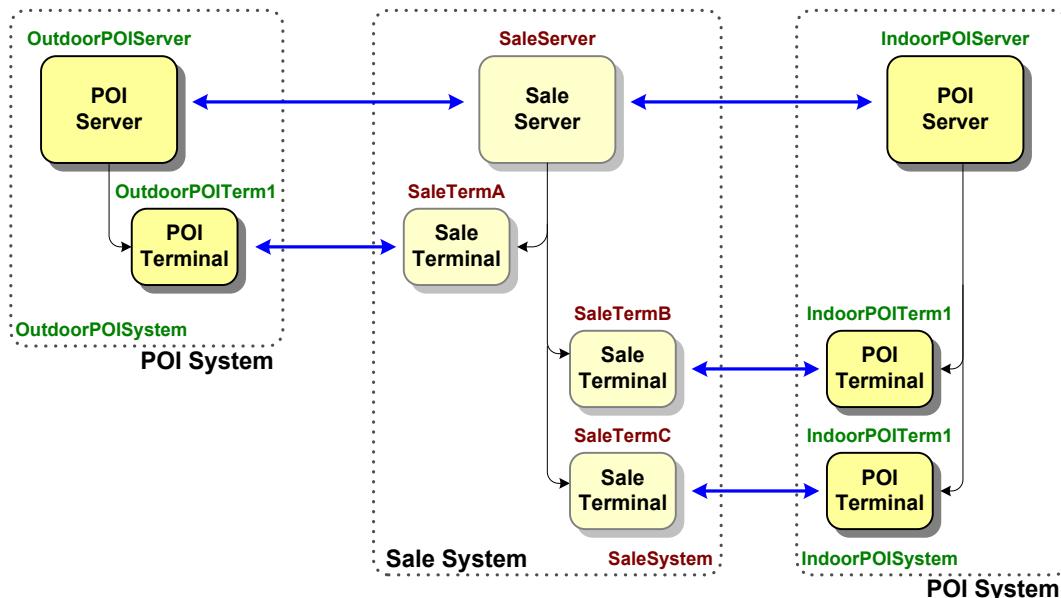


Figure 20: Example of Multiple POI Systems

2.5.3 Logical Connections Between System Components

The Sale to POI protocol exchanges messages between the following pairs of components which are in relation:

- The Sale Server and the POI Server, when the processing of the request does not involve the cooperation of any Terminal,
- The Sale Terminal and the POI Terminal which are used together to make a transaction, with the help or not of the Servers.

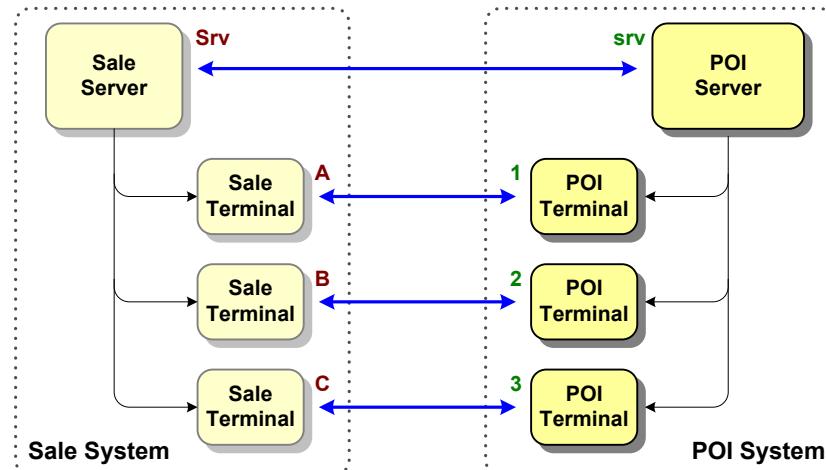
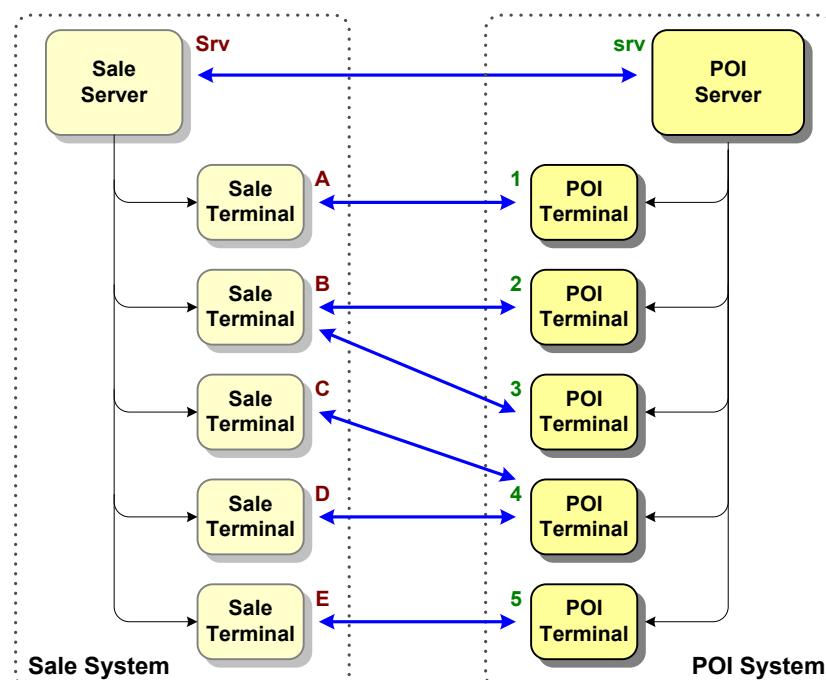


Figure 21: Relationships between System Components

The Figure 21: Relationships between System Components above presents an example of these relationships between components of the POI and Sale Systems.

However, a one-to-one relationship between Sale Terminal and POI Terminal is not required. A Sale Terminal might handle several POI Terminals, and a POI Terminal may be shared between several Sale Terminals as in the example below. In this example, the Sale Terminal B uses the POI Terminals 2 and 3, and the POI Terminal 4 is shared by the Sale Terminals C and D.



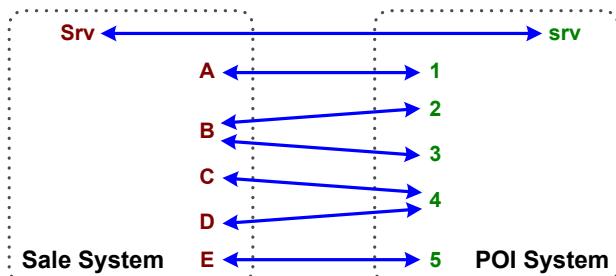


Figure 22: Specific Relationships

These relationships between system components are the channels that support exchanges of messages between the Sale System and POI Systems. They are *Logical Connections* between system components, supported by *Physical Connection* (i.e. transport connections) between the two systems.

Depending on the architecture of the Sale and the POI systems, these related transport connections are established directly between these two components, or through another component.

The two following sections present the various cases of these direct or indirect Logical Connections between or through Terminals and Servers of the Sale and POI Systems.

2.5.4 Logical Connections Between Terminals

A Sale Terminal and a POI Terminal are in relation in the Sale to POI protocol when:

- A transaction is realised in real time on the Sale Terminal, which asks to a particular POI Terminal to process part of the transaction.
- The Sale Terminal needs to use a device or a low-level service of particular POI Terminal.
- The POI Terminal needs to use a device or a low-level service of particular Sale Terminal.

This section details the various transport connections which support the Logical Connections between a Sale and a POI Terminal:

1. The case where all the POI Terminals are independents (i.e. there is no global POI modules, so no POI Server).
2. The case of the Terminal to Terminal links, where each POI Terminal is connected to a Sale Terminal.
3. The case of the Terminal to Server links, where all the connections to the POI Terminals are carried through the POI Server.
4. The case of the Server to Server links, where all the connections from the Sale Terminals to the POI Terminals are carried from the Sale Server to the POI Server.
5. The case of the Server to Terminal links, where all the connections from the Sale Terminals are carried through the Sale Server.

2.5.4.1 Connected POI Architecture

The Connected POI is the architecture where all the POI Terminals are independents, without POI Server or global processing. In this case, there is a dedicated link for each pair of Sale and POI Terminals.

On the figure below, links supported by one or several transport connections, are represented by a continuous blue line, and the arrow gives the direction of the association initiation.

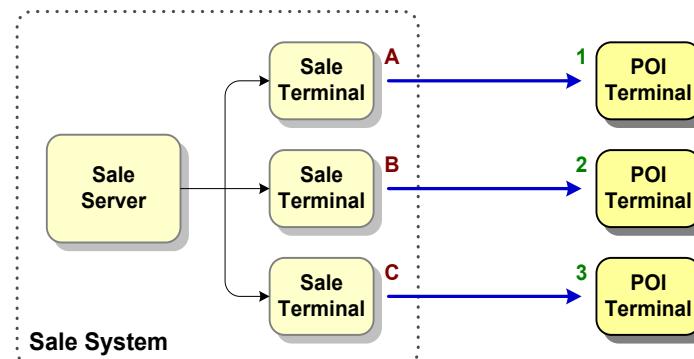


Figure 23: Connected POI Terminal Architecture

2.5.4.2 Terminal to Terminal Transport Connections

This is the typical case of the Clustered POI configuration. Each POI Terminal is connected to a Sale Terminal. In this case, there is also a dedicated link supported by one or several transport connections for each pair of Sale and POI Terminals.

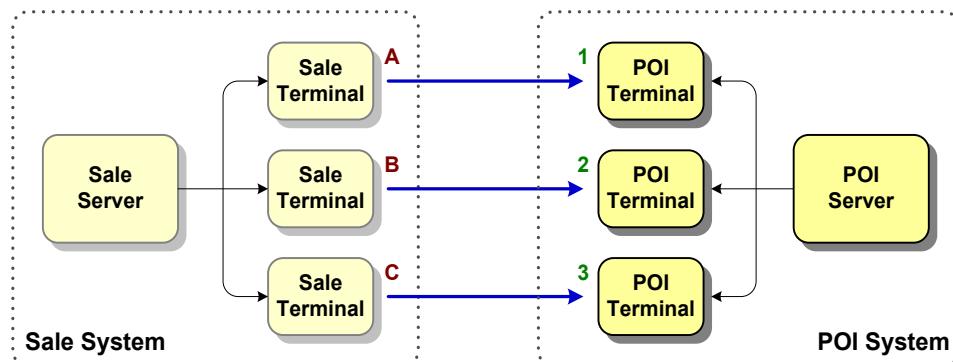


Figure 24: Terminal to Terminal Links

2.5.4.3 Terminal to Server Transport Connections

This is a typical case of the Distributed POI configuration. The POI Server manages globally the interface with the Sale System.

To be in relation with a POI Terminal, a Sale Terminal opens a connection to the POI Server, and will send inside the application messages the logical identification of the targeted POI Terminal.

As the Sale to POI protocol define interface between systems only, the way the POI Server reaches the POI Terminal is POI System implementation dependant (they are denoted by dotted lines between the POI Server and the POI Terminal).

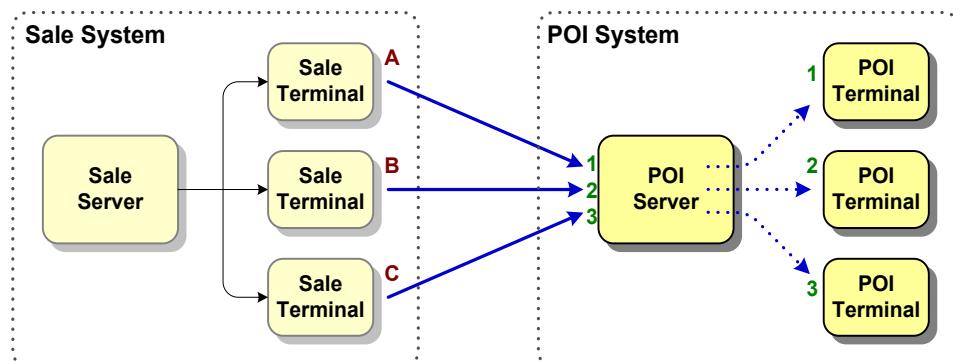


Figure 25: Terminal to Server Links

2.5.4.4 Server to Server Transport Connections

In this case, the Sale Server and the POI Server manage on each side, the Sale to POI interface.

Every transport connection between the Sale Server and the POI Server could be dedicated to a logical link between a Sale Terminal and a POI Terminal. Like the previous case, links and connections management between the Server and the Terminals of the same System are implementation dependant, and have no influence to the Sale to POI protocol (they are also symbolised by dotted lines between the POI Server and the POI Terminal).

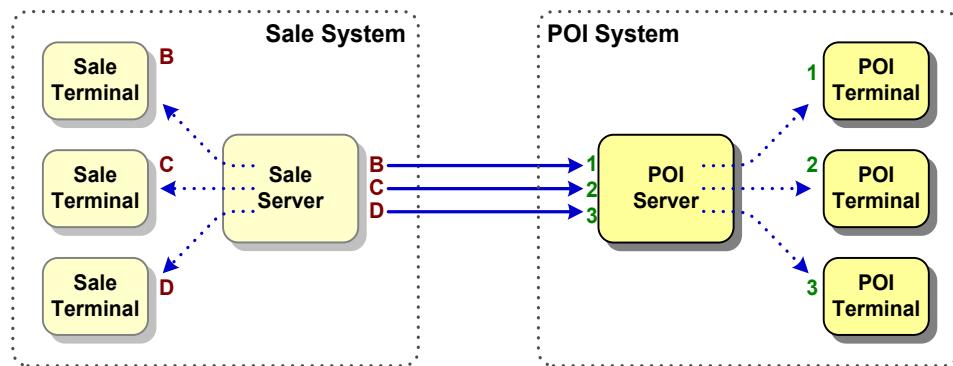


Figure 26: Server to Server Links

2.5.4.5 Server to Terminal Transport Connections

This is the last case, where the Sale Server manages the interface to the POI System directly with the POI Terminals. Once again, the links and connections management between the Sale Server and the Sale Terminals are implementation dependant.

There is a transport connection between the Sale Server and each POI Terminals. Once again, the links and connections management between the Sale Server and the Sale Terminals are implementation dependant.

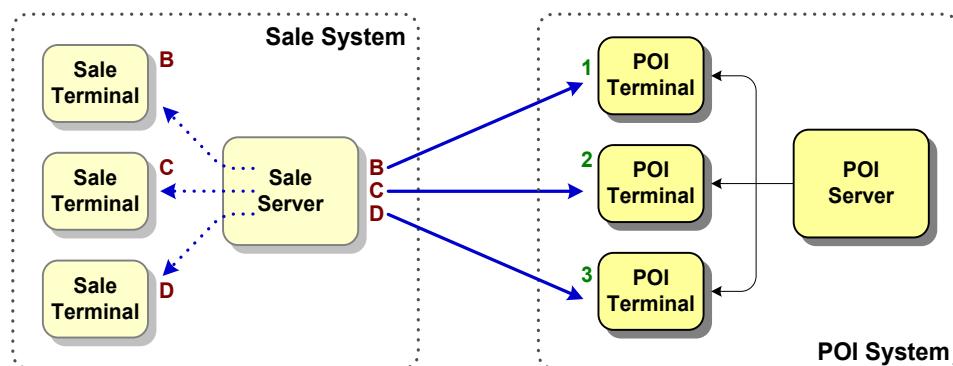


Figure 27: Server to Terminal Links

2.5.5 Logical Connections Between Servers

The Sale Server and the POI Server are in relation in the Sale to POI protocol when a transaction is global to the Systems and does not involve particular Terminal or device.

This section details the various types of associations between the Sale Server and the POI Server:

1. The case of the Connected POI architecture, where all the POI Terminals are independent.
2. The cases of direct Server to Server link, whatever the connections from the Sale Terminal to the POI Terminal are carried through.
3. The cases of indirect Server to Server link, where the Server to Server connection goes through one or several Terminals.

2.5.5.1 Connected POI Architecture

In the Connected POI architecture, all the POI Terminals are independent without POI Server. So the dedicated link between Sale and POI Terminals conveys the information from both the Sale Terminal and Sale Server. A POI Terminal is like a whole POI System and is then considered by the Sale System as a whole system including a Terminal and a Server.

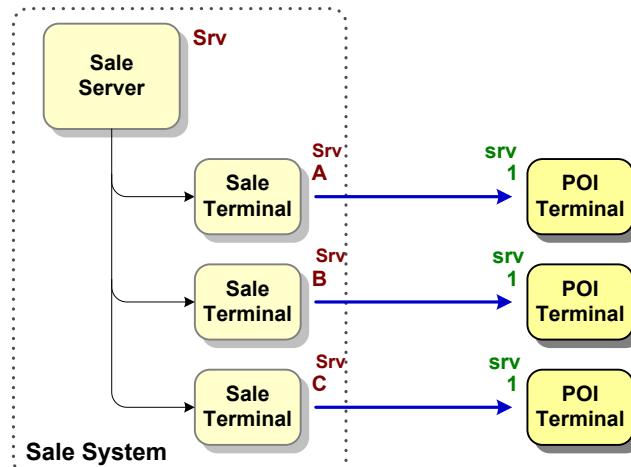


Figure 28: Connected POI Architecture

2.5.5.2 Direct Server to Server Logical Connections

In this case, there is a direct link supported by one or several transport connections between the Sale Server and the POI Server that carries commands for global services.

This type of Server to Server dialogue can be supported by various system architectures of the Sale and the POI Systems, as those we just describe for the links between Terminals.

The first situation is the Clustered POI System architecture where the POI Server is not reachable by the Sale Terminal, but it might be available by the Sale Server.

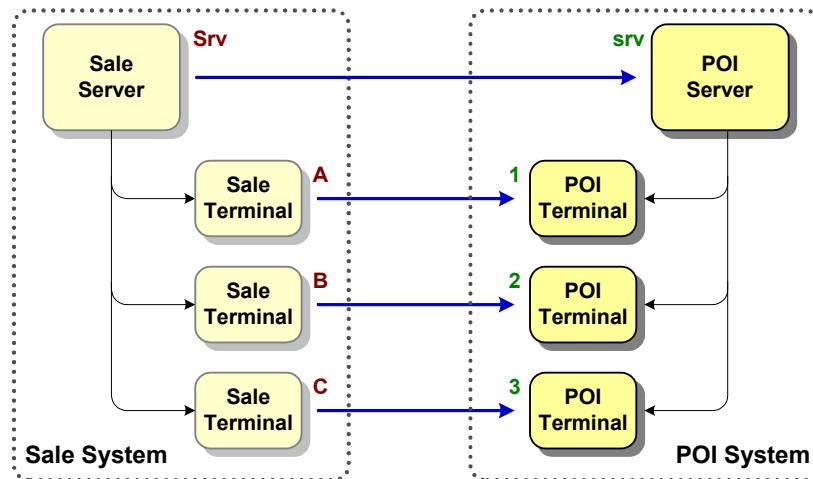


Figure 29: Server to Server Link for Clustered POI

Another case is when the POI Server manages completely the Sale to POI interface. The POI Server receives in addition the Sale Server connections, as presented in the figure below.

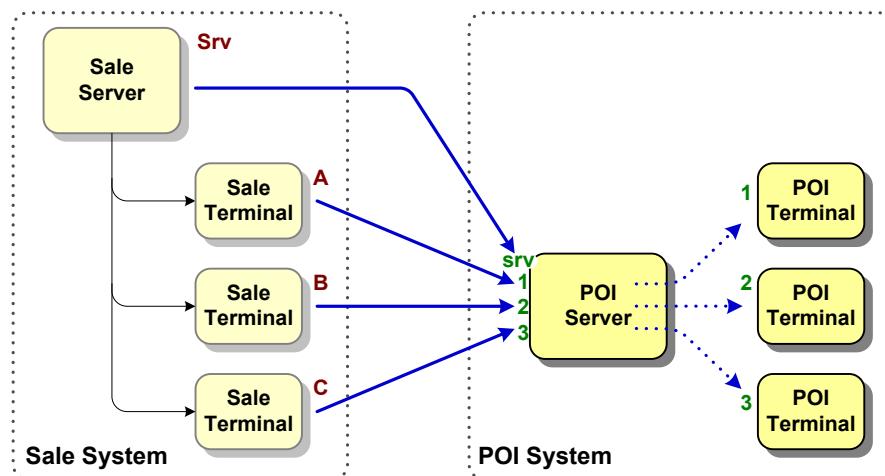


Figure 30: Server to Server Link, Terminal to Server case

An even more manifest case is when Servers at each side manage the Sale to POI interface. All the Terminals and Server transport connections are directly between the two Servers.

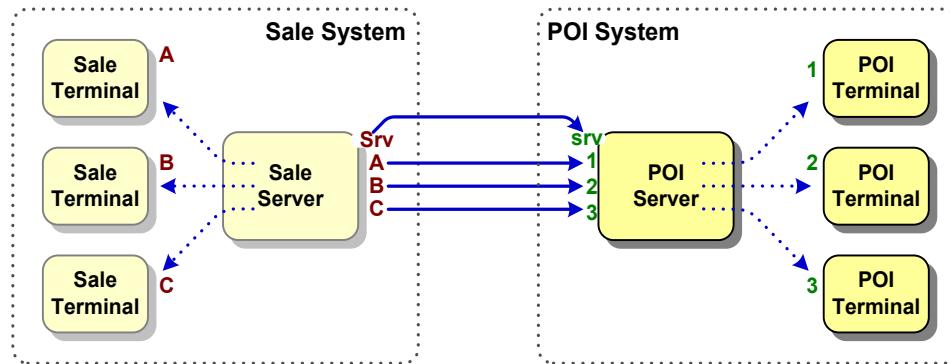


Figure 31: Server to Server Link, Server to Server case

A usual case is when the Sale Server manages completely the Sale to POI interface. The Sale Server opens in addition a connection to the POI Server.

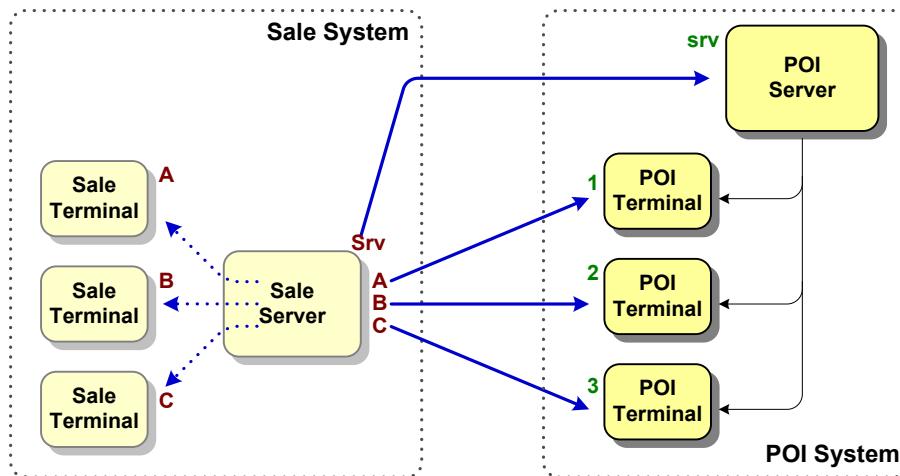


Figure 32: Case 2.4 – Server to Server Link, Server to Terminal case

2.5.5.3 Indirect Server to Server Logical Connections

Other cases occur when there is no direct links between the Sale and the POI Server to carry global services. We cannot describe all the possible combinations of the Sale POI and Server to Server architectures, but only some typical examples.

The first example is in the Clustered POI System where both the Sale Server and the POI Server use a Sale Terminal to POI Terminal connection to support the Server to Server connection.

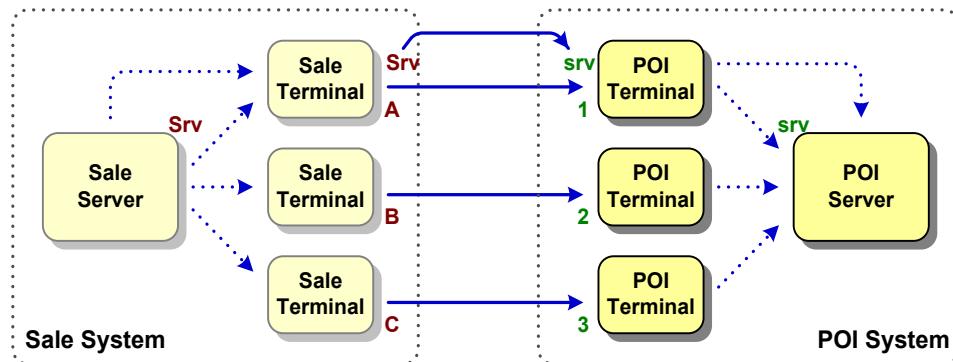


Figure 33: Server to Server Link through Terminal Connections

The second example is in the Distributed POI configuration, where the Sale Server uses a POI Terminal to connect to the POI Server.

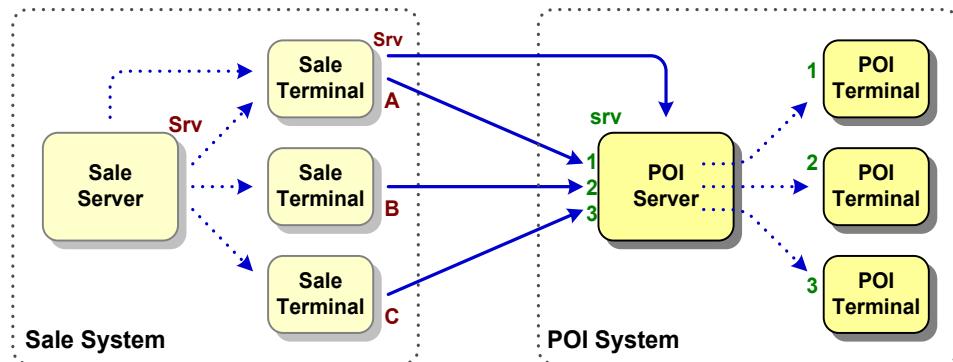


Figure 34: Server to Server Link through Sale Terminal Connection

The last example is where the Sale Server manages the interface to the POI System directly. The Sale Server uses a Sale Terminal to connect to the POI Server.

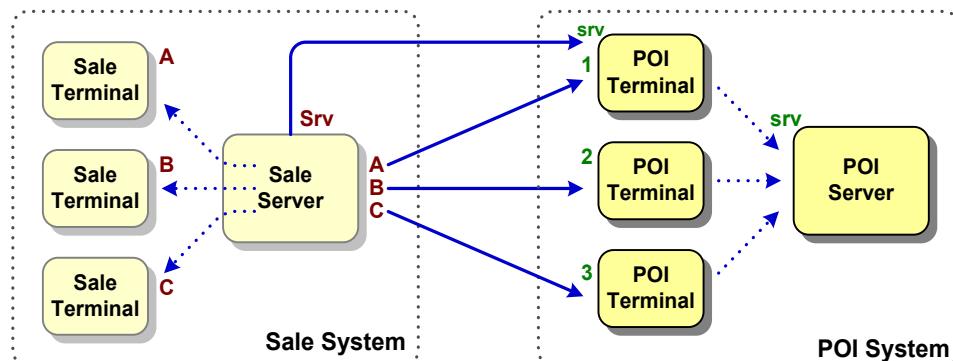


Figure 35: Server to Server Link through POI Terminal Connection

2.6 Configuration and Examples

2.6.1 Architecture Configuration Parameters

All the configuration parameters which are presented below, could be explicit or implicit (i.e. either parameters internal to the implementation, or "hard coded" or "by design" configuration). They are presented to simplify the understanding of the specifications against the various possible architectures.

The Sale and the POI Systems have to know identification of each system to allow exchange of Sale to POI messages.

Name	POI System Name
<i>Definition</i>	POI System Name identification.
<i>Usage</i>	Prefix of all the POI System components identifiers POIID
<i>Specification</i>	2.5.1 Identification of Systems and Components

Name	Sale System Name
<i>Definition</i>	Sale System Name identification.
<i>Usage</i>	Prefix of all the Sale System components identifiers SaleID
<i>Specification</i>	2.5.1 Identification of Systems and Components

Configuration 1: Sale and POI Systems Identification

For each Terminal to Terminal relationship, the type of link and the identification of the remote Terminal:

Name	Terminal Relation
<i>Definition</i>	Type of Terminal to Terminal relationship.
<i>Usage</i>	Transport connection and message dialogue.
<i>Specification</i>	2.5.4 Logical Connections Between Terminals

<i>Label</i>	<i>Description</i>
Connected	Connected POI System
Term2Term	Terminal to Terminal link.
Term2Serv	Terminal to Server link.
Serv2Term	Server to Terminal link.
Serv2Serv	Server to Server link.

Name	Sale (or POI) Terminal Identifier
<i>Definition</i>	Identification of the Terminal.
<i>Usage</i>	Logical addressing of the Terminal.
<i>Specification</i>	2.5.1 Identification of Systems and Components

Configuration 2: Terminals Identification and Logical Connections

The same information for the Server to Server relationship:

Name	Server Relation
Definition	Type of Server to Server relationship.
Usage	Transport connection and message dialogue.
Specification	2.5.5 Logical Connections Between Servers

Label	Description
Connected	Connected POI System
Direct	Direct Server to Server link.
Indirect	Indirect Server to Server link.

Name	Sale (or POI) Server Identifier
Definition	Identification of the Server.
Usage	Logical addressing of the Server.
Specification	2.5.1 Identification of Systems and Components

Configuration 3: Servers Identification and Logical Connections

3 Protocol Management

*"Protocol implementation shall follow a general principle of robustness: be strict in what you do, be tolerant in what you accept from others."*⁸

This chapter specifies the management of the Retailer protocol. It includes the organisation of the protocols levels, the transport protocols that can be used, the specifications of the transport protocol services, and the specifications of the application messages management.

3.1 General Organisation

The Retailer Protocol follows the traditional organisation in five layers of the model defined for the TCP/IP protocol suite.

The two layers we are interested in are:

- The Application Layer where the protocol specified here is located, and
- The Transport Layer that the Application protocol interfaces with the Transport Services.

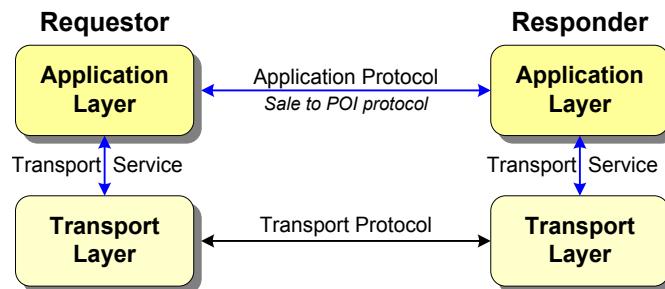


Figure 36: Protocols Organisation

The Sale to POI Application Protocol is independent of the Transport Protocol, and can be used without modification with various Transport Protocols, at the exception of the interface to the Transport Services, especially the transport addresses.

The Transport Protocol and therefore the Application Protocol are also independent of the communication infrastructure used below these layers, at the exception of the quality of service of the communication involving the tuning of some value of configuration parameters (e.g. value of timeout).

The Transport Protocol is a standard protocol allowed by the Application Protocol. The Application Protocol requires the Transport Services which can be used by the Application Protocol, and specifies the way to use them.

⁸ From the RFC 793, *Transmission Control Protocol* (TCP transport protocol specifications)

If a standard Transport Protocol authorized by the Application Protocol does not offer a particular Transport Service required by the Application Protocol, the Application Protocol specifies this Service through a Transport Adaptation Layer. This is for instance the case for the application message delimitation, which is a service not provided by the TCP Transport Protocol, but required for the decoding of a XML message before to deliver it to the Application Layer. Depending on the implementation of the application protocol, it could also be used to provide a particular flow control, for the connection management, etc...

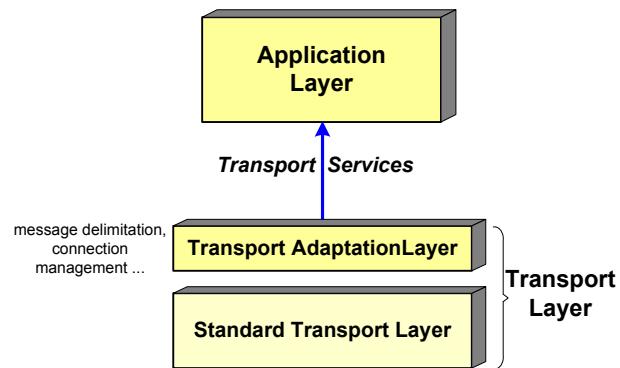


Figure 37: Transport Adaptation Layer

As far as the transport services and their usage are well defined, the definition in the future of a new Transport Protocol compliant to the defined Transport Service, will not impact the specifications of the Application Protocol.

3.2 Transport Services Management

This section specifies the transport services used by the Sale to POI protocol. The section is completed by the state diagrams of the two systems, and their configuration parameters.

3.2.1 Transport Connection Handling

3.2.1.1 General Rules

One or several transport connections are established between components of the Sale System and the POI System, as described in the sections *Logical Connections Between Terminals* and *Logical Connections Between Servers*.

The number of transport connections open between a Sale Terminal and a POI Terminal or between the Sale Server and the POI Server is a configuration parameter.

The Sale system has always the initiative of the exchanges at the application level⁹, i.e. the exchange of messages defined in this specification. However, transport connections can be initiated by either the Sale System or the POI System.

In addition, depending on the transport protocol (TCP, HTTP...), the POI may or may not open its own transport connections during the processing of a Sale System request, for the request of device exchanges.

- Rule 1: The Sale System can open a transport connection for requesting an application message exchange with the POI. If successful, the Sale System is then responsible for initiating the application exchange.
- Rule 2: The POI System can open a transport connection. If successful, the Sale System is then responsible for initiating the application exchange. If connection fails, POI system must retry indefinitely to connect to Sale System with an adaptative delay between attempts.

When transport connection are open by the Sale System, if all transport connections are closed, the POI System cannot send unsolicited notification to the Sale System.

The transport connection is closed by the transport adaptation layer only when this layer cannot continue the processing (e.g. message too big for the transport service layer, response to the transport connection request...), as defined in the section 3.2.4 *Transport Error Handling*.

⁹ At the exception of the Event notification sent by the POI system.

3.2.1.2 Standard Sequence Flow

Sequence flow of a message pair exchange inside a transport connection initiated by the Sale System contains the steps presented below.

1. The Sale System component sends a transport connection request to the POI System component. The Sale System arms the timeout TC1 to wait for the transport connection response from the POI System.
2. The POI System receives the transport connection request, accepts it and sends a transport connection confirmation. At this time the transport connection is established for the POI System, which can send message, according to the dialogues and message specification.
3. After reception of the connection confirmation, the transport connection is established for the Sale System, and disarms the timeout TC1.
4. The Sale System and the POI System could exchanges messages of the Sale to POI protocol on this transport connection.
5. When there is no message pair in progress¹⁰ and it decides to stop dialogue between these two entities, the Sale System sends a transport disconnection request. The connection is then considered by the Sale System as closed, and it cannot receive any data on this transport connection.
6. The POI System receives the transport disconnection request and sends a transport disconnection response to the Sale System¹¹. The transport connection is then considered as released for the POI System.

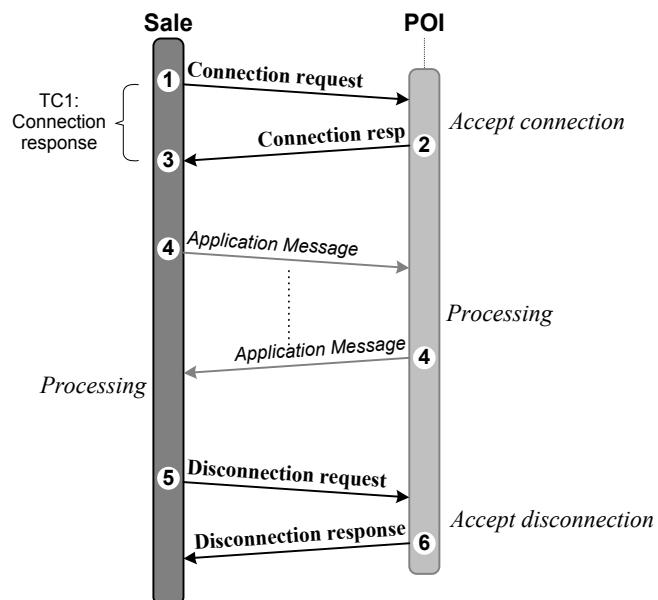


Figure 38: Sale Initiated Transport Connection Standard Sequence Flow

¹⁰ A good practice is that the receptor of the last response closes the transport connection. This is normally the case because, with the exception of the notification from the POI (which are unsolicited), the Sale System always receive the last response message of a dialogue.

¹¹ On the TCP protocol implementations, only the protocol stack receives this request, because the Sale System has already closed the connection.

Sequence flow of a message pair exchange inside a transport connection initiated by the POI System contains the steps presented below.

1. The POI System component sends a transport connection request to the Sale System component. The POI System arms the timeout TC1 to wait for the transport connection response from the Sale System.
2. The Sale System receives the transport connection request, accepts it and sends a transport connection confirmation. At this time the transport connection is established for the Sale System, which can send message, according to the dialogues and message specification.
3. After reception of the connection confirmation, the transport connection is established for the POI System, and disarms the timeout TC1.
4. The Sale System and the POI System could exchanges messages of the Sale to POI protocol on this transport connection.
5. When there is no message pair in progress and it decides to stop dialogue between these two entities, the Sale System sends a transport disconnection request. The connection is then considered by the Sale System as closed, and it cannot receive any data on this transport connection.
6. The POI System receives the transport disconnection request and sends a transport disconnection response to the Sale System. The transport connection is then considered as released for the POI System.

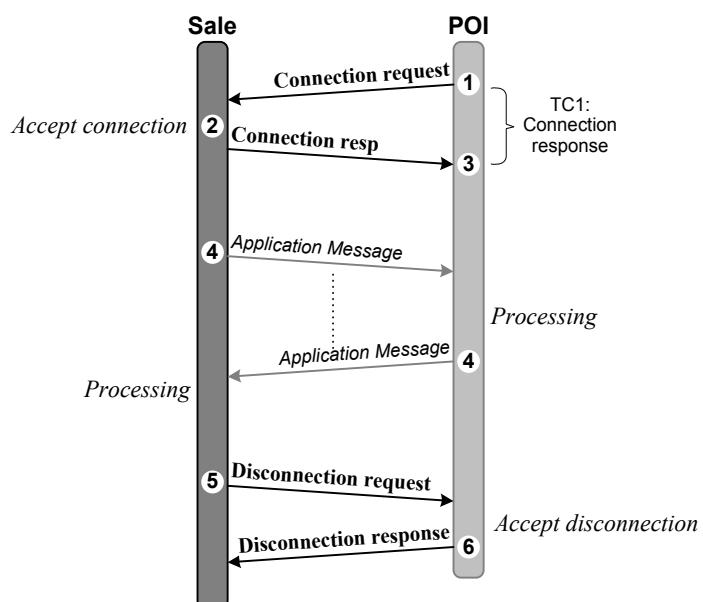


Figure 39: POI Initiated Transport Connection Standard Sequence Flow

3.2.2 Data Transfer

3.2.2.1 General Rules

Connections cannot be assigned to a particular pair of Sale Terminal and POI Terminal. Only messages tell which components of the Systems are the sender and the receiver of the message. So it is not recommended, except for a physical link between Terminals, for the POI to restrict an established transport connection to one Sale Terminal only.

Data are transferred inside a transport connection, at the request of the Application protocol to send a Sale to POI message which is the atomic quantity of data exchanged by the application protocol.

The dialogue of application messages might result in sending a message and receiving another message at the same time. Typically, during the reception of a message, which could be splitted in several segments due to the communication infrastructure, the application protocol could request to send a message.

In some cases the application requires sending a new message before the previous one was completely sent by the transport layer. To enable the reception and recognition of a message, the sender has to serialise the sending of consecutive messages.

3.2.2.2 Standard Processing Flow

For sending a message:

- The Application Protocol layer requests to send an application message to the interface of the transport protocol,
- The transport interface adds four bytes containing the length of the application message, and requests to the transport to send these four bytes followed by the application message.

For the reception of a message, the interface to the transport protocol:

- Waits for the reception of four bytes to get the length L of the message to receive, and arms the timeout TC2 to monitor the reception of the complete application message.
- Waits for the reception L bytes to provide a message to the Application Protocol layer.
- At the reception of the last data unit of this message, stop the TC2 timer, and deliver to the application layer, the L bytes, content of the application message.

These processing flows are included in the general state diagrams of a transport connection management for the Sale and the POI Systems.

3.2.3 Addressing

A transport address is the identification of a peer, Requestor or Responder, which permits to establish a transport connection with this peer.

The form of the transport address depends on the type of transport protocol using this address (e.g. an IP address or a DNS name of the node, and a TCP port for the TCP transport protocol).

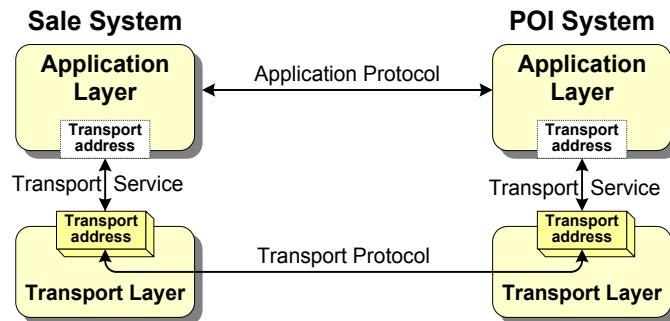


Figure 40: Transport Address

The transport address of the Responder is necessary for the Requestor to establish a transport connection with this Responder.

A transport connection is identified at least by the couple of transport address at each end point of the connection. In case of network connected mode above the transport protocol, other information like the sequence of nodes composing the connection path.

The transport addresses of every POI components to connect are configuration parameters used at the application level by the Sale System as a destination address to establish transport connection carrying out exchange of messages.

Each application protocol has to define assignment of the POI components transport addresses, globally for the whole application protocol, per class of message, or per service.

3.2.4 Transport Error Handling

This section defines errors detected by the transport protocol and transmitted to the application layer. The responsibility and the choice of error resolution are left to the application layer, unless the transport service layer needs to perform some action to continue to work.

3.2.4.1 Sale System Unable to Establish or receive a Transport Connection (ERTR01)

Definition

The transport layer is unable to establish the transport connection that the application layer of the Sale System has requested to open, because:

- The lower protocol layer has notified a "out of service" communication state as permanent or in response to the attempt to open the connection or the attempt to accept incoming connection.
- The TC1 timer has expired, and the connection in progress is considered as broken (and the connection establishment as a failure).
- The POI target component has denied the connection request, for instance because the POI component has reached the maximum number of connections it can manage in parallel.
- The transport layer has reached the maximum number of connection it can manage in parallel.

Transport Service Behaviour

The transport service layer considers the establishment of the transport connection as a failure, and notifies the result to the application layer of the Sale System.

The maximum number of connections open in parallel is fixed by the configuration parameter "Max Number of Connections" of both the Sale and the POI Systems.

3.2.4.2 Transport Connection Broken (ERTR02)

Definition

The transport layer realises that an open connection is broken. Several reasons might cause this error:

- The transport layer receives a disconnection request.
- An error occurred from lower level when the transport layer waits for the end of a message after the reception of the length prefix and incomplete data units.
- The lower level notifies the transport layer that a defect occurs (e.g. on the physical interface), and the transport connection is broken.
- The remote peer has not respected the state diagram of the connection management.
- The remote peer has released the transport connection.

Transport Service Behaviour

The transport service layer notifies the release of the transport connection to the application layer. Partial message are deleted, and not sent to the application layer.

3.2.4.3 Unable to Send a Message (ERTR03)

Definition

The transport layer cannot send an application message. Several reasons might cause this error:

- An error occurred from lower level when the transport layer sends an application message at the request of the application layer.
- The connection was broken just beforehand, but the application layer did not receive the disconnection notification before the request to the transport layer to send a message.

Transport Service Behaviour

The transport service layer notifies the error to the application layer.

It is worth noticing that the transport is not always able to report error to the application layer. In particular when a connection is broken, the transport service has removed the connection context, and might have some difficulties to identify the context of the error.

3.2.4.4 Message Too Big (ERTR04)

Definition

The transport layer cannot receive an application message, because the prefix contains a message length above the limit of the transport layer memory.

Transport Service Behaviour

The transport service layer has an internal parameter containing the size limit of a message, or a global limit for the set of message buffers. When the limit is reached, transport layer:

- Release the connection on which the message has to be received.
- Notifies the error to the application layer.

As the transport layer cannot receive the message, it has to release the transport connection. Most of the time, errors of this class happen when there is a desynchronisation of message delimitation between the two peers.

The maximum size of an application message to be received is fixed by the configuration parameter "Max Message Size" of both the Sale and the POI Systems.

3.2.4.5 Late Arrival (ERTR05)

Definition

The transport service layer receives a data unit for the application after a timeout expiration and the request for a disconnection to the underneath transport connection.

The same phenomenon might arrive with a greater likelihood at the application layer.

The drawing below shows such error example, when the application timeout on the message response expires.

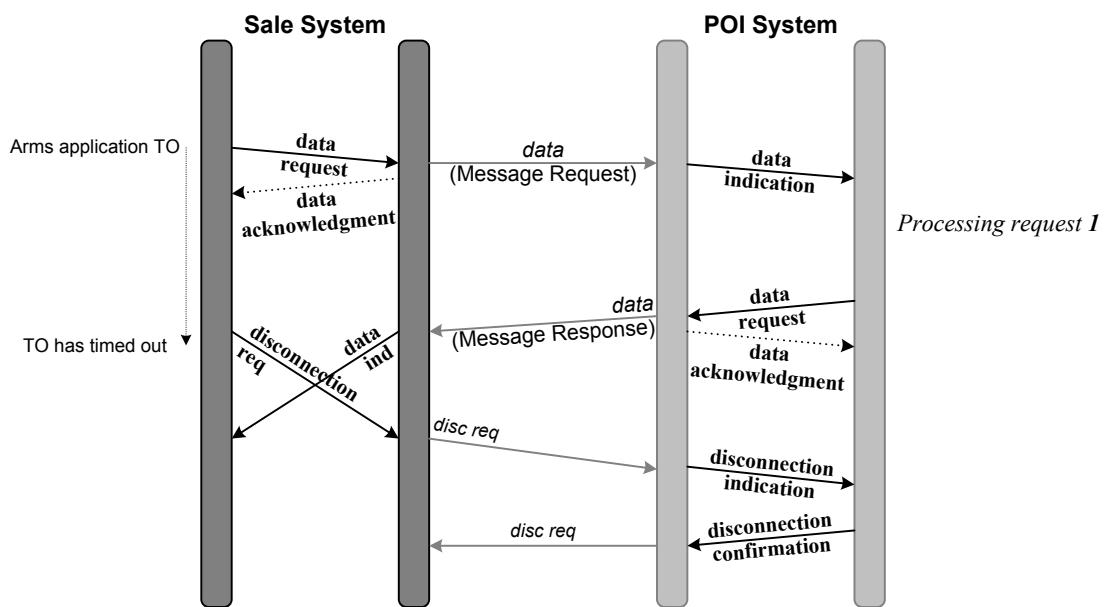


Figure 41: Late Arrival Error Example

Transport Service Behaviour

The transport service layer ignores the message or data units.

As already noticed, the transport has removed the connection context, and might have some difficulties to identify the context of the error.

Implementation has to be careful not to assign the data unit to another connection.

3.2.4.6 Max Global Number of Connections (ERTR06)

Definition

On the arrival of an incoming connection request, the transport service layer cannot open a new transport connection, because it has reached the maximum number of transport connections it can handle in parallel.

The same phenomenon might arrive with a greater likelihood at the application layer.

Transport Service Behaviour

The behaviour is implementation dependant. The transport service layer or the application layer has the configuration parameter "Max Number of Connections" containing the limit number of open connections. When the limit is reached, transport layer:

- Decline the incoming connection request.
- Notifies the error to the application layer on the connection requestor side.

3.2.4.7 Incomplete Application Message (ERTR07)

Definition

After the reception of an application message header, the transport layer arms the timeout TC2 to monitor the reception of the complete application message.

The expiration of the TC2 timeout terminates the reception of the application on this error case which might happen:

- When a serious error occurs on a lower communication level or on the communication infrastructure between the 2 peers.
- The remote transport or application layer encounters a serious error, or is out of order.

Transport Service Behaviour

The transport service layer:

1. Discards the uncompleted application message,
2. Close the transport connection to avoid problem of data synchronisation (the next data unit can be the start of the next application message, or part of the previous application message),
3. Notifies the error to the application layer.

3.2.4.8 Other Errors

All other errors are detected or resolved at the application layer, as:

- Non understandable message: all decoding of message are made by the application layer, so this error is detected and resolved at the application layer.
- Timeout: most of the time, the application layer limits the time the other peer can has to process a message and answer to it. In this case, it is recommended to close the transport connection.
- Late Arrival Message: in case of a big number of events to perform, some crossing event can be observed between the application layer and the transport layer. A disconnection request from the application layer can cross an application message from the transport layer.
- Busy situation: when the application layer has not enough resources to process al the message received, it can answer that it is busy at the application level to the other peer.

Depending on the relationship between the Sale and POI component (Terminal or Server), the application limits the number of transport connections between these two components (configuration parameter "Number of Connections").

Example

The relationship between a Sale Terminal and a POI Terminal is a Terminal to Terminal link (i.e. the parameter "Terminal Relation" has the value "Term2Term"), and the parameter "Number of Connections" for these two terminals is 1. The Sale Terminal may only open one transport connection towards this POI Terminal. Be careful to the crossing of connection and disconnection as described in the figure below.

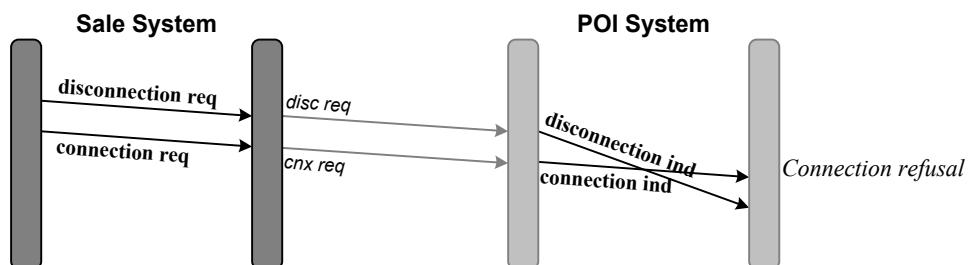


Figure 42: Crossing of Disconnection and Connection

3.2.5 State Diagrams

3.2.5.1 Sale System

This is the state diagram of the Sale System to manage a transport connection. The states and events of this diagram are detailed in the tables below.

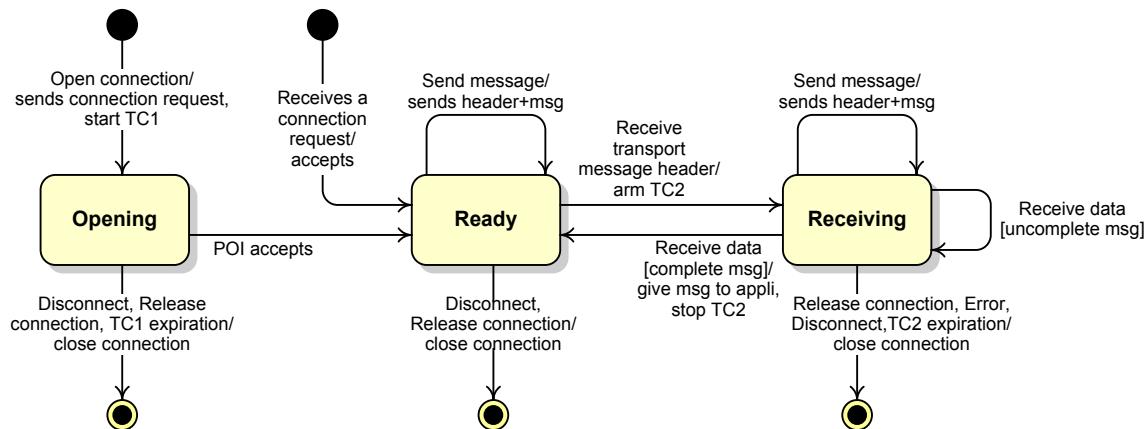


Figure 43: Sale System Transport Connection Management State Diagram

State	Definition
Opening	This state exists only when the Sale System opens the connections. The transport connection is opening. The Sale System has sent a connection request to the POI component, and is waiting for the response.
Ready	The transport connection is open. The connection between the Sale System and the POI component is established and Sale System is waiting to send or receive messages.
Receiving	The transport connection is open. The Sale System has received the transport protocol header containing the length of a message from the POI component, possibly part of the message, and is waiting for the end of the message.

Table 1: Transport Connection Sale System States

Event	Definition
Open connection	The application protocol request to open the transport connection.
Connection request	The remote peer (POI component) sent a connection request.
Release connection	The application protocol request to release the transport connection.
Disconnect	The remote peer (POI component) or the communication infrastructure releases the transport connection.
POI accepts	The POI accepts to open the transport connection.
Send message	The application protocol request to send an application message.
Receive transport message header	The complete message length has been received in the transport message header (the four bytes of the length header for TCP, or HTTP header containing the Content-Length header field).
Receive data	Data are received on the transport connection.
TC1/TC2 expiration	The timer TC1 (resp. TC2) has expired.
Error	An error is detected on the transport interface: Transport Connection Broken, Unable to Send a Message, or Message Too Big.

Table 2: Transport Connection Sale System Events

3.2.5.2 POI System

This is the state diagram of the POI System to manage a transport connection. The states and events of this diagram are detailed in the tables below.

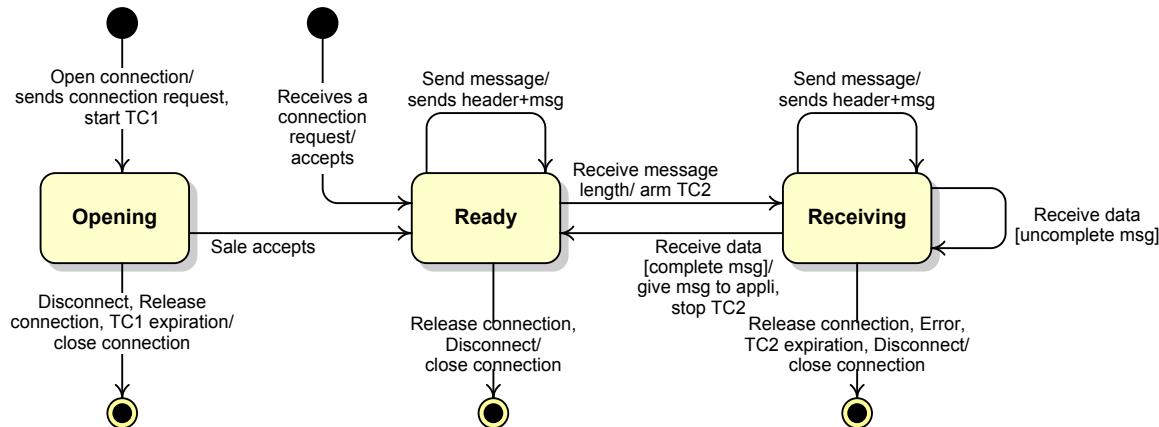


Figure 44: POI System Transport Connection Management State Diagram

State	Definition
Opening	This state exists only when the POI System opens the connections. The transport connection is opening. The POI System has sent a connection request to the Sale component, and is waiting for the response.
Ready	The transport connection is open. The connection between the Sale System and the POI component is established and Sale System is waiting to send or receive messages.
Receiving	The transport connection is open. The POI System has received the transport protocol header containing the length of a message from the Sale component, possibly some part of the message, and is waiting for the end of the message.

Table 3: Transport Connection POI System States

Event	Definition
Open connection	The application protocol request to open the transport connection.
Connection request	The remote peer (Sale component) sent a connection request.
Release connection	The application protocol request to release the transport connection.
Disconnect	The remote peer (POI component) or the communication infrastructure releases the transport connection.
Send message	The application protocol request to send an application message.
Receive transport message header	The complete message length has been received in the transport message header (the four bytes of the length header for TCP, or HTTP header containing the Content-Length header field).
Receive data	Data are received on the transport connection.
TC1/TC2 expiration	The timer TC1 (resp. TC2) has expired.
Error	An error is detected on the transport interface: Transport Connection Broken, Unable to Send a Message, or Message Too Big.

Table 4: Transport Connection POI System Events

3.2.6 Transport Service Configuration Parameters

This section summarises the set of configuration parameters required by the transport service management. These parameters can be for instance downloaded by the EPAS TMS protocol or another way, and their values cannot be fixed by the protocol specifications as their value depends on the communication infrastructure, or the architecture of the Systems.

The Sale and the POI Systems¹² have the following configuration parameters for the transport service management:

Name	TC2
Definition	Application message reception timer
Usage	This timer is armed at the reception of the application message prefix to supervise the reception of the complete application message. It allows the detection of an incomplete application message reception, avoiding a deadlock of the transport connection.
Specification	3.2.2 Data Transfer 3.2.4 Transport Error Handling (Message Too Big, Incomplete Application Message)

Name	Max Number of Connections
Definition	Maximum number of transport connections the Sale or The POI System component can manage simultaneously.
Usage	To avoid error on opening too many transport connections, from the application protocol layer or the remote application protocol.
Specification	3.2.4 Transport Error Handling (Unable to Establish a Transport Connection, Max Global Number of Connections)

Name	Max Message Size
Definition	Maximum size of a message the Sale or The POI System component can receive.
Usage	To detect error of messages length, in particular synchronisation of messages exchange, with the prefix length and the content of the message.
Specification	3.2.4 Transport Error Handling (Message Too Big)

Configuration 4: Transport Service

The Sale System has in addition the following configuration parameters for the transport service management:

Name	TC1
Definition	Transport connection establishment timer
Usage	After the sending of a transport connection request, the Requestor arms the TC1 timer to watch on the response from the Responder. TC1 is reset on the transport connection confirmation reception.
Specification	3.2.1 Transport Connection Handling 3.2.4 Transport Error Handling (Unable to Establish a Transport Connection)

Configuration 5: Sale Transport Service

¹² Of course, the values for the Sale System and the POI System might be different.

For each Terminal to Terminal relationship, and the Server to Server if any:

Name	Number of Connections
Definition	Number of transport connections the Sale System component may establish simultaneously toward this POI System component.
Usage	To avoid error on opening too many transport connections between two components. The number depends on the type of relation between these two components defined by the parameter "Terminal Relation" and "Server Relation" in 2.6 Configuration and Examples Architecture Configuration Parameters.
Specification	3.2.4 Transport Error Handling (Other Errors)

Name	POI Component Address
Definition	Transport address of the POI Component
Usage	This (or these) transport address is used by the Sale System to establish a transport connection with a POI System component.
Specification	3.2.3 Addressing

Configuration 6: Transport Connection

3.3 Transport Protocols

3.3.1 Introduction

Given the great variety of contexts, particularly for the communication between systems and between terminals, the number of transport protocols to take into account is limited. For each transport protocol which is used by Application Protocols, the following items have to be described:

- Specification of transport services required by the application protocol, which are not provided by the transport protocol.
- Specification of the ways transport services are used by the application protocol, that are particular to this transport protocol.
- Configuration parameters to be used to manage the transport protocol: addressing, time-out, data context, limits...

For the current specification two transport protocols may be used:

- TCP (Transmission Control Protocol),
- HTTP (Hypertext Transfert Protocol)

They are specified in the two following sections.

3.3.2 TCP Protocol

The TCP transport protocol is specified in the RFC 793 (Transmission Control Protocol - DARPA Internet Program) in September 1981.

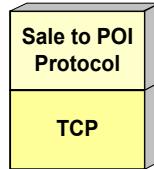


Figure 45: TCP Protocol

3.3.2.1 Typical Use

Because of its widespread availability, the TCP transport protocol remains one of the favourite transport protocols. It can be used as:

4. An end-to-end transport protocol between two entities at each extremity of the Application Protocol.

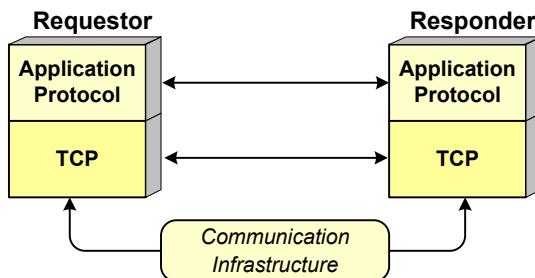


Figure 46: Peer-to-peer TCP Transport Protocol

5. An interface to a gateway or a driver to make the conversion of transport protocol with the other side of the Application Protocol. This case is only a transitory solution allowing the adaptation of a legacy system to the Application Protocol, avoiding the specification of all the existing communication protocols.

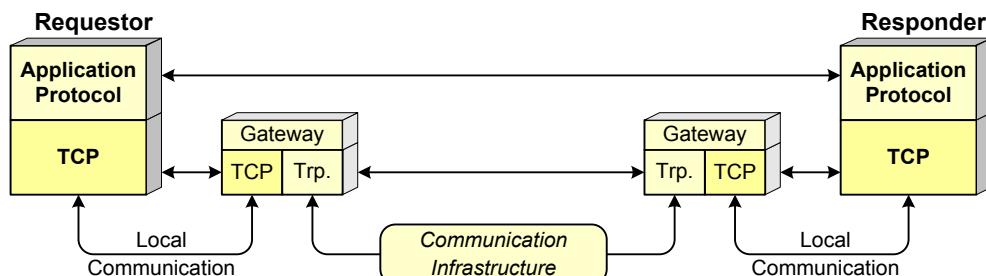


Figure 47: Gateway TCP Transport Protocol

6. An interface between two applications using Application Protocol when they are on the same platform.

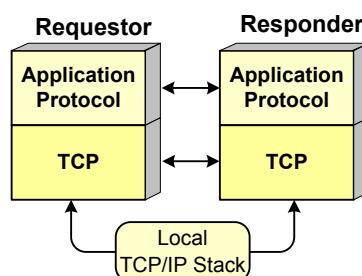


Figure 48: Local TCP Transport Protocol

3.3.2.2 Message Delimitation

The TCP protocol is a stream protocol which does not offer message delimitation or data-unit delimiting. It uses the general mechanism of message delimitation provided for all the transport protocols.

Definition

Message delimitation or data-unit delimiting is the service provided by the transport protocol allowing the recognition of an application data-unit by the transport protocol, in order to transfer a complete application message to the Application Protocol.

Specifications

Application messages are preceded by four bytes containing the length in network order¹³, of the application message. The length does not include the four bytes containing the length.

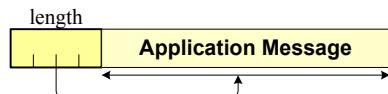


Figure 49: Message Header Length

Rule 1: If the four bytes of the message length are all equal to zero, it is considered as a zero length application message. These four bytes are ignored, and the next four bytes if any, are interpreted as the header length of the following application message.

Notes

An example of application message with the length, to be sent on the transport connection, is provided in Annex B.

Use of message delimitation at the transport level ensures:

- Independence of this service on the data coding used by the application, which can provide this functionality (e.g. ASN.1/BER data coding).
- Restrict message delimitation to the transport layer interface avoids partial progressive decoding of the message mixed with reception of message fragments,
- Separates decoding of the message from its reception processing.

Use of the message length to delimit a message has the advantage to be able to send any value in the application message, but prohibits resynchronisation of message after a data loss.

Use of a fixed number of bytes to contain the length of the message facilitates implementation of message reception.

Use of specific characters to delimit the message requires either a parsing of the message to treat these specific characters, either to forbid use of these characters by the application in the messages.

A number of four bytes has been chosen to carry out the message length, because of the 64 K bytes limitation for a 2-bytes length, and the effort required passing from 2-bytes to 4-bytes length.

¹³ Most significant byte first (i.e.big endian).

3.3.2.3 Transport Connection Handling

To allow a maximum of flexibility, the Sale System has the entire responsibility to manage the connection at its convenience, and during the time it considers necessary to its own management.

The Sale System might decide to use permanent transport connections, opens a connection for each message, or mixes any type of management according the requirements of the sales.

Connection Opening

General rules for opening connections applies as defined in section 3.2.1 *Transport Connection Handling*.

Connection Release

The Sale System may release a transport connection whenever the application judges it appropriate. The Sale System has to ensure that all the response messages have been received before to close the transport connection.

Rule 1: The Sale System may close a transport connection at any time.

The POI System may close the transport connection only in case of error.

When the transport connection is closed, the POI System cannot send unsolicited notification to the Sale System.

Keep Alive message

In a connectionless network mode (as IP), it could be difficult to detect a broken transport connection without sending data. In particular, some protocol stacks does not provide such out of band "keep alive" service.

The *Message Delimitation* transport services defined in the section 3.3.2.2 provides the "Keep Alive" service:

- After a certain time without exchange on an open connection, send four null bytes (hexadecimal 00) on the transport connection.

As the reception of those four null bytes is considered as an application message of length zero, they are ignored by the receiver.

3.3.2.4 Addressing

A transport address for the TCP protocol is composed of:

1. The *IP Address* or the *DNS Name* to resolve of the host on which the application protocol lives.
2. The *TCP Port*, to dispatch the connections to the application.

3.3.3 HTTP Protocol

3.3.3.1 Introduction

The current version of the HTTP protocol is v1.1 specified in the RFC 2616 in June 1999 (Hypertext Transfer Protocol -- HTTP/1.1).

HTTP is an application protocol above the TCP transport protocol.

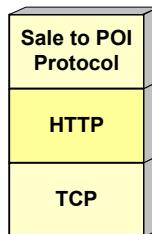


Figure 50: HTTP Protocol

The main advantages to use HTTP instead of TCP are:

- To traverse Firewalls without any addition for incoming connections in the communication infrastructure, as HTTP port is configured everywhere.
- Facilitate the remote location of payment services.
- Simplify the use of intermediary agent at the transport level, as very often used today in the card payment communications.
- Make possible the adoption of Web services. Web services have to be clearly studied before any modification of the specification to adopt them.

HTTP is a request/response protocol between HTTP Client and HTTP Server to exchange multimedia documents that might be used, instead of TCP, to support the Sale to POI protocol.

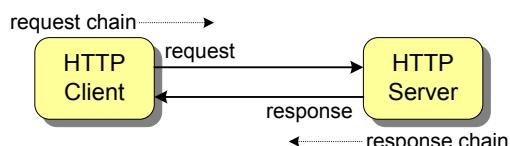


Figure 51: Basic HTTP Communication

HTTP has been designed to include HTTP Intermediaries between the Client and the Server, with dedicated functions.

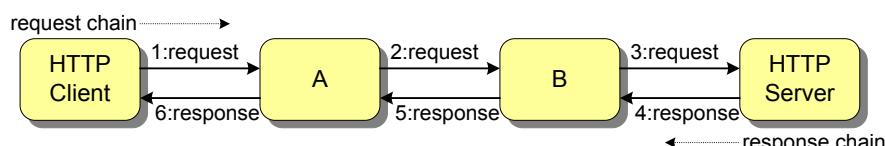


Figure 52: HTTP Request/response Chain

The structure of HTTP request and response messages contains three parts:

- The *Start Line*, indicating for a request the nature of the demand, and for a response the outcome of the request.
- The *Header Fields*, containing information on the management of the message or the protocol. Each *Field*, of the form *HeaderName:HeaderValue*, is located on a separate line. An empty line closes the *Header Fields* part of the HTTP message.
- The *Body*, containing the Sale to POI message, which is present if a message has to be sent.

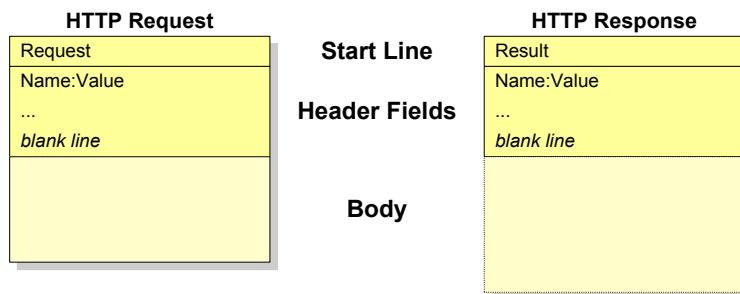


Figure 53: HTTP Message General Format

The header fields are separated in four classes:

- The *General HTTP Headers*, not specific to any particular kind of message (HTTP Request or HTTP Response) or message content (Body),
- The *HTTP Request Headers*, used only in HTTP Request messages,
- The *HTTP Response Headers*, used only in HTTP Response messages,
- The *HTTP Entity Headers*, concerning the resource carried in HTTP Body.

It is recommended to place the general header fields first, then the request or response header fields, and the entity headers at the end of the HTTP header.

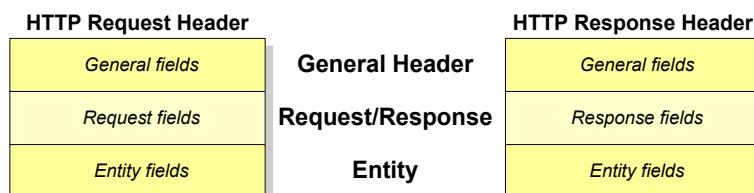


Figure 54: HTTP Header Fields

3.3.3.2 Transport Connection Handling

The standard sequence flow of an HTTP exchange is:

1. The HTTP Client open a TCP connection with the HTTP Server,
2. The HTTP Client sends an *HTTP Request* message,
3. The HTTP Server process the request of the client, and sends an *HTTP Response* message containing the result of the processing,
4. If the connection is persistent, the Client might send other *HTTP Request* messages, potentially in parallel,
5. The HTTP Client closes the TCP connection.

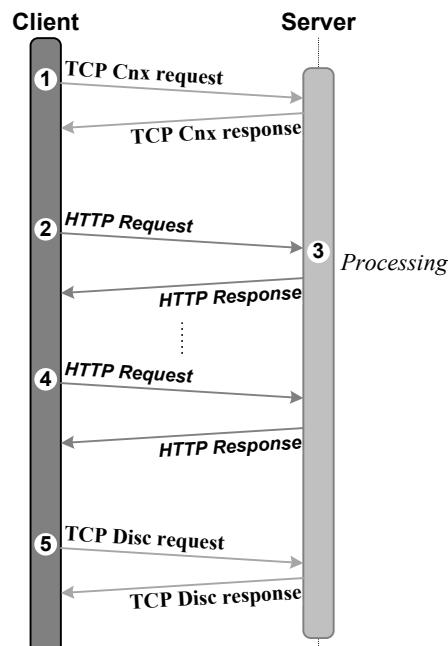


Figure 55: Standard HTTP Sequence Flow

- Rule 1: An HTTP Request message body contains a Sale to POI request or notification message. An HTTP Response message body contains the related Sale to POI response message, and is empty if there is no response to the request message.
- Rule 2: The Sale System may open a TCP connection for requesting a Sale to POI message exchange to the POI. The Sale System might decide to use persistent TCP connections, uses parallel TCP connections, opens a TCP connection for each exchange, or mixes any type of management according the requirements of the sales.
- Rule 3: The POI System may open a TCP connection for requesting Sale to POI device message exchange to the Sale System:
1. During the processing of a Sale System Sale-to-POI request,
 2. To send a notification.

The POI System might decide to use persistent TCP connections or not, uses parallel TCP connections but the TCP connection have to be close before sending the response to the initial Sale System request message.

3.3.3.3 Application Dialogues

To validate the rules and specify how they apply, this section presents the way HTTP is used for the application dialogues defined in section 3.4.2 *Dialogue Management*.

Service Dialogue

Services dialogues are specified in section 3.4.2.3 *Service Dialogue*.

The service dialogue respects the following process flow:

1. If there is no open TCP connection between the Sale and the POI, the Sale opens a TCP connection.
The Sale sends to the POI an HTTP Request containing the Service Request application message.
The POI process the requested Service.
2. If a Device Request must be sent during the processing the POI open a TCP connection with the Sale.
3. The POI sends to the Sale an HTTP Request containing the Device Request application message.
4. The Sale performs the device request, and sends to the POI the Device Response application message,
5. The POI disconnects the TCP connection, when no more Device application request must be exchanged with the Sale.
6. At the end of the service processing, the POI sends an HTTP Response to the Sale containing the Service

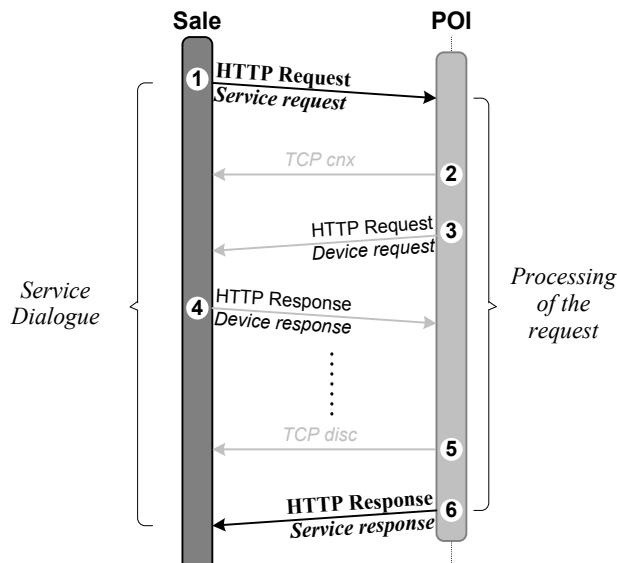


Figure 56: HTTP Service Dialogue

If there are no Device requests to send to the Sale when performing the service request, the steps 2 to 5 are skipped.

To be able for the POI system to be an HTTP client, it must know the TCP address of the Sale system requiring the Service.

During the processing of a Service dialogue, the POI Terminal can send several Device request messages in parallel, and is not required to wait the Device response message of the preceding request before sending the next one.

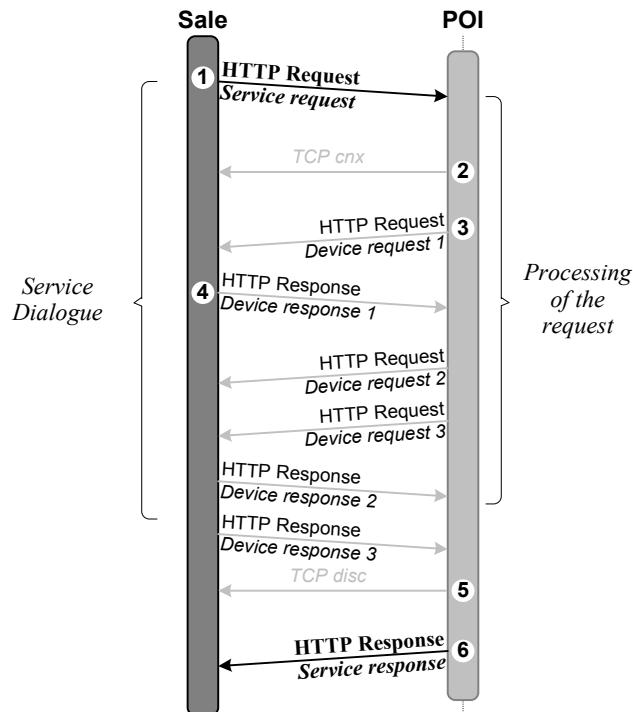


Figure 57: HTTP Service with Device Requests in Parallel

When several Device requests are sent in parallel:

- The same TCP connection might be used, as HTTP allows on a persistent connection sending a request before having received the response of the previous request. The constraint is that the (HTTP) responses are sent in the order of the requests. So if the POI starts a Print then a Display, the Display response is received after the Print response.
- To avoid this constraint, several TCP connections might be open by the POI, and the Display response is independent from the Print response received in another TCP connection.

Some Device request application messages do not require response message, only the Device Request message is exchanged.

In this case, the Device request is sent in the body of an HTTP request, a related HTTP response is sent without body and the status code 204 (No Content).

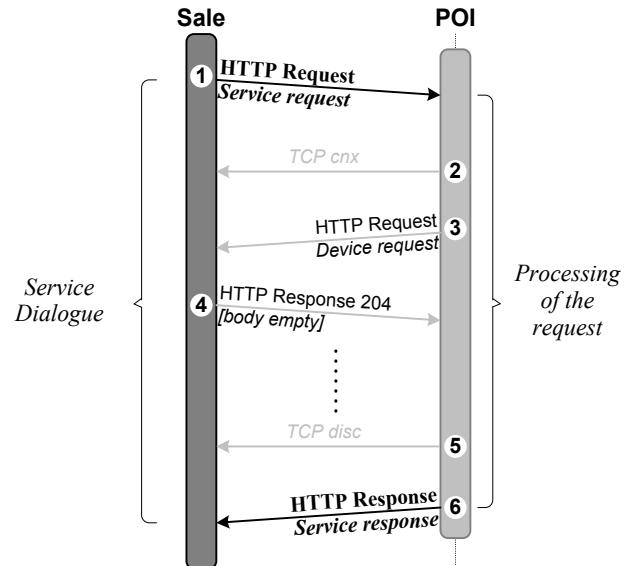


Figure 58: HTTP Service and Device Request without Response

Error Resolution Dialogue

Error resolution dialogues are specified in section 3.4.2.4 *Error Resolution Dialogue*.

The dialogue uses the Abort message defined in the section 4.7.2.3 *Abort Processing*. This message has no applicative response.

The standard abort processing with the HTTP protocol is the following:

1. The Service starts with the sending of a Service (or Device) application message request, inside an HTTP Request, by the Sale to the POI.
2. After a timeout, or an operator action, the Sale aborts the service, sending to the POI an HTTP Request containing an Abort application message.
3. An HTTP response associated with the Abort request without body (status code 204), is then sent by the POI.
4. An HTTP Response containing the Service response associated to the HTTP Request containing the Service Request.

The Abort Request has to be sent inside a different TCP connection if the Sale system wants to receive it before the Service response (both are allowed).

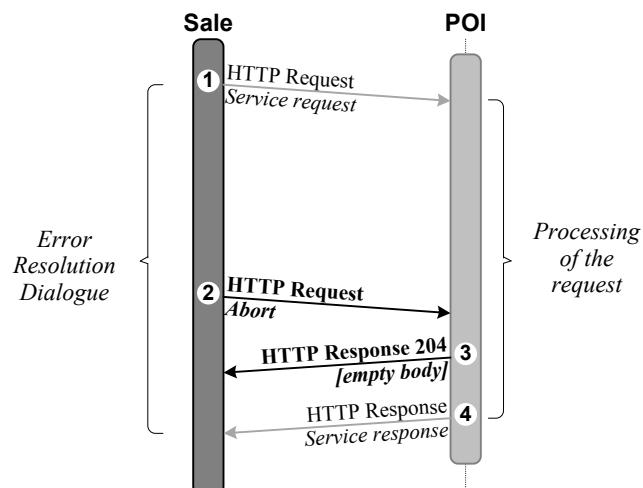


Figure 59: HTTP Error Dialogue

If the Service request to abort is not found, an Event notification (Reject) has to be sent in the body of the associated HTTP Response.

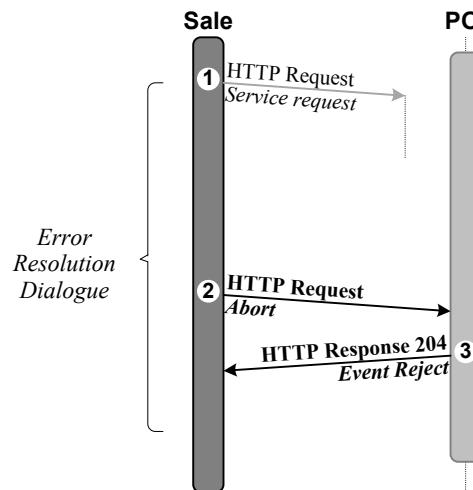


Figure 60: HTTP Abort of an Unknown Request

When the Service response is not received (e.g. disconnection), but the Service response was sent by the POI, an Event notification “Completed” has to be sent in the related HTTP Response.

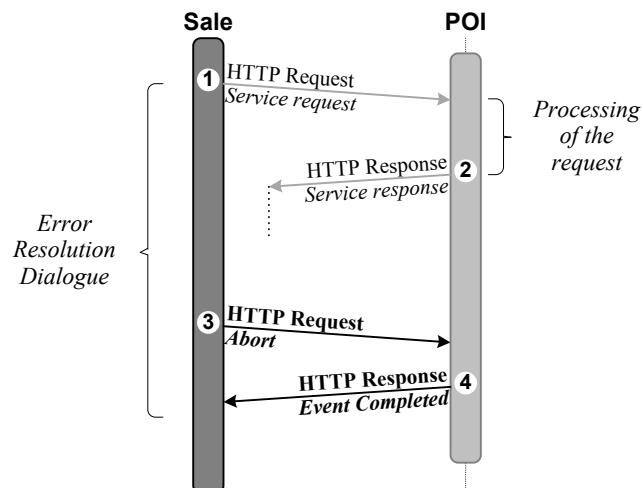


Figure 61: HTTP Error Dialogue

As a reminder, a TransactionStatus request has to be sent before any Abort as defined in the section 4.7.5.2 *Error Resolutions Specifications on the Sale System*.

The TransactionStatus request/response messages are specified in the section 4.7.1.4 *Transaction Status Processing*. They have to be sent in an HTTP Request/Response exchange.

A different connection has to be used to receive the TransactionStatus response before the Service response.

Device Dialogue

Devices dialogues are specified in section 3.4.2.5 *Device Dialogue*.

The dialogue follows the standard http Request/Response exchange containing the Device request/response messages.

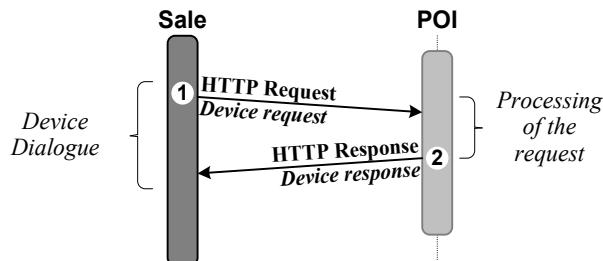


Figure 62: HTTP Device Dialogue

Notification Dialogue

Notification dialogues are specified in section 3.4.2.6 *Notification Dialogue*.

The dialogue is based on the Event Notification message defined in the section 4.7.3 *Event Notification Message*. This message has no applicative response.

If the Event is directly related to the processing of a Service or a Device, that cannot be processed as a "Reject" or "Completed", and without response (as for Abort or a reject), the event is sent directly in the HTTP Response of the Service or Device requests.

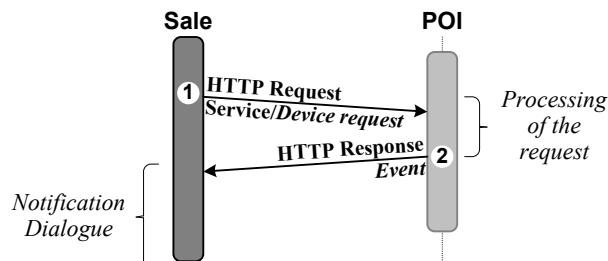


Figure 63: HTTP Exchange for a Linked Notification Dialogue

For other cases, the POI sends the Event notification in an HTTP Request, and the related HTTP Response is sent by the POI with the status code 204 and no message body.

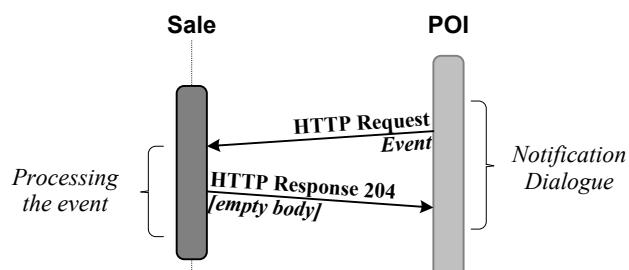


Figure 64: HTTP Exchange for an Isolated Notification Dialogue

3.3.3.4 Request Start Line

POST is the HTTP method used for the transport of Sale to POI request and response messages¹⁴.

The URI never includes the host. The path in the URI contains:

- At the first level the name of the protocol “EPASSaleToPOI”, and
- At the second level the *ProtocolVersion* “3.0”.

To be able to make a first switch of the request, the query contains the following data elements of the message header:

- the *MessageClass*, with the name “Class”
- the *MessageCategory*, , with the name “Category”
- the *SaleID*, with the name “SaleID”, and the value if the content is acceptable in a URI,
- the *POIID*, with the name “POIID”, and the value if the content is acceptable in a URI.

The version of HTTP to use is 1.1

The format of the start line of an HTTP Request is summarised in the figure below.

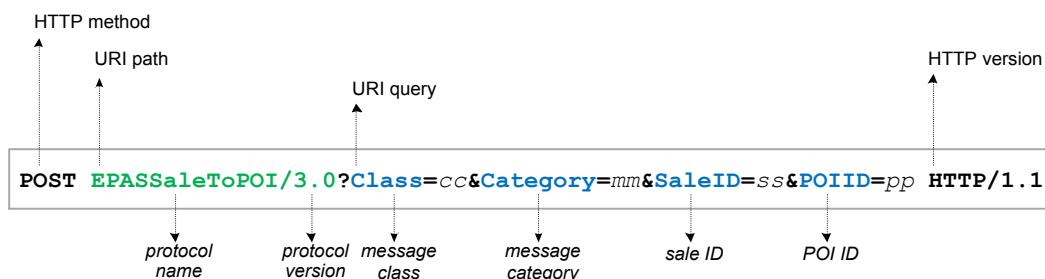


Figure 65: HTTP Request Start Line

Rule 1: All the non US-ASCII characters of the *SaleID* value or the *POIID* value must be encoded with their UTF-8 value using the percentage encoding, i.e. each byte is encoded as a characters triplet: the character “%” following by the two hexadecimal digits representing their octet’s numeric value (the digits A-F and their lower case corresponding values a-f are equivalents).

For example the character ‘é’, which has the ASCII value E9, the utf-8 value C3 A9, will be encoded in %C3%A9.

¹⁴ RESTFUL services will be added in a following version.

3.3.3.5 Error Handling and Response Start Line

The format of the start line of an HTTP Response is summarised in the figure below.

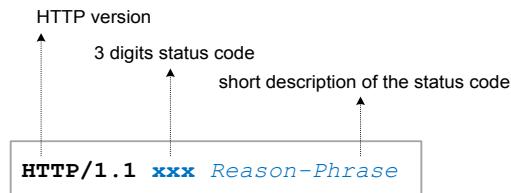


Figure 66: HTTP Response Start Line

Rule 1: Informal status codes (1xx) are not allowed.

Rule 2: Two successful status codes (2xx) are allowed:

- 1) 200 OK: when the Sale to POI message request included in the HTTP Request has been processed and a Sale to POI message response is present in the HTTP Response, whatever the response code of the Sale to POI message response.
- 2) 204 No content: when the Sale to POI message request included in the HTTP Request has been validated and processed and the HTTP Response does not contain any Sale to POI message.

Rule 3: Redirection (3xx) and Server Error (5xx) status codes are allowed without any relationship to the Sale to POI protocol.

Rule 4: Client error status codes (4xx):

- 1) 408 Request Timeout: is sent at the expiration of the TC2 timeout (see section 3.2.4.7 *Incomplete Application Message* and section 3.2.5 *State Diagrams*).
- 2) 413 Request Entity Too Large: is sent when the Content-length of the HTTP body is greater than the maximum size of the message (see section 3.2.4.4 *Message Too Big*).
- 3) All other client error status codes are related to the HTTP Request message only.

3.3.3.6 HTTP General Header Fields

Exchange of application do not use any cache functionalities of HTTP, and the storage of Sale To POI messages must be forbidden.

Rule 1: The `Cache-Control` general header field is mandatory in both HTTP Request and HTTP Responses with the value `no-store`.

The `Pragma` general header field is mandatory in HTTP Request with the value `no-cache`, for compatibility with version 1.0 of HTTP.

Rule 2: Other HTTP request header fields:

1) The following fields should be absent, they have to be ignored:

Date
Transfer-Encoding
Pragma (other than no-cache)
Upgrade
Via
Warning

2) If present, the following fields have to be perform according to the HTTP specifications:
Connection

3.3.3.7 HTTP Request Header Fields

- Rule 1: If the `Accept` HTTP request header field is present, it must have the value `text/xml`, `application/asn1` or `application/json` according to the coding of the application message.
- Rule 2: If the `Accept-Charset` HTTP request header field is present, it must have the value `utf-8`. The HTTP server has to ignore other value, and answer with the `utf-8` charset of the application response message.
- Rule 3: The `Host` HTTP request header field is mandatory. It must contain the Internet address of the Sale component or the Internet address of the POI component the Sale to POI message request is addressed.
The format of the `Host` value is:
`host [: port]`
where `host` is the IP or the dns address of the taget and,
The TCP port number `port` might be absent if the value is `80`, default TCP port number of the HTTP protocol.
- Rule 4: If the `User-Agent` HTTP request header field is present, it should have the product name and version of the products managing the Sale to POI protocol and the HTTP protocol.
- Rule 5: If the `Accept-Encoding` HTTP request header field is present, it must have the value `Identity`. The HTTP server has to ignore other value.
- Rule 6: Other HTTP request header fields:
- 1) The following fields should be absent, they have to be ignored:
`Accept-Language`
`Expect`
`From`
`If-Match`
`If-Modified-Since`
`If-None-Match`
`If-Range`
`If-Unmodified-Since`
`Range`
`Referer`
`TE`
 - 2) If present, the following fields have to be perform according to the HTTP specifications:
`Authorization`
`Max-Forwards`
`Proxy-Authorization`

3.3.3.8 HTTP Response Header Fields

Rule 1: The `Location` HTTP response header field might be used in an HTTP Response (status 3xx), to redirect an application request message to a backup component able to perform the request.

The value of the field must be an HTTP “absolute” URI, i.e. in front of the path and the query, the host address and port number if not the default.

An example is:

```
Location: http://test.epasorg.eu:8080/EPASSaleToPOI/3.0?Class=cc&Category=nnn&SaleID=ss&POIID=pp
```

Rule 2: If the `Server` HTTP request header field is present, it should have the product name and version of the products managing the Sale to POI protocol and the HTTP protocol.

Rule 3: Other HTTP response header fields:

1) The following fields should be absent, they have to be ignored:

Accept-Ranges
Age
ETag
Vary
Retry-After

2) If present, the following fields have to be perform according to the HTTP specifications:

Proxy-Authenticate
WWW-Authenticate

3.3.3.9 HTTP Entity Header Fields

Rule 1: If the `Allow` entity header field is present in an HTTP request, the value must contain `POST`. If an HTTP request does not use the `POST` method, the `Allow` entity header must be present in an HTTP response (status `405`), with the value `POST`.

The HTTP protocol is a document oriented protocol which offers message delimitation or data-unit delimiting. It uses the `Content-Length` header field to provide the number of bytes of the HTTP request or response body where is located the Sale to POI message.

Rule 2: If the HTTP Request or Response contains an application message, the `Content-Length` HTTP entity header field must be present. The value is the number of byte of the application message.
If the HTTP Response does not contain any application message, the `Content-Length` HTTP entity header field must be absent or present with the value `0`. In this case the status code of the response stat line must be: `204 No content`.

Rule 3: In any HTTP Request or Response containing an application message, the `Content-Type` HTTP entity header field must be present. It must have the value `text/xml ; charset=utf-8, application/asn1 ; charset=utf-8 or application/json ; charset=utf-8` according to the coding of the application message. In case of unappropriate `Content-Type`, the application error handling has to be performed.

Rule 4: Other HTTP entity header fields:

- 1) If present, the following fields have to be perform according to the HTTP specifications:
`Content-MD5`
- 2) The following fields should be absent, they have to be ignored:
`Content-Encoding`
`Content-Location` (initial location of the “resource”)
`Content-Range`
`Expires`
`Last-Modified`
and any other non HTTP 1.1 extension header.

3.3.3.10 Addressing

To establish a TCP connection, HTTP needs a TCP protocol transport address. This TCP address is composed of:

1. The *IP Address* or the *DNS Name* to resolve of the host on which the application protocol lives.
2. The *TCP Port*, to dispatch the connections to the application, the default TCP port of HTTP is 80.

3.3.4 HTTPS Protocol

3.3.4.1 Introduction

HTTPS, or HTTP over SSL/TLS is defined in the RFC 2818 (HTTP over TLS) issued in May 2000.

HTTPS protocol is defined as the HTTP protocol, end to end protected by the SSL or TLS protocol, the application security protocol over TCP, as presented in the figure below.

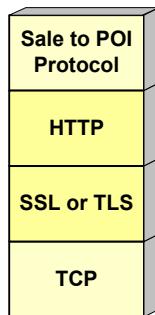


Figure 67: HTTPS Protocol

In the following of the chapter, we will use the name TLS instead of SSL/TLS. SSL is now unrecommended and use of SSL and early TLS (TLS 1.0) should be avoid.

The protocol HTTPS provides the same interface than the protocol HTTP, however, the usage of TLS which has been designed to be managed by the application requires some additional process.

The section 2 presents the characteristics of the application dialogues over HTTPS, which are quite similar to the application dialogues over HTTP, replacing TCP by TLS.

The section 3 lists the constraints of TLS on HTTP.

The section 4 presents the transport and security services which are offered by TLS. It provides rules and recommendations for the usage of these services.

The last section presents the X.509 certificate requirements, their validation and the PKI management.

3.3.4.2 Application Dialogues

As presented in the previous chapter (section 3.3.3.1), an HTTP client initiates connections and sends HTTP request messages.

As presented in the following section (3.3.4.4 TLS Services), a TLS client initiates connections, initiates and process the TLS handshakes¹⁵, potentially reusing an existing TLS session.

After a successful handshake, the data flow is not qualified for TLS, and the application protocol may send any type of message or data, using whatever type of dialogue.

The standard HTTPS sequence flow is presented in the figure below:

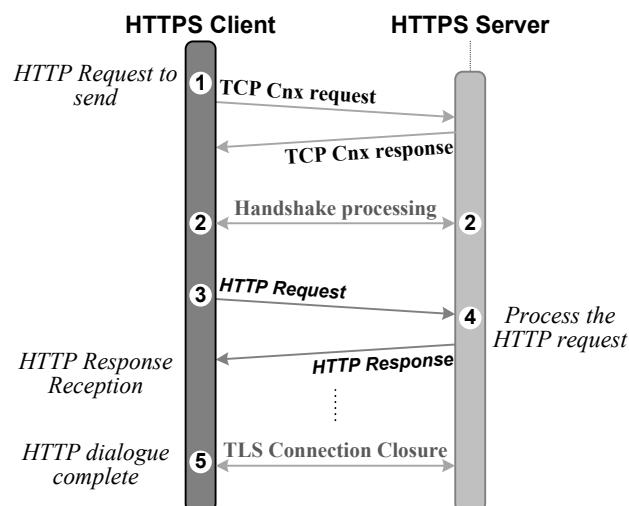


Figure 68: Standard HTTPS Sequence Flow

1. The application protocol requests to send an *HTTP request* message. Then the HTTPS Client open a TCP connection with the HTTPS Server,
2. The HTTPS Client initiates and processes with the HTTPS server a TLS handshake, potentially reusing an existing TLS session,
3. The HTTPS Client sends the *HTTP Request* message in a TLS data record,
4. The HTTPS Server process the *HTTP request* of the client, and sends the *HTTP Response* message containing the result of the processing in a TLS data record. If the connection is persistent, the HTTPS Client might send other *HTTP Request* messages,
5. The HTTPS client closes the connection, resulting in a TLS Close Notify alert and a TCP disconnection.

As the HTTPS client initiates a TLS connection, sends HTTP request, and accepts only HTTP responses, the TLS session cannot be used by the HTTPS server to send HTTP requests to the HTTP client.

Rule 1: The HTTPS client must be the TLS client. The TLS session established by an HTTPS client has to be used by the HTTPS server to answer to HTTP requests.

¹⁵ TLS handshake negotiates the security algorithms, exchanges the shared secret between the TLS client and the TLS server.

As a consequence, using HTTPS transport services, the Sale to POI protocol dialogue must follow the dialogue defined by the HTTP transport services (section 3.3.3.3 *Application Dialogues*).

Service Dialogue

The general process flow of the service dialogue follows the *Figure 56: HTTP Service Dialogue*, replacing TCP by TLS:

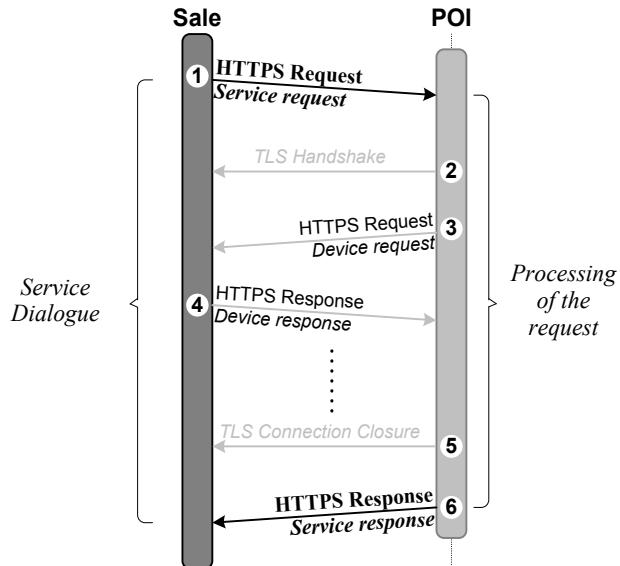


Figure 69: HTTPS Service Dialogue

If there are Device requests to send to the Sale during the performing of the service request (steps 2 to 5), the POI must establish a TLS session to send the Device request message to the Sale which plays then the role of a TSL server.

Conforming to the *Figure 57: HTTP Service with Device Requests in Parallel*, when several Device request are sent in parallel:

- The same TLS session might be used, as HTTP allows on a persistent connection sending a request before having received the response of the previous request. The constraint is that the (HTTP) responses are sent in the order of the requests. So if the POI starts a Print then a Display, the Display response is received after the Print response.
- To avoid this constraint, several TLS sessions might be open by the POI, and the Display response is independent from the Print response received in another TLS session.

For Service or Device request that not require response message (see *Figure 58: HTTP Service and Device Request without Response*), the Device request is sent in the body of an HTTP request, a related HTTP response is sent without body and the status code 204 (No Content).

Error Resolution Dialogue

Error resolution dialogues are conforming to the HTTP error resolution dialogues specified in *Error Resolution Dialogue* at page 85.

Device Dialogue

Device dialogues are conforming to the HTTP device dialogues specified in *Device Dialogue* at page 87.

Notification Dialogue

Notification dialogues are conforming to the HTTP notification dialogues specified in *Notification Dialogue* at page 87.

3.3.4.3 HTTPS Constraints

HTTPS is conforming to the start line rule defined in the section *3.3.3.4 Request Start Line*.

HTTPS is conforming to the error handling rules and to the response start line rules defined in the section *3.3.3.5 Error Handling and Response Start Line*.

HTTPS is conforming to the general header fields rules defined in the section *3.3.3.6 HTTP General Header Fields*.

HTTPS is conforming to the request header fields rules defined in the section *3.3.3.7 HTTP Request Header Fields*.

HTTPS is conforming to the response header fields rules defined in the section *3.3.3.8 HTTP Response Header Fields*.

HTTPS is conforming to the entity header fields rules defined in the section *3.3.3.9 HTTP Entity Header Fields*.

To establish a TCP connection, HTTPS needs a TCP protocol transport address. This TCP address is composed of:

1. The *IP Address* or the *DNS Name* to resolve of the host on which the application protocol lives.
2. The *TCP Port*, to dispatch the connections to the application, the default TCP port of HTTPS is 443.

3.3.4.4 TLS Services

The SSL (Secure Socket Layer) protocol has been specified by Netscape, the last version being SSL v3.0.

The TLS (Transport Layer Security) protocol has been specified by IETF:

- RFC 2246 - The TLS Protocol version 1.0 – January 1999
- RFC 4346 - The Transport Layer Security (TLS) Protocol version 1.1 – April 2006
- RFC 5246 - The Transport Layer Security (TLS) Protocol version 1.2 – August 2008

The SSL version 2.0 has known deficiencies and does not provide a sufficient level of security, the RFC 6176 describes these deficiencies.

Rule 1: When TLS clients and servers establish a connection, they must never negotiate the use of SSL version 2.0. In particular the version 2.0 of the handshake initial message (Client Hello) must never be sent by a TLS Client.

TLS is designed to provide a secure end-to-end service with TCP. TLS is composed of two layers of protocols, as presented in the figure below.

1. The top level with the following protocols:
 - a. The *TLS Handshake Protocol* which allows the negotiation of cryptographic algorithms and keys, and client and server authentication,
 - b. The *TLS Change Cipher Spec Protocol* processing the update of cryptographic algorithms and keys, just negotiated,
 - c. The *TLS Alert Protocol* which convey warnings or fatal errors.
2. The *TLS Record Protocol* which provides the format of the message, including application data, on which the security services are performed.

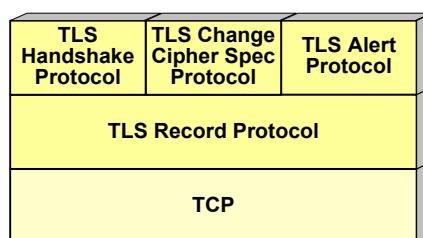


Figure 70: TLS Protocol Layers

The *TLS Record Protocol* performs the operations of fragmentation, compression, MACing and encryption

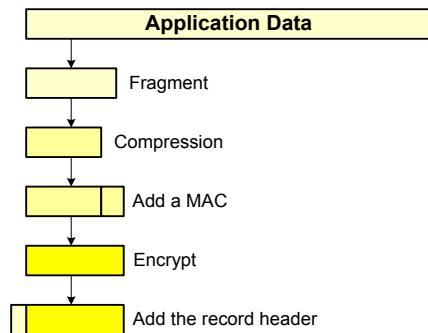


Figure 71: TLS Record Protocol

TLS Session

When a TLS client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public-key encryption techniques to generate shared secrets.

These processes are performed in the handshake protocol and the result is the establishment of a *TLS session*.

1. After connecting to the TLS Server, the TLS Client sends a *ClientHello* message containing the sequence of cryptographic algorithms it can manage.
2. Then the TLS Server sends a *ServerHello* message containing the selected algorithm. It sends a *Certificate* message, either for key exchange, or to sign a public key for encryption with the *ServerKeyExchange* message. It sends a *CertificateRequest* message to request a client authentication with a certificate. The *ServerHelloDone* message terminates the Server initialisation phase.
3. To be authenticated, the TLS Client sends the *Certificate* message containing its authentication certificate, and the *CertificateVerify* message containing the signature of a Server challenge. The Client sends the *ClientKeyExchange* message containing the pre-master secret, encrypted by the server public key, to generate the keys of the session. Then the Client sends the *ChangeCipherSpec* message to switch to the new set of keys and cryptographic algorithms, and the *Finished* message which is the 1st message protected by these news keys, and which terminates the handshake for the Client.
4. In turn, the TLS Server sends the *ChangeCipherSpec* message, the *Finished* message, and which terminates the handshake.
5. Application data may then be exchanged between the TLS Client and the TLS Server without any constraint on the type of dialogue.

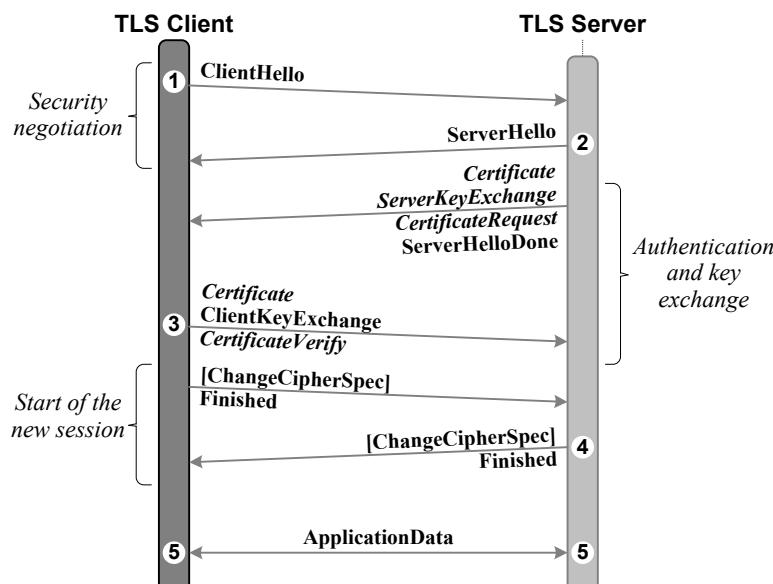


Figure 72: New TLS Session (Full Handshake)

Rule 2: It is recommended to use the same TLS Server certificate for authentication and encryption of the pre-master secret.

In the *ServerKeyExchange* message, the public encryption key cannot be presented as a certificate, and is protected by the signature of the public key of the authentication certificate.

When the Client and Server decide to resume a previous session or duplicate an existing session (instead of negotiating new security parameters), the handshake process is simplified.

1. After connecting to the TLS Server, the TLS Client sends a *ClientHello* message containing the session identification he wants to resume.
 2. Then the TLS Server sends a *ServerHello* message containing the selected algorithm. It sends the *ChangeCipherSpec* message to switch to the new set of keys and cryptographic algorithms, and it sends the *Finished* message which terminates the handshake resume.
 3. Then the Client sends the *ChangeCipherSpec* message and the *Finished* message which terminates the handshake resume.
 4. Application data may then be exchanged between the TLS Client and the TLS Server without any constraint on the type of dialogue.

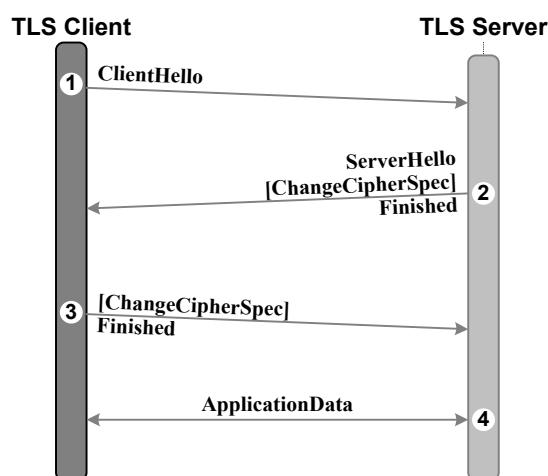


Figure 73: Resume Previous TLS Session

Rule 3: TLS Server shall manage for each session the time limit of validity of the session, typically 8 hours.

After this time limit, the TLS Server must refuse any resuming of the session, and send a *HelloRequest* message if the session is active to force the TLS Client sending a *Hello* message.

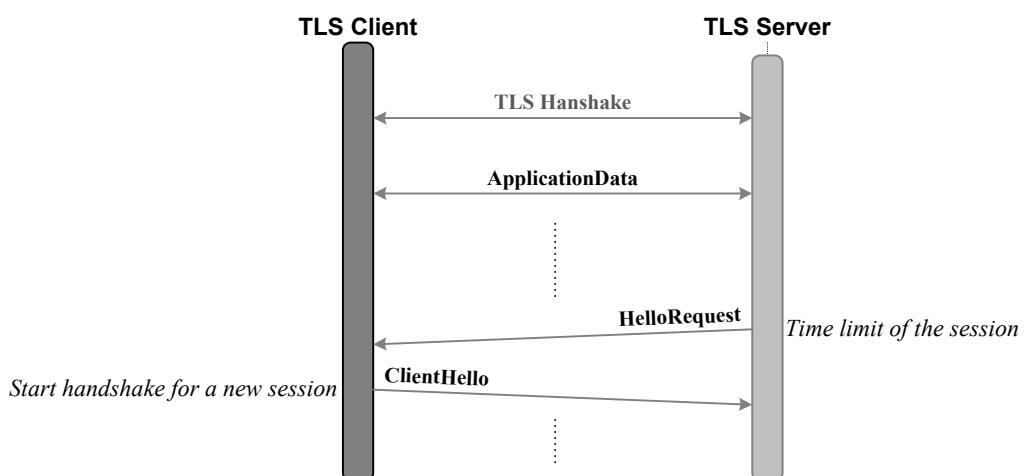


Figure 74: Force the End of a TLS Session

Client and Server Authentication

Rule 4: TLS Client and TLS Server must implement TLS Server authentication. TLS Client authentication is optional, depending on the context.

In the section 3.3.4.2, when the POI implementation sends Device request messages or Event messages or initiates connection, the POI must play the role of a TLS Client. As TLS Server authentication is mandatory, the Sale System must manage security for its own authentication.

Rule 5: When POI Device request messages or Event messages is accepted by the Sale system, mutual authentication is required by TLS Client and Server.

Cryptographic algorithms

Rule 6: The recommended type of key exchange is RSA.

It requires a TLS Server certificate for authentication, and the encryption of a pre-master-secret by the key of the certificate or a temporary RSA key to generate the keys.

Rule 7: The minimum recommended of cryptographic algorithms are Triple DES with a triple key length, CBC encryption mode, and CBC with SHA1 MAC algorithm.

3.3.4.5 TLS Certificates

The X.509 certificates management has been specified by IETF in April 2002 by the RFC 3280 "Internet X.509 Public Key Infrastructure Certificate (PKI) and Certificate Revocation List (CRL) Profile". This specification is often referred as PKIX (Internet PKI).

An X.509 certificate is coded in ASN.1 and is composed of:

- The certificate's data in the structure *CertificateTBS* (certificate to be signed),
- The identification of the signature algorithm,
- The signature of *CertificateTBS*, with a specific format, different from PKCS#7.

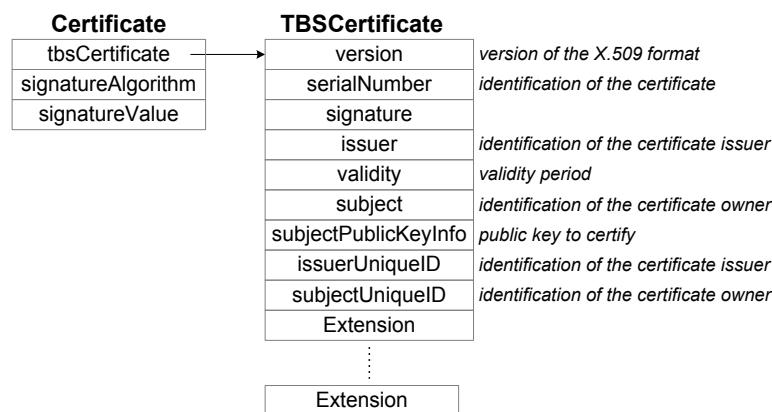


Figure 75: X.509 Certificate Format

Rule 1: X.509 version 3 of the certificate is mandatory.

Certificate Path Validation

The certificate path is the sequence of certificate starting with the certificate of the root Certificate Authority and ending with the certificate of the public key to validate. Each certificate of the path signs the following certificate, except the first one (root certificate) which is auto-signed.

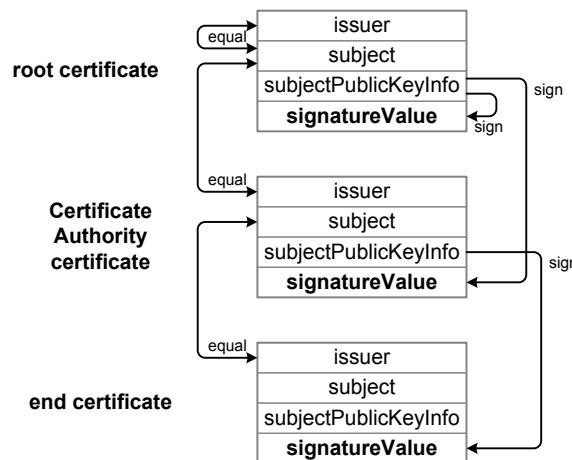


Figure 76: Certificate Path

The PKI (Public Key Infrastructure) of the Sale system and the POI system could be:

- Independent, where the root of the Sale certificate and the root of the POI certificate are not the same.
- Shared, where the root of the Sale certificate and the root of the POI certificate are the same, or
- With cross-certificates (network topology), a CA certificate of the Sale PKI is signed by a CA of the POI, and the opposite.

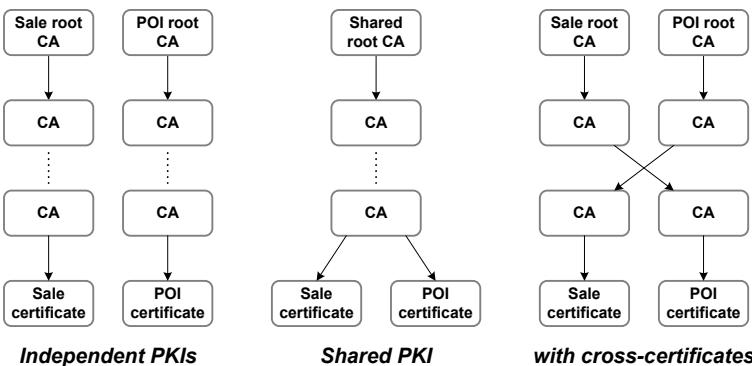


Figure 77: PKI Topologies

Rule 2: No topology of the Sale and PKI is recommended, independent, shared, or with cross-certificates.

The complete certificate path validation has to be verified by the TLS Client and the TLS Server. In other words, the validator of a certificate path needs to have stored securely the root of the certificate path.

Rule 3: The validator of a certificate path must store securely the root of the certificate path.

Certificate Information

Information contained in the extensions of the certificate contains information and property of the certificate.

- Rule 4: The public key usage must appear in one of the following certificates extensions:
The Key Usage certificate extension (`id-ce-keyUsage ::= { id-ce 15 }`) with the digitalSignature or keyEncipherment value.
The Netscape certificate type extension (`netscape-cert-type ::= { netscape-cert-extension 1 }`) with the TLS client or TLS server value.
- Rule 5: The public key owner authentication must be performed, for instance with the Subject Alternative Name certificate extension (`id-ce-subjectAltName ::= { id-ce 17 }`) with the dNSName or iPAddress value.

Certificate Revocation List (CRL)

RSA key length and certificates validity period must be consistent with the security rules of the POI system.

- Rule 6: Management of CRL or real-time validation of certificates is optional.

3.3.5 Sale to POI Protocol over TLS

3.3.5.1 *Introduction*

This section specifies the interface between the Sale to POI and the SSL/TLS security protocol, i.e. when no HTTP protocol is used above SSL/TLS.

For security reason, the Sale to POI protocol between a Sale Server and a POI Server could use SSL/TLS without the HTTP protocol for simplification of the application dialogues.

SSL/TLS is an application protocol between the Sale to POI protocol and the TCP transport protocol. This protocol was presented in a previous section (3.3.4.4 *TLS Services*).

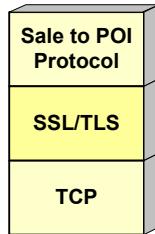


Figure 78: Sale to POI Protocol over SSL/TLS

As in the previous section, we will use in this section the name TLS instead of SSL/TLS.

3.3.5.2 Transport Service Constraints

Message Delimitation

The TLS protocol, as TCP, does not provide any message delimitation or data-unit delimiting, and the Sale to POI protocol uses the message delimitation as specified in 3.3.2.2 *Message Delimitation*.

TLS Session

To allow a maximum of flexibility, the Sale System has the entire responsibility to manage the TLS session at its convenience, and during the time it considers necessary to its own management.

The Sale System might decide to use permanent transport connections, to open a TLS session for each message, or to mix any type of management according to the requirements of the sales.

TLS Session Creation or Resume

- Rule 1: The Sale System can initiate or resume a TLS session for requesting an application message exchange with the POI. If successful, the Sale System is then responsible for initiating the application exchange.
- Rule 2: The POI System can initiate or resume a TLS session for requesting an application message exchange with the Sale System. If successful, the Sale System is then responsible for initiating the application exchange. If connection fails, POI system must retry indefinitely to connect to Sale System with an adaptative period between connection attempts.

TLS Session Release

The Sale System may release a TLS session whenever the application judges it appropriate. The Sale System has to estimate the need for receiving the last response message before to close the TLS session.

- Rule 3: The Sale System may close a TLS session at any time.
The POI System may close the TLS session only in case of error.

When the TLS session is closed, the POI System cannot send unsolicited notification to the Sale System.

Addressing

To establish TLS sessions, TLS needs a TCP protocol transport address. This TCP address is composed of:

1. The *IP Address* or the *DNS Name* to resolve of the host on which the application protocol lives.
2. The *TCP Port*, to dispatch the connections to the application.

3.3.5.3 Application Dialogues

The standard sequence flow is presented in the figure below:

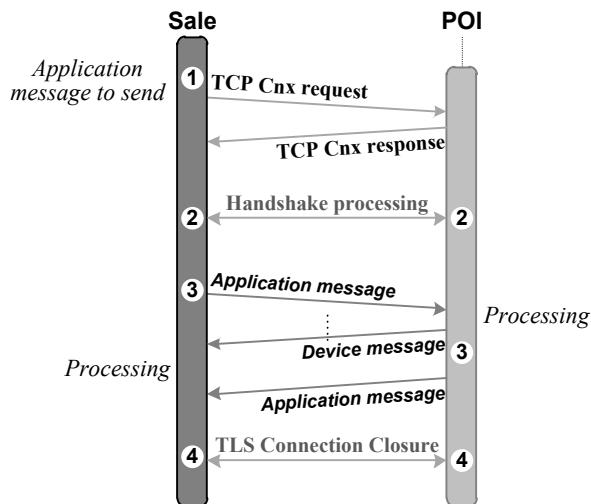


Figure 79: Standard Sequence Flow

1. The Sale System requests to send an application message, and open a TCP connection with the POI System,
2. The Sale System as a TLS Client initiates and processes a TLS handshake, potentially reusing an existing TLS session,
3. The Sale System and the POI System exchanges messages of the Sale to POI protocol on this TLS session, Any sequence of request, devices command or admin message may be exchanged.
4. The Sale System or the POI System closes the connection, resulting in a TLS Close Notify alert and a TCP disconnection.

All the application dialogues use a TLS session initiated or resumed by the Sale System as TLS Client.

3.3.5.4 TLS Services and Certificates

The Sale to POI protocol over TLS uses the TLS services as specified in the section 3.3.4.4 *TLS Services*, except the Rule 5 which must be removed.

The Sale to POI protocol over TLS uses the TLS certificates as specified in the section 3.3.4.5 *TLS Certificates*.

3.4 Messages Management

This section specifies the management of the messages, with the information located in the Message Header:

- It starts with the section *Messages Organisation*, which presents the decoding of the messages, the structures of messages and the types of messages. In addition, the section describes the content of the Message Header and the common information in response message reporting the result of the request processing.
- The section 3.4.2, *Dialogue Management* presents all the type of exchanges and dialogs depending on type of messages and the initiator of the dialog. It also includes general state diagrams to specify the behavior of each system.
- The following sections are dedicated to the specification of:
 - The management of protocol versions (3.4.3, *Protocol Version Management*),
 - The identification of the origin and the destination of the messages (3.4.4, *Origin and Destination of Messages*),
 - The identification of message exchanges (3.4.5, *Identification of Message Exchanges*).
- The chapter ends with the section 3.4.6, *Message Security* which presents the protection of messages, the specifications of the cryptographic mechanisms terminates the *Message Management* chapter.

3.4.1 Messages Organisation

3.4.1.1 Message Coding and Decoding

There are three types of data coding for the messages:

- **XML/Schema:** the schema definition files (xsd files) are provided with messages examples in addition to the specifications. The schema definitions are automatically generated from the data dictionary included in this specification. Particularity of the XML coding are:
 - The use of XML attributes to reduce the volume of message and improve its readability.
 - When a data structure has only attributes, at the exception of one data element (e.g. *TrackData*), this data element has no tag and its value is the value of the data structure tag.
TrackData contains the attributes *TrackNumb*, *TrackFormat* and the only element *TrackValue*. An example of *TrackData* encoded data structure is:
`<TrackData TrackNumb="1" TrackFormat="ISO">>%B4970100202013617^CARTE DE DEVELOPPEMENT`1105901958097000000000000000000?</TrackData>`
 - **ASN.1/DER:** the ASN.1 modules definitions are provided in addition to the specifications. The DER (Distinguish Encoding Rules) is used to avoid several encoding of the same value, as it is the case in BER. Particularity of the ASN.1 coding is to use a specific tag of the class APPLICATION to identify every data structure and data elements.
 - **JSON:** JSON does not specify any language data definition, at the exception of the basic types defined by the JSON grammar. There is an automatic translation of the types defined by EPAS in the basic JSON types. The JSON coding is specified in the section 3.4.1.6 *JSON Messages Coding/Decoding*.

A design rule of the data definition is to have a "one to one" translation between the various coding, i.e. any value in one coding have one and only one value representation in the other coding.

Parsing errors or warnings involve a response containing *ErrorCondition* with the value "MessageFormat" (see section 4.6.2.1 *Message Format*).

Rule 1: If a message request or notification cannot be decoded and recognised by the receiver, an EventNotification message must be sent with *EventToNotify* = "Reject" and the content of the wrong message in the data element *RejectedMessage*.
If a message response cannot be decoded and recognised by the receiver, it has to be ignored.

Rule 2: If a data element or data structure is not known, this data is ignored. During a debugging or testing phase, a warning could be included in the response of the message with, *ErrorCondition* set to "MessageFormat". see section 4.6.2.1.13 *Unknown Data*.

Rule 3: Other parsing errors use the *ErrorCondition* set to "MessageFormat" and *AdditionalResponse* containing:

General Parsing Error: part of the message cannot be parsed.

General Parsing Error: part of the message cannot be parsed; Invalid Number of Repetitions: the number of repetitions of the data exceeds the limit

Invalid Number of Repetitions: the number of repetitions of the data exceeds the maximum allowed.

Invalid Enumerated Value: the value is not belonging to the set of enumerated

Data Size: the size exceeds the maximum value of the implementation.

An empty cluster, i.e. a cluster without value (XML list empty, a JSON empty string, or ASN.1 BIT STRING with all bits to 0) is accepted¹⁶.

It is recommended to encode messages in the XML canonical form, where all the useless spaces, tabulations, ends of line and XML comments are stripped.

Examples of XML/Schema coded messages are provided in *Annex B*.

Rule 4: For an XML encoding of the messages:

All the implementations must accept version 1.0 of the XML recommendation. Version 1.1 is optional.

The only required character encoding of XML messages is UTF-8. Any implementation may refuse another encoding of characters.

¹⁶ For an XML coded cluster, the repetition of the same enumerated value is ignored (e.g. “value1 value2 value1” is considered as equivalent to “value1 value2”).

For a JSON coded cluster, the repetition of the same enumerated value is ignored as well.

3.4.1.2 Message Structure

The messages of the Sale to POI protocol share a common structure:

- The *Message Envelope*, which is a data container for the whole message. The envelope identifies the protocol and the position inside a message pair: the request or the response.
- The *Message Header*, which contains information dedicated to the management of the Sale to POI protocol. All the messages of the protocol contain the same message header structure.
- The *Message Body*, where the data related to the service are grouped. Each message has a different message body structure.
- An optional *Security Trailer*, containing the protection of the whole message.

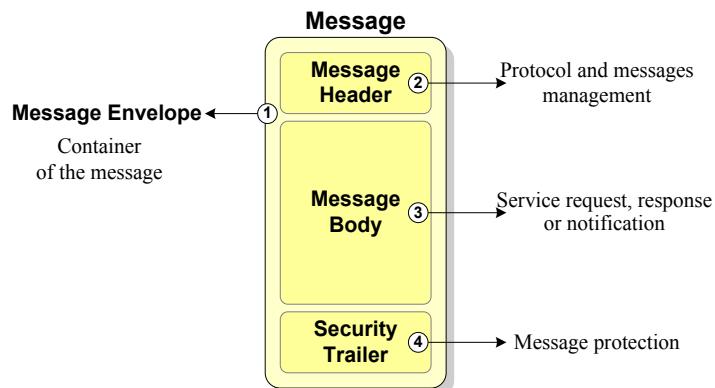


Figure 80: Sale to POI Message Structure

The envelope of the message request and the message response are referenced in the Data Dictionary respectively by *SaleToPOIRequest* and *SaleToPOIResponse*.

The message header is referenced in the Data Dictionary by *MessageHeader*.

The messages bodies are referenced in the Data Dictionary by:

- *RequestMessageBody*, for a request or notification message, which is a choice between the following data structures: *AbortRequest*, *AdminRequest*, *BalanceInquiryRequest*, *BatchRequest*, *CardAcquisitionRequest*, *CardReaderAPDUREquest*, *CardReaderInitRequest*, *CardReaderPowerOffRequest*, *DiagnosisRequest*, *DisplayRequest*, *EnableServiceRequest*, *EventNotification*, *GetTotalsRequest*, *InputRequest*, *InputUpdate*, *LoginRequest*, *LogoutRequest*, *LoyaltyRequest*, *PaymentRequest*, *PINRequest*, *PrintRequest*, *ReconciliationRequest*, *ReversalRequest*, *SoundRequest*, *StoredValueRequest*, *TransactionStatusRequest*, *TransmitRequest*,
- *ResponseMessageBody*, for a response message, which is a choice between the following data structures: *AdminResponse*, *BalanceInquiryResponse*, *BatchResponse*, *CardAcquisitionResponse*, *CardReaderAPDUREsponse*, *CardReaderInitResponse*, *CardReaderPowerOffResponse*, *DiagnosisResponse*, *DisplayResponse*, *EnableServiceResponse*, *GetTotalsResponse*, *InputResponse*, *LoginResponse*, *LogoutResponse*, *LoyaltyResponse*, *PaymentResponse*, *PINResponse*, *PrintResponse*, *ReconciliationResponse*, *ReversalResponse*, *SoundResponse*, *StoredValueResponse*, *TransactionStatusResponse*, *TransmitResponse*.

3.4.1.3 Messages Typology

Messages are defined and specified by category and type.

Message Category

There are 27 categories of messages referenced in the Data Dictionary by *MessageCategory* and listed below by alphabetical order:

Abort	Admin	Balance Inquiry	Batch	Card Acquisition
Card Reader APDU	Card Reader Init	Card Reader Power Off	Diagnosis	Display
Enable Service	Event	Get Totals	Input	Input Update
Login	Logout	Loyalty	Payment	PIN
Print	Reconciliation	Reversal	Sound	Stored Value
Transaction Status	Transmit			

Message Type

Those message categories use 3 types of messages referenced in the Data Dictionary by *MessageType* and listed below:

- Request
- Response
- Notification

Combinations of Message Categories and Types

At the exception of *Abort*, *Event*, and *Input Update* all these message categories have a specific message type for the request and a specific message for the response. For instance there are two *Payment* messages *Payment Request* and *Payment Response*.

At the opposite there is only the message *Abort Request* (without an *Abort Response*), only the message *Event Notification* (without response), and only the message *Input Update* (without response).

So there are 51 different combinations of message categories and types listed below by alphabetical order:

Abort Request	Admin Request	Admin Response
Balance Inquiry Request	Balance Inquiry Response	Batch Request
Batch Response	Card Acquisition Request	Card Acquisition Response
Card Reader APDU Request	Card Reader APDU Response	Card Reader Init Request
Card Reader Init Response	Card Reader Power Off Request	Card Reader Power Off Response
Diagnosis Request	Diagnosis Response	Display Request
Display Response	Enable Service Request	Enable Service Response
Event Notification	GetTotals Request	GetTotals Response
Input Request	Input Response	Input Update
Login Request	Login Response	Logout Request
Logout Response	Loyalty Request	Loyalty Response
Payment Request	Payment Response	PIN Request
PIN Response	Print Request	Print Response
Reconciliation Request	Reconciliation Response	Reversal Request
Reversal Response	Sound Request	Sound Response
Stored Value Request	Stored Value Response	Transaction Status Request
Transaction Status Response	Transmit Request	Transmit Response

The category and the type of message are included in the header of the message in the data structure *MessageHeader* (see the next section).

It allows knowing the category of request for a request message, the response to a previous request message, and unsolicited notifications.

In addition the message category could be present in the body of the following messages,

- The *Transaction Status Request and Response*, to know the category of transaction for which the status is requested: Payment, Loyalty, StoredValue, Reversal, CardAcquisition, Batch, CardReaderAPDU, or Transmit.
- The *Abort Request*, to know the transaction to abort: BalanceInquiry, CardAcquisition, Display, Input, Loyalty, Payment, Batch, CardReaderInit, CardReaderPowerOff, Reconciliation, Reversal, StoredValue, or Transmit.
- The *InputUpdate*, to verify the Input message display to update.

Rule 1: Message type and category have to be consistent to the message body of the request or the response.

For a message request, a message response of the same category have to be sent with *Result*=”Failure” and *ErrorCondition*=”MessageFormat” (see section 4.6.2.1.3 *Unexpected Data*).

For a message response (without a related request sent before) or a notification, it has to be ignored.

3.4.1.4 Message Header Content

The message header is the data structure *MessageHeader*, common to all messages. It contains the following information:

1. The higher version of the Sale to POI protocol managed by the component sending the message: *ProtocolVersion*.
2. The kind of message: *MessageClass*, *MessageCategory*, and *MessageType*.
3. The identification of the message or the message pair inside the flow of messages: *ServiceID* and *DeviceID*.
4. The logical identification of the origin and the destination of the message or the message pair: *SaleID* and *POIID*.

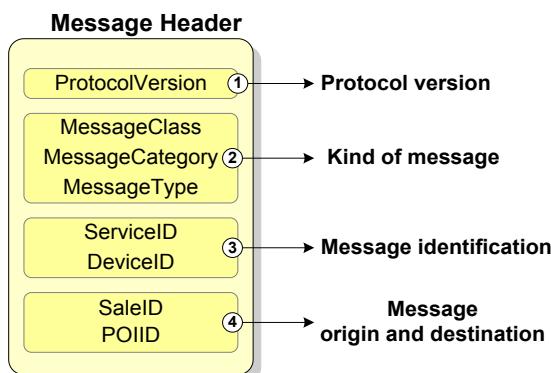


Figure 81: Message Header Information

Data Element	Definition
ProtocolVersion	Version of the Sale to POI protocol specifications, this document (see 3.4.3 <i>Protocol Version Management</i>).
MessageClass	Class of message dialogue (see 3.4.2 <i>Dialogue Management</i>).
MessageCategory	Category of message (see 3.4.1.3 <i>Messages</i>).
MessageType	Type of message of the Sale to POI protocol (see 3.4.1.3 <i>Messages</i>)
ServiceID	Identification of a message pair, which processes a transaction (see 3.4.5 <i>Identification of Message Exchanges</i>).
DeviceID	Identification of a device message pair (see 3.4.5 <i>Identification of Message Exchanges</i>).
SaleID	Identification of a Sale System or a Sale Terminal (see 3.4.4 <i>Origin and Destination of Messages</i>).
POIID	Identification of a POI System or a POI Terminal (see 3.4.4 <i>Origin and Destination of Messages</i>).

Table 5: Message Header Data Elements

Some of these data elements are optional in the message header of a request message or the *Event Notification*.

Data Element	Presence Condition
ProtocolVersion	Mandatory for Login request, absent for the others (see 3.4.3 <i>Protocol Version Management</i>).
MessageClass	Mandatory, see 3.4.2 <i>Dialogue Management</i>
MessageCategory	Mandatory, see 3.4.1.3 <i>Messages</i>
MessageType	Mandatory, see 3.4.1.3 <i>Messages</i>
ServiceID	see 3.4.2 <i>Dialogue Management</i>
DeviceID	see 3.4.2 <i>Dialogue Management</i>
SaleID	Mandatory, see 3.4.4 <i>Origin and Destination of Messages</i>
POIID	Mandatory, see 3.4.4 <i>Origin and Destination of Messages</i>

Table 6: Message Header Presence Condition for a Request

Some of these data elements are optional in the message header of a response message or the value of the request is copied.

Data Element	Presence or Value Condition
ProtocolVersion	Mandatory for Login response, absent for the others (see 3.4.3 <i>Protocol Version Management</i>)
MessageClass	Copy value from the Request.
MessageCategory	Copy value from the Request.
MessageType	Mandatory, correspond to the response of the message request.
ServiceID	If present in the request, copy value from the Request.
DeviceID	If present in the request, copy value from the Request.
SaleID	Copy value from the Request.
POIID	Copy value from the Request.

Table 7: Message Header Presence and Value Condition for a Response

3.4.1.5 Message Body Response

The message body holds a different content for each message type. However, the first element of the body of a message response is the data structure *Response* which contains the result of the processing requested in the message request.

Data Element	Definition
Result	Global result of the processing of the message request (success or failure).
ErrorCondition	Condition that has produced a failure, allowing resolution of the failure by the requestor.
AdditionalResponse	Additional information related to the result of a message request processing.

Table 8: Response Data Element

The presence condition of these data elements is presented in the table below.

Data Element	Presence Condition
Result	Mandatory for all response messages.
ErrorCondition	Mandatory, if Result is "Failure".
AdditionalResponse	Optional whatever the Result value.

Table 9: Response Presence Condition

The messages *Display Response*, *Input Response*, and *Print Response* messages have not the *Response* data structure directly as the first element of the message body. As the request could contain several display, input or print requests, the result of these multiple requests (the *Response* data structure), is included in a global structure which could be repeated (the data structures *OutputResult* and *InputResult*).

3.4.1.6 JSON Messages Coding/Decoding

JSON (JavaScript Object Notation) is a quite simple text data coding for data interchange based on value separator as XML.

The data coding is specified in the RFC 4627 "The application/json Media Type for JavaScript Object Notation (JSON)" (July 2006).

A JSON text is composed of Unicode characters, it accepts the following coding: UTF-8, UTF-16 (big endian and little endian), and UTF-32 (big endian and little endian). The default encoding is UTF-8.

JSON contains four primitive types (strings, numbers, booleans, and null) and two structured types (objects and arrays).

Rule 1: An EPAS message is represented by a JSON object. This object contains one of the following member:

- The *SaleToPOIRequest* JSON object for a request message,
- The *SaleToPOIResponse* JSON object for a response message,

Rule 2: Every data structure included in the Data Dictionary (section 5.2.2 *Data Elements and Structures*) is defined by a JSON object containing one member per component of the data structure where:

- The name of the JSON object member is the name of the EPAS data structure component,
- The value of the JSON object member is an acceptable value for the component type defined in the Data Dictionary,
- The presence of the JSON object member is specified by the multiplicity of the component.

Rule 3: Every repetition data structure included in the Data Dictionary is defined by a JSON array where each occurrence of the repetition is a value of the JSON array.

Rule 4: The primitives EPAS types are defined in the table below. For the definition of EPAS types, see section 5.2.1.5 *Type*.

EPAS Type	JSON coding
<i>Boolean</i>	The <i>boolean</i> JSON type with one of the two values true or false.
<i>TextString</i>	The <i>string</i> JSON type with a sequence of Unicode characters between two quotation characters.
<i>DigitString</i>	The <i>string</i> JSON type with a sequence of decimal digit characters (value '0' through '9').
<i>ByteSequence</i>	The <i>string</i> JSON type after a base64 conversion of the binary bytes.
<i>Enumeration</i>	The <i>string</i> JSON type defined by the set of values defined by the labels of the enumeration.
<i>Integer</i>	The <i>number</i> JSON type for the signed integer.
<i>Decimal</i>	The <i>number</i> JSON type for the signed decimal values.
<i>Cluster</i>	The <i>array</i> JSON type with a string value per label defined in the Cluster value.

Table 10: EPAS to JSON Data Type Conversion

The example of Login request message:

SaleToPOIRequest	
MessageHeader	
ProtocolVersion	3.0
MessageClass	Service
MessageCategory	Login
MessageType	Request
ServiceID	498
SaleID	SaleTermA
POIID	POITerm1
LoginRequest	
DateTime	2009-01-29T09:13:51.0+01:00
SaleSoftware	
ManufacturerID	PointOfSaleCo
ApplicationName	SaleSys
SoftwareVersion	01.98.01
CertificationCode	ECTS2PS001
SaleTerminalData	
	Attended
TerminalEnvironment	
SaleCapabilities	PrinterReceipt CashierStatus CashierError CashierDisplay CashierInput
SaleProfile	
GenericProfile	Extended
ServiceProfiles	Loyalty PIN CardReader
TrainingModeFlag	True
OperatorLanguage	sp
OperatorID	Cashier16
ShiftNumber	2
POISerialNumber	78910AA46010005

Provides the following JSON text (the white spaces and line feed added for the presentation can be removed to have a canonical form of the message):

```
{  
    "SaleToPOIRequest": {  
        "MessageHeader": {  
            "ProtocolVersion": "3.0",  
            "MessageClass": "Service",  
            "MessageCategory": "Login",  
            "MessageType": "Request",  
            "ServiceID": "498",  
            "SaleID": "SaleTermA",  
            "POIID": "POITerm1"  
        },  
        "LoginRequest": {  
            "DateTime": "2009-01-29T09:13:51.0+01:00",  
            "SaleSoftware": {  
                "ManufacturerID": "PointOfSaleCo",  
                "ApplicationName": "SaleSys",  
                "SoftwareVersion": "01.98.01",  
                "CertificationCode": "ECTS2PS001"  
            },  
            "SaleTerminalData": {  
                "TerminalEnvironment": "Attended",  
                "SaleCapabilities": ["PrinterReceipt", "CashierStatus", "CashierError",  
                    "CashierDisplay", "CashierInput"],  
                "SaleTerminalData": {  
                    "GenericProfile": "Extended",  
                    "ServiceProfiles": ["Loyalty", "PIN", "CardReader"]  
                }  
            },  
            "TrainingModeFlag": true,  
            "OperatorLanguage": "sp",  
            "OperatorID": "Cashier16",  
            "ShiftNumber": "2",  
            "POISerialNumber": "78910AA46010005"  
        }  
    }  
}
```

3.4.2 Dialogue Management

3.4.2.1 Type of Messages Exchange

As presented in the section 3.4.1.3 *Messages*, related to the category of message, there are several types of messages exchanges.

The *first type* of message exchange is of the type *question/answer*. The requestor sends a message request, the responder process the service requested, and sends a message response containing the result of the processing.

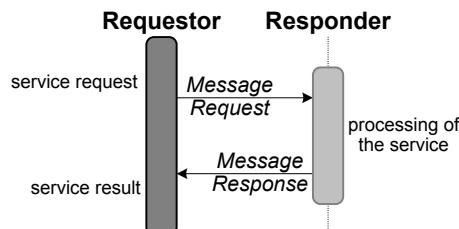


Figure 82: Request/Response Message Exchange

The messages pairs which always use this type of exchange are listed below by alphabetical order:

<i>Admin</i>	<i>Balance Inquiry</i>	<i>Batch</i>	<i>Card Acquisition</i>
<i>Card Reader APDU</i>	<i>Card Reader Init</i>	<i>Card Reader Power Off</i>	<i>Diagnosis</i>
<i>Enable Service</i>	<i>Get Totals</i>	<i>Input</i>	<i>Login</i>
<i>Logout</i>	<i>Loyalty</i>	<i>Payment</i>	<i>PIN</i>
<i>Reconciliation</i>	<i>Reversal</i>	<i>Stored Value</i>	<i>Transaction Status</i>
<i>Transmit</i>			

Other messages pairs sometimes use this type of exchanges if the response is required to know the result; they are listed below by alphabetical order:

<i>Display</i>	<i>Print</i>	<i>Sound</i>
----------------	--------------	--------------

The *second type* of message exchange is of the sending of a *message request only*. The Requestor sends a message request, the responder process the service requested, and do not send a message response at the end of the processing.

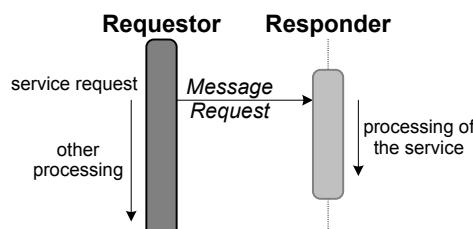


Figure 83: Request Only Message Exchange

The messages which use this type of exchange, for the following reasons:

- The response is not required, whatever the result of the service: *Display*, *Print*, and *Sound*,
- The response is not required, because the message affects directly the processing of the previous message request: *Abort*, and *InputUpdate*.

are listed below by alphabetical order:

<i>Abort</i>	<i>Display</i>	<i>InputUpdate</i>	<i>Print</i>
<i>Sound</i>			

The *third type* of message exchange is of the sending of an *unsolicited message notification* by the POI independently the processing of a request sent by the Sale system.

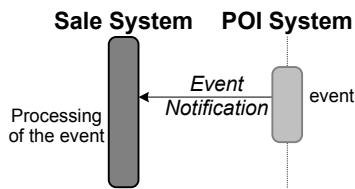


Figure 84: Unsolicited Notification Message Exchange

Only the *Event Notification* message uses this type of exchange:
Event

3.4.2.2 Type of Dialogues

A dialogue is a sequence of messages which are exchanged in order to provide a service.

There are four types of dialogues, all interactive, between the Sale System and the POI System:

1. The *Service dialogue* for the processing of a transaction by the POI System at the request of the Sale System.
2. The *Error Resolution dialogue* initiated by the Sale System attempting to resolve an error situation.
3. The *Device dialogue* for the processing of a device or low level operation by the POI System at the request of the Sale System.
4. The *Notification dialogue* when an unsolicited event occurs at the POI System which has to inform the Sale System of this occurrence.

3.4.2.3 Service Dialogue

A Service dialogue is the request from the Sale System to the POI System to process a transaction. The message categories involving a Service dialogue are listed below by alphabetical order:

Admin	Balance Inquiry	Batch	Card Acquisition
Diagnosis	Enable Service	Get Totals	Login
Logout	Loyalty	Payment	Reconciliation
Reversal	Stored Value	Transaction Status	

The *MessageClass* data element of the *MessageHeader* has then the value "Service".

Rule 1: Messages initiating a Service dialogue (*AdminRequest*, *BalanceInquiryRequest*, *BatchRequest*, *CardAcquisitionRequest*, *DiagnosisRequest*, *EnableServiceRequest*, *GetTotalsRequest*, *LoginRequest*, *LogoutRequest*, *LoyaltyRequest*, *PaymentRequest*, *ReconciliationRequest*, *ReversalRequest*, *StoredValueRequest*, and *TransactionStatusRequest*) must have in the *MessageHeader* the *MessageClass* to the value "Service".

If the Rule 1 is not respected, the receiver has to send back a message response with *Result*="Failure" and *ErrorCondition*="MessageFormat" (see section 4.6.2.1.6 *Unexpected Value*).

Between the request and the response, during the processing of the transaction, the POI may exchange the following Device requests:

Display	Input	Input Update	Print
Sound	Transmit		

The *MessageClass* data element of the *MessageHeader* has then the value "Device", and the *ServiceID* must be present and contains the identification of the Service request.

Rule 2: Between the request and the response of a Service dialogue, the Device messages *DisplayRequest*, *InputRequest*, *InputUpdate*, *PrintRequest* *SoundRequest*, and *TransmitRequest* must have in the *MessageHeader* the *MessageClass* to the value "Device".

If the Rule 2 is not respected and a response may be returned, the receiver has to send back a message response with *Result*="Failure" and *ErrorCondition*="MessageFormat" (see section 4.6.2.1.6 *Unexpected Value*).

Sequence flow of a Service Dialogue contains the following steps.

1. The Sale Server or a Sale Terminal sends a message request to the POI Server or a POI Terminal.
2. The POI Server or POI Terminal processes the service requested by the Sale System. This processing could require the POI to request some Device services to the Sale, which are then attached to the original transaction.
3. At the end of the processing, the POI System sends the message response containing the result of the transaction. This ends the Service Dialogue.

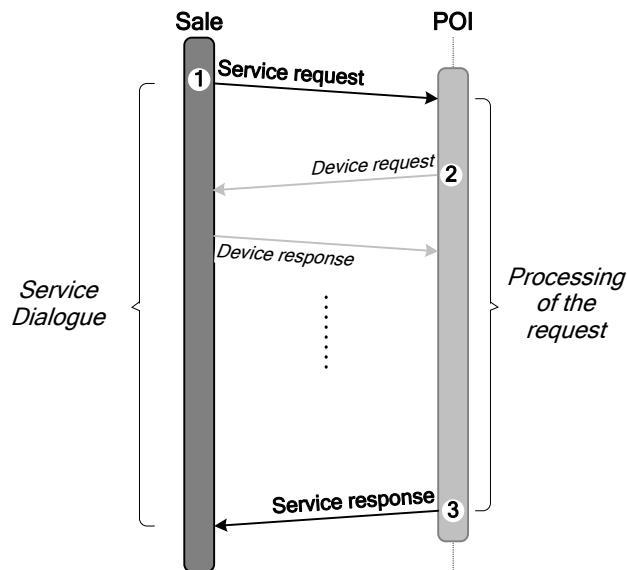


Figure 85: Service Dialogue Sequence Flow

Rule 3: During a Service dialogue, no other dialogue may be initiated by the Sale Terminal to this POI Terminal. There is only one Service request in progress on a logical connection between a Pair of Terminals or between a pair of Server (at the exception of *Transaction Status*, *Payment/Completion*, and *Abort*).

If a message request initiates a Service dialogue not conform to the Rule 3, the receiver has to send back a response message with *Result*="Failure" and *ErrorCondition*="NotAllowed" (see section 4.6.4.1.1 *Forbidden Dialogue*).

Rule 4: During the processing of a Service dialogue, the POI Terminal can send several Device request messages in parallel, and is not required to wait the Device response message of the preceding request before sending the next one.

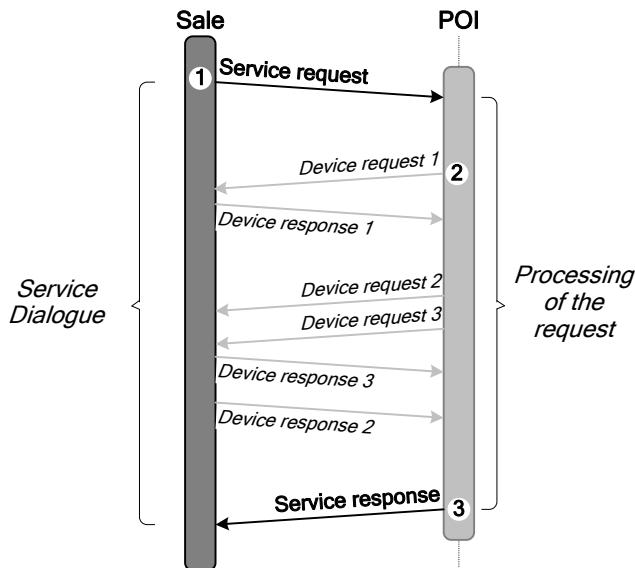


Figure 86: Service with Device Requests in Parallel

If the Device request cannot be served because the request requires a device already used by another request in progress, the receiver has to send back a message response with *Result*="Failure" and *ErrorCondition*="Busy" (see section 4.6.5.1.3 *Device Busy*).

As noticed before, some Device request messages do not require response message, only the message request is exchanged.

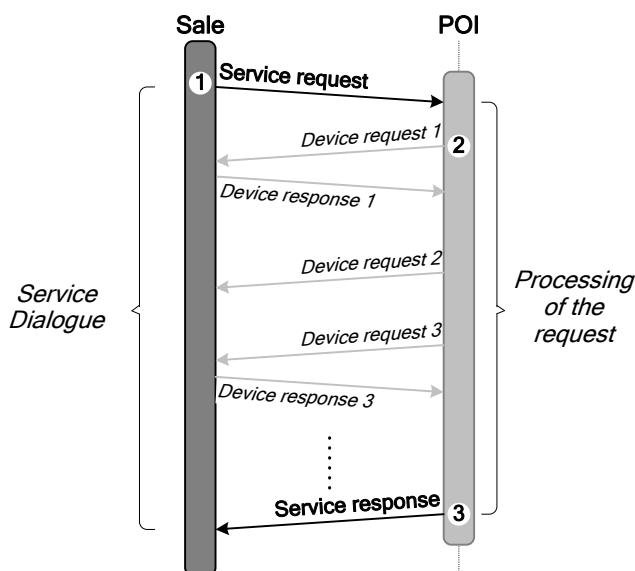


Figure 87: Service Dialogue with Device Request without Response

Rule 5: If there is no Service dialogue in progress and the POI Terminal sends a Device request message to a Sale Terminal, the Sale Terminal has to send back a response message with *Result*="Failure" and *ErrorCondition*="NotAllowed" (see section 4.6.4.1.3 *Forbidden Message*).

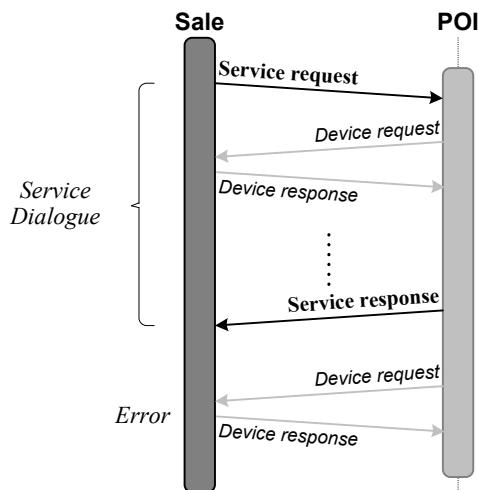


Figure 88: POI Sends a Device Request outside a Service Dialogue

3.4.2.4 Error Resolution Dialogue

In case of suspected error during a Service dialogue, the Sale System may launch an Error Resolution through an Error dialogue based on the message categorie:

Abort

The MessageClass data element of the *MessageHeader* has the value "Service".

Sequence flow of an Error Resolution contains the following steps.

1. The Sale Server or a Sale Terminal sends a message request to the POI Server or a POI Terminal.
2. While the POI System processes the service, the Sale System suspects some error at the POI System (e.g. evaluating the processing requires too much time). The Sale System then enters in an Error Resolution dialogue, and sends an *Abort* request message to the POI. This ends the Service Dialogue.
3. The POI System aborts the processing of the service, and sends the message response to the service. This ends the Error Resolution Dialogue.

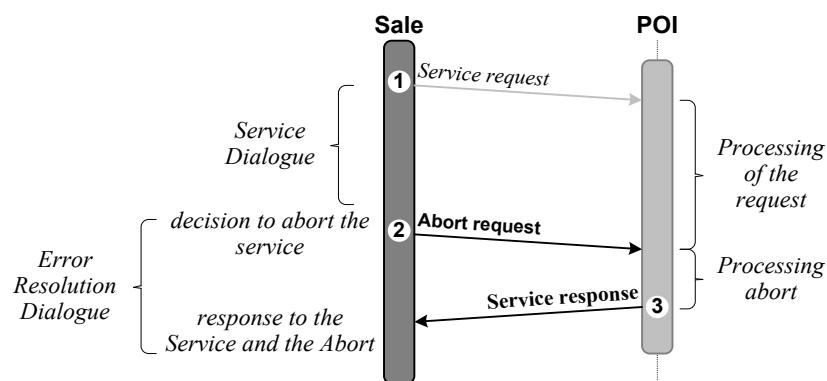


Figure 89: Error Resolution Dialogue Sequence Flow

See the section 4.7.2.3 *Abort Processing* and the section 4.7.5 *Error Resolution and Error Situations* for further details.

3.4.2.5 Device Dialogue

A Device dialogue is a Device exchange from the Sale System to the POI System to process a device or a low level operation. The message categories involving a Device dialogue are listed below by alphabetical order:

<i>Card Reader APDU</i>	<i>Card Reader Init</i>	<i>Card Reader Power Off</i>	<i>Display</i>
<i>Input</i>	<i>PIN</i>	<i>Print</i>	<i>Sound</i>
<i>Transmit</i>			

The *MessageClass* data element of the *MessageHeader* has then the value "Device", and the *ServiceID* must be absent, because the Device exchange does not depend and is not included in a Service dialogue.

Rule 1: Messages initiating a Device dialogue (*CardReaderAPDUREquest*, *CardReaderInitRequest*, *CardReaderPowerOffRequest*, *DisplayRequest*, *InputRequest*, *InputUpdate*, *PINRequest*, *PrintRequest*, *SoundRequest*, and *TransmitRequest*) must have in the *MessageHeader* the *MessageClass* to the value "Device".

If the Rule 1 is not respected and a response may be sent, the receiver has to send back a message response with *Result*="Failure" and *ErrorCondition*="MessageFormat" (see section 4.6.2.1.6 *Unexpected Value*).

Sequence flow of a Device Dialogue contains the steps presented below.

1. The Sale System or a Sale Terminal sends a device message request to the POI System or a POI Terminal.
2. The POI System process the operation requested by the Sale System, and sends the message response containing the result of the device service. This ends the Device Dialogue.

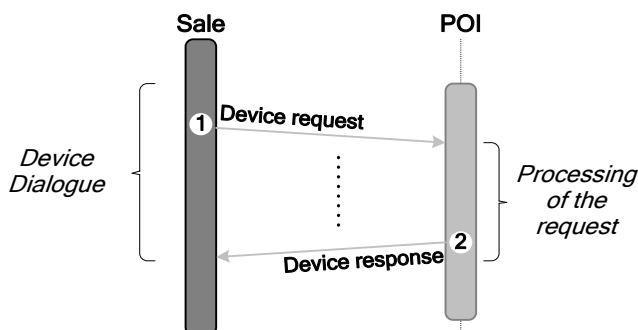


Figure 90: Device Dialogue Sequence Flow

Rule 2: During a Device dialogue, no Service dialogue may be initiated by the Sale Terminal to this POI Terminal.

If a message request initiates a Device dialogue not conform to the Rule 2, the receiver has to send back a message response with *Result*="Failure" and *ErrorCondition*="NotAllowed" (see section 4.6.4.1.1 *Forbidden Dialogue*).

Rule 3: Depending on the context, the Sale System may initiate two Device Dialogues in parallel (e.g. to display a message as you are printing a receipt).

Rule 4: During the processing of the Device request below, the POI System may initiate Device requests *Display* or *Input*.

[*Card Reader APDU*](#)[*Card Reader Init*](#)[*Card Reader Power Off*](#)

- Rule 5: During the processing of the Device request below, the POI System cannot initiate Device requests *Display* or *Input*.

PIN

If the Device request cannot be served because the request requires a device already used by another request in progress or the command is forbidden, the receiver has to send back a message response with *Result*="Failure" and *ErrorCondition*="Busy" (see section 4.6.5.1.3 *Device Busy*).

3.4.2.6 Notification Dialogue

Sequence flow of a Notification Dialogue is limited to the sending of the message *Event Notification* from the POI System or a POI Terminal to the Sale System or a Sale Terminal.

The MessageClass of the *MessageHeader* has the value "Event".

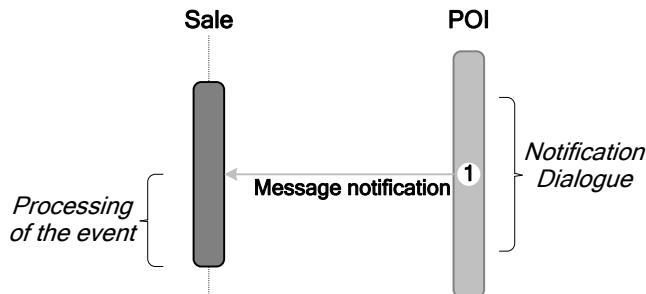


Figure 91: Notification Dialogue Sequence Flow

Rule 1: A Notification dialogue may be initiated at any time by the POI Terminal.

Rule 2: If there is no open transport connection between the Sale and the POI component, the POI cannot send the notification. The POI has no obligation to store the event in order to send the notification when a new transport connection is available).

3.4.2.7 State Diagrams

Sale System State Diagram

This is the state diagram of the Sale System to manage the message application dialogues. The states and events of this diagram are detailed in the tables below.

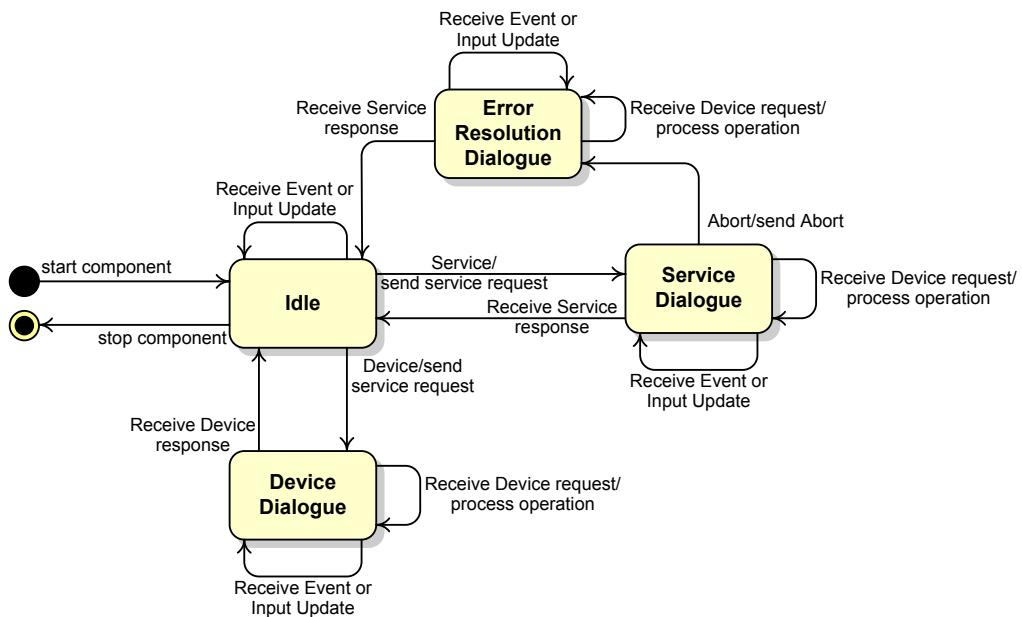


Figure 92: Sale System Dialogues Management State Diagram

State	Definition
Idle	The Sale System is ready.
Service Dialogue	A Service dialogue is ongoing. The Sale System has sent a Service request to the POI component, and is waiting the response message.
Error Resolution Dialogue	An Error Resolution dialogue is ongoing. During a Service dialogue, the Sale System has sent an Abort request to the POI component, and is waiting the response message.
Device Dialogue	A Device dialogue is ongoing. The Sale System has sent a Device request to the POI component, and is waiting the response message.

Table 11: Dialogues Management Sale System States

Event	Definition
Service	The Sale application requests a Service.
Abort	The Sale application requests an Abort.
Receive Service response	The Sale component receives the Service response.
Device	The Sale application requests a Device operation.
Receive Device request	The Sale component receives a Device request from the POI.
Receive Device response	The Sale component receives the Device response.
Receive Event notification	The POI component sends an Event Notification message..

Table 12: Dialogues Management Sale System Events

POI System State Diagram

This is the state diagram of the POI System to manage the message application dialogues. The states and events of this diagram are detailed in the tables below.

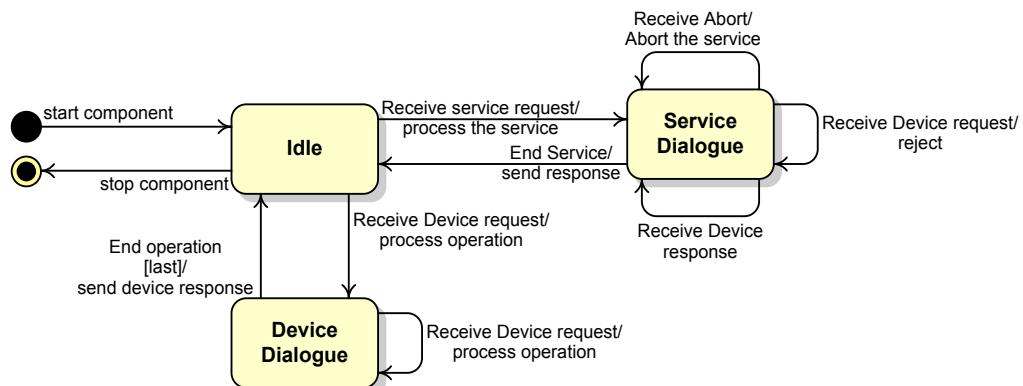


Figure 93: POI System Dialogues Management State Diagram

<i>State</i>	<i>Definition</i>
Idle	The POI System is ready.
Service Dialogue	A Service dialogue is in progress. The POI System has received a Service request from the Sale component, and is processing the service.
Device Dialogue	A Device dialogue is in progress. The POI System has received a Device request from the Sale component, and is processing the operation.

Table 13: Dialogues Management POI System States

<i>Event</i>	<i>Definition</i>
Receive service request	The POI component receives a Service request from the Sale component.
Receive Abort	The POI component receives an Abort request from the Sale component.
End Service	The POI component has completed the Service.
End operation	The POI component has completed the Device operation.
Receive Device request	The POI component receives a Device request from the Sale.
Receive Device response	The POI component receives the Device response from the Sale.

Table 14: Dialogues Management POI System Events

3.4.3 Protocol Version Management

In some cases the Sale System and the POI System exchange messages using a different version of the Sale to POI protocol. Difference between versions involves either:

- A behaviour modification to respect, either
- A modification of messages providing a new function or a reorganisation of data.

Any evolution of the protocol has to specify the way to use the new version with the previous one. Of course, strong gap between versions could break compatibility between versions. This is the case when the complete structure of the message is changed, but even in this case some compatibility between versions could be achieved¹⁷.

The highest version of the protocol managed by the sender of a message is conveyed in the *ProtocolVersion* data element of the *MessageHeader*, with the following sequence:

1. The sender of the message request, a Sale System component, put in the *MessageHeader.ProtocolVersion* the highest version *N* of the Sale to POI protocol managed by the component.
2. The receiver of the request, a POI System component, manages *M* as the highest version of the Sale to POI protocol.
 - a. If the version *N* is considered as obsolete, the POI component answers with an appropriate error response and the value *M* in the *MessageHeader.ProtocolVersion*.
 - b. After processing of the message request, the POI component put the value *M* in the *MessageHeader.ProtocolVersion*. If the version *M* is newer than *N*, the POI answers with the version *M*, and the Sale has to accept.
3. After receiving the message, the Sale System component, may stop the dialogue if it consider that the protocol version *M* managed by the POI System component is too old.

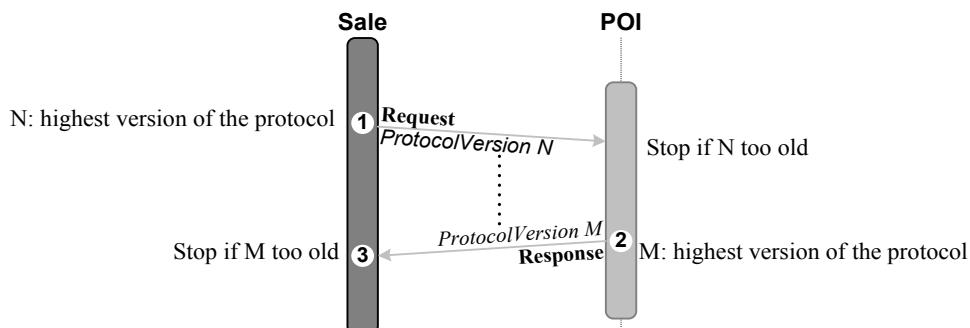


Figure 94: Exchange of Protocol Version

The Sale System and the POI System have configuration parameters to store the versions they can accept for the other side. In any case the versions they can manage together have to be tested during integration process before to put it in the field.

¹⁷ For instance to ensure the compatibility between change of structure of SSL messages in version 2 and version 3, particularly the location of the SSL protocol version, any implementation of the SSL version 3 always sends the first message of an SSL session in the version 2 format.

Name	POI Version
<i>Definition</i>	Minimum and maximum protocol version that the POI can manage.
<i>Usage</i>	Protocol version management.
<i>Specification</i>	3.4.3 Protocol Version Management

Name	Sale Version
<i>Definition</i>	Minimum and maximum protocol version that the Sale can manage.
<i>Usage</i>	Protocol version management.
<i>Specification</i>	3.4.3 Protocol Version Management

Configuration 7: Sale and POI Protocol Version

Both systems need to know the protocol version used by the other system, and has to be exchanged periodically.

Rule 1: The *ProtocolVersion* is mandatory in the Login Request and Login Response messages.
 (MessageFormat, mandatory data absent, Sale Terminal cannot login).

If *ProtocolVersion* is absent from the Login Request, the receiver has to send back a message response with *Result*="Failure" and *ErrorCondition*="MessageFormat" (see section 4.6.2.1.2 *Mandatory Data Absent*).

Rule 2: The *ProtocolVersion* must be absent in the messages exchanged between the Login Response and the following Logout Request. If *ProtocolVersion* is present in these messages, it must be ignored.

Rule 3: The *ProtocolVersion* is mandatory for the messages exchanged between a Logout Response and the following Login Request, or between a Login Request and Login Response messages, if and only if the version of protocol has changed since the last *ProtocolVersion* exchange¹⁸, and the messages exchanged have changed their format.

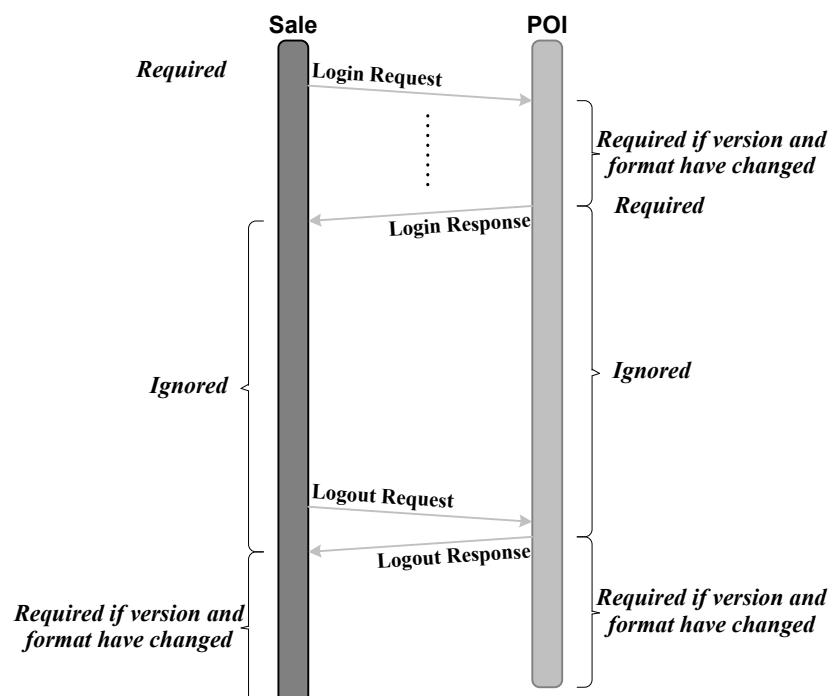


Figure 95: Required Exchange of Protocol Version

¹⁸ The POI can send Display or Input Request messages between the Login Request and the Login Response.

3.4.4 Origin and Destination of Messages

As presented in the section 2.5 *System Components Identification and Relationship*, the Sale to POI protocol exchanges messages between a component of a Sale System and a component of a POI System:

- Either between the Sale Server and the POI Server, which is considered as an exchange between the Sale System and the POI System,
- Either between a Sale Terminal and a POI Terminal.

The following message categories might be exchanged between Servers or Systems:

Abort	Admin	Balance Inquiry	Batch	Card Acquisition
Diagnosis	Display	Event	Get Totals	Login
Logout	Loyalty	Payment	Reconciliation	Reversal
Stored Value	Transaction Status			

The following message categories might be exchanged between Terminals:

Abort	Admin	Balance Inquiry	Batch	Card Acquisition
Card Reader APDU	Card Reader Init	Card Reader Power Off	Diagnosis	Display
Enable Service	Event	Get Totals	Input	Input Update
Login	Logout	Loyalty	Payment	PIN
Print	Reconciliation	Reversal	Sound	Stored Value
Transaction Status	Transmit			

Rule 1: If the *SaleID* or *POIID* is not recognised in a request message, the receiver of the message must send a related response message with the *Response* component value “Failure”-“NotAllowed” (see section 4.6.4.1.4 *NotAllowed Value*).

In all the case the message request is not processed, in particular if the message request was a Login, the session is not open.

Rule 2: If in a response message, the *SaleID* or *POIID* is not recognised or cannot match an unanswered request, the message is ignored.

3.4.5 Identification of Message Exchanges

Messages are identified by the data elements *ServiceID* and *DeviceID* of the *MessageHeader*. These identifications allow:

- The association of a message response to a message request. So a message response has the same identification value than the request.
- The detection of duplicate messages.
- The linking of a sequence of messages which have to be associated in the application for a particular service.

Rule 1: Identification of a message pair, between a Sale System/Terminal and a POI System/Terminal has to be unique during a period of time, sufficient to reduce the probability of duplicate messages.

If the Rule 1 is not respected, the receiver has to send back a message response with *Result*="Failure" and *ErrorCondition*="MessageFormat" (see section 4.6.2.1.9 *Repeated Message*).

A simple management of the evolution of the identifier is sufficient, e.g. incrementation of a value for each message, with a sufficient length in order to come back on the same value during an acceptable period of time. However, it is strongly recommended to store the last used value, to avoid reinitialisation of the value at the starting of the protocol stack.

The numeric part of the identifier could be prefixed by any kind of identification allowing the sender to link a set of messages for a certain purpose. In this case, the prefix cannot be recognised by the receiver, and the identification is considered as a whole by the involved parties.

Rule 2: The data element *ServiceID* is mandatory for the message of the "Service" *MessageClass*. The sender of the request (i.e. the Sale System) put the value in the request, which has to be repeated in the response.

Abort	Admin	Balance Inquiry	Batch
Card Acquisition	Diagnosis	Enable Service	Get Totals
Login	Logout	Loyalty	Payment
Reconciliation	Reversal	Stored Value	Transaction Status

If the Rule 2 is not respected, the receiver has to send back a message response with *Result*="Failure" and *ErrorCondition*="MessageFormat" (see section 4.6.2.1.2 *Mandatory Data Absent*).

Rule 3: For Device requests Display, Input and Print, sent by the POI during the processing of a Service:

- The data element *ServiceID* is mandatory with the same value than the "Service" message request.
- The data element *DeviceID* is mandatory with a value which identifies the message inside the Service dialogue.

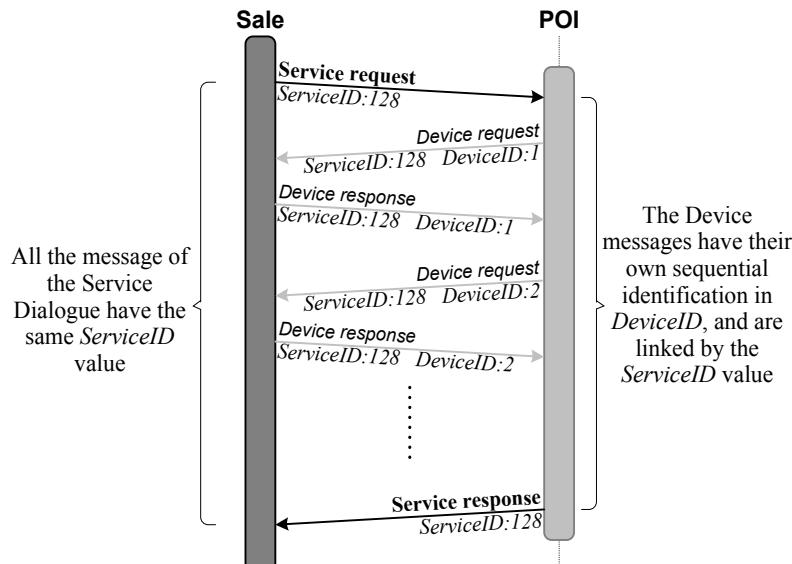


Figure 96: Service Dialogue Messages Identification

Rule 4: The data element *DeviceID* is mandatory for the message sent by the Sale System outside a Service message pair. The sender of the request (i.e. the Sale System) put the value in the request, which has to be repeated in the response. The data element *ServiceID* has to be absent for these messages.

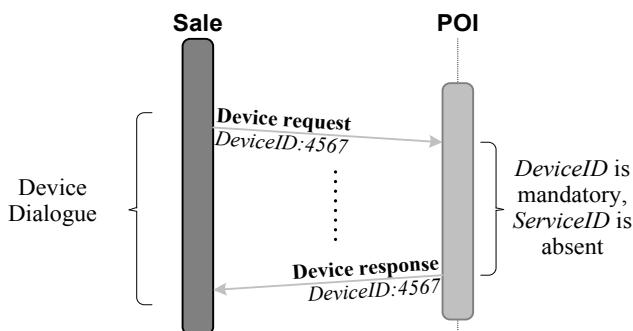


Figure 97: Device Dialogue Messages Identification

If the rule 3 or 4 is not respected for the message request, the receiver has to send a message response with "Failure - MessageFormat - Mandatory Data Element or Structure Absent" or "Failure - MessageFormat - Unexpected Data Element or Structure" in the *Response* data structure.

Rule 5: Whatever the *MessageClass* of the message request, the data elements *ServiceID* and *DeviceID* cannot have the same value than a previous message during a period of time. The receiver of such message request has to refuse the request and send a message response with "Failure - MessageFormat - Repeated Message" in the *Response* data structure (see section 4.6.2.1.9 *Repeated Message*).

Rule 6: Whatever the *MessageClass* of the message, if the identification value of a message response (data elements *ServiceID* and *DeviceID*) do not correspond to a message request in progress, the receiver of such message response has to ignore the message response.

3.4.6 Message Security

This section presents protection of information for the messages of the Retailer protocol. POI System performs transactions for card requiring various levels of security. Mechanisms presented in the specifications have been introduced to comply with the security policies of these card schemes.

Messages of the Retailer protocol may contain data:

- Protected by the application (e.g. customer personal data), and
- Protected by the protocol before sending the message (e.g. card data), conforming to the scheme rules security policy (e.g. PCI-DSS).

In addition to the protection of specific data of the messages, the whole message, header and body, may be protected by a Message Authentication Code (MAC) in an optional message trailer.

All the protected data and the related information are formatted according to the Cryptographic Message Syntax (CMS) standard.

3.4.6.1 CMS Principles

The Cryptographic Message Syntax (CMS) defines a generic data structure (`ContentInformationType`). This data structure is an encapsulation of an encrypted content, an authentication code (MAC), a digital signature, or a digest of any arbitrary part of a message.

The CMS is general enough to convey various attributes related to the protected data (e.g. identifications of the used keys, encrypted keys, cryptographic algorithms with their parameters, certificate and revocation lists, time stamps), and can support various architectures of key management. In addition the syntax of the data structure accepts multiple encapsulations, and these encapsulations can be nested.

As illustrated in the figure below, the CMS generic data structure is used by the Retailer protocol:

- 1 To reformat the data protected by the application with the related information (e.g. encrypted PIN).
- 2 To protect the sensible data transferred in the message with the required security (e.g. card data).
- 3 To protect the complete message (header and body) by a MAC.

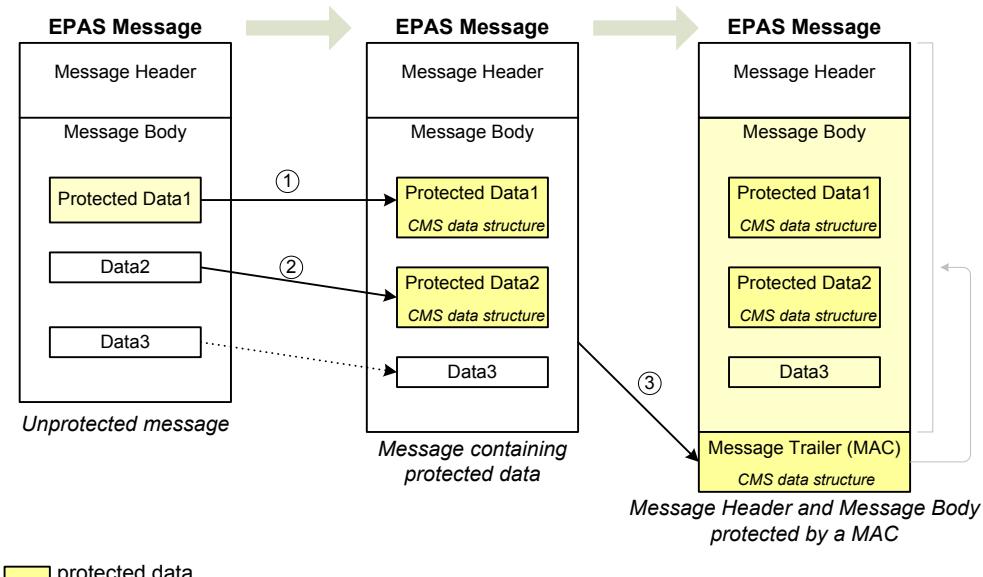


Figure 98 : CMS Data Protection in Retailer Protocol Messages

3.4.6.2 Cryptographic Mechanisms

Retailer protocol uses standard cryptographic algorithms and hash functions mechanisms.

Triple-DES with 112 bits key length, according to FIPS PUB 46-3, is the only encryption algorithm used.

The block cipher mechanisms CBC-Mode, as defined in ANSI X9.23 is used for data encryption (e.g. card data). Key encryption uses Triple-DES CBC-Mode encryption as defined in ANSI X9.52 - 1998: Triple Data Encryption Algorithm, Modes of Operation.

The authentication of messages uses the Retail CBC-MAC, as defined in ISO 9797-2 v2002, applied to the digest of the message generated with the SHA-256 hash function (Retail CBC MAC with SHA-256).

For the Sale to POI communication, a simple key management mechanism is available, with a session key, generated for every exchange of messages, and transported by a symmetric Key Encryption Key (KEK). The session key encryption algorithm, which uses Triple DES algorithm in CBC mode, is identified by des-ede3-cbc, and the encrypted session key is located in the EncryptedKey data element.

Installation of the KEK is out of scope of the protocol.

3.4.6.3 Key Sets

Each initiating party will have a key set (comprising MAC and data encryption keys) for each recipient party that the initiator of the exchange communicates with.

Therefore a key set is hold per recipient party identified by the key name and the key version:

MAC Key

The MAC key is used for the generation and verification of Message Authentication Codes authenticating the message. The MAC key is identified by a name containing the suffix "MACKey".

Data Encryption Key

The data encryption key is used for the encryption and decryption of application data elements, with the exception of the PIN. The Data encryption key is identified by a name containing the suffix "DATKey".

Test and Production Keys are differentiated by the key name (e.g. the identifiers "TestMACKey" and "TestDATkey" are used for test keys).

The key version identifies the creation date of the key. The parties usually hold only one version of the key set.

3.4.6.4 Key Encryption

For the *EnvelopedData* (protection by encryption) and *AuthenticatedData* (protection by MAC), the *Recipient* data structure contains information about the encryption key related to recipients.

The Retailer protocol retains the mechanism with a session key transported by a Key Encryption Key (KEK) using a Triple-DES encryption in ECB mode.

This mechanism uses the *KEK* alternative of the *Recipient* data structure, with:

- *Version* containing the value "v4",
- *KEKIdentifier* containing the name of the KEK as described in the section 3.4.6.3, *Key Sets* in the *KeyIdentifier* data element, and the version of the key in the *KeyVersion* data element.
- *KeyEncryptionAlgorithm* contains the *Algorithm* data element with the value "des-ede3-cbc" without *Parameter*.
- *EncryptedKey* contains the session key encrypted by the KEK using the figure below.

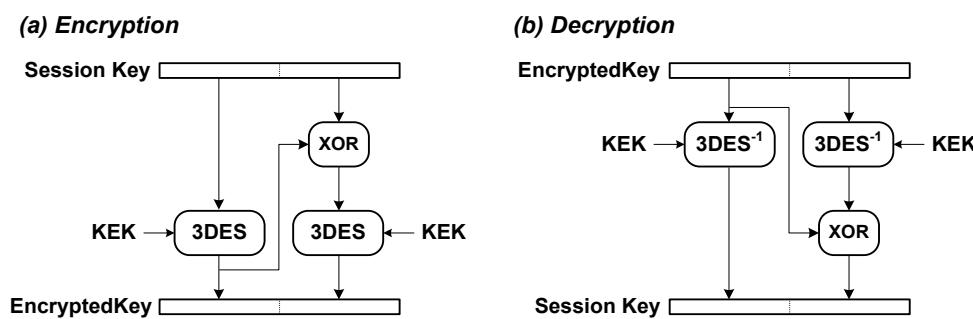


Figure 99 : Encryption/Decryption of a Session Key

An example of key encryption is provided in *Annex B*.

3.4.6.5 Data Encryption

Data encryption uses the *EnvelopedData* alternative of the *ContentInformationType* data structure with *ContentType* containing the value "id-envelopedData".

EnvelopedData data structure contains:

- *Version* containing the value "v0",
- *Recipient* containing the encryption key, encrypted by the KEK, as described in the section 3.4.6.4, *Key Encryption*.
- *EncryptedContent* contains :
 - *ContentType* with the value "id-data" without *Parameter*.
 - *ContentEncryptionAlgorithm* which contains *Algorithm* with the value "des-ed3-cbc" with the *Parameter.InitialisationVector* containing the 8 bytes length initial value of the CBC mode.
 - *EncryptedData*, containing the result of the encryption.

Encryption Process:

- (i) Padding of the data to encrypt M: the hexadecimal byte 80 is appended to M. If the new length is not a multiple of 8, M is extended by null bytes (hexadecimal 00), to reach a length multiple of 8.
- (ii) The result M of the padded data is split in blocks of 8 bytes $M_1 \dots M_n$
- (iii) With the session key K previously generated, and initialising C_0 by the value of *InitialisationVector*, the encrypted data is the concatenation of $C_1 \dots C_n$, where

$$C_i = E_K(C_{i-1} \text{ xor } M_i)$$

 E_K being the Triple-DES encryption with K

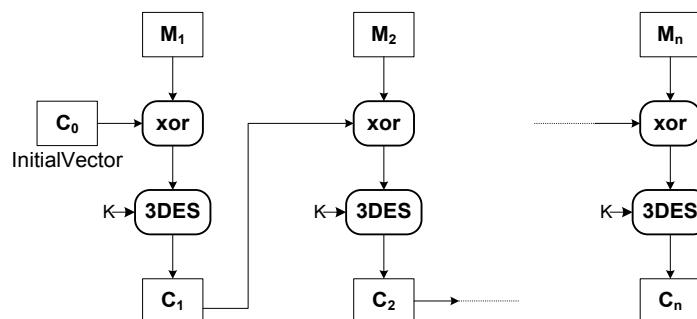


Figure 100 : Data Encryption Process

Decryption Process:

- (i) The encrypted data C is split in bloc of 8 bytes $C_1 \dots C_n$
- (ii) With the session key K previously decrypted, and initialising C_0 by the value of *InitialisationVector*, compute the following blocs $M_1 \dots M_n$, where
$$M_i = D_K(C_i) \text{ xor } C_{i-1}$$
$$D_K \text{ being the Triple-DES decryption with } K$$
- (iii) The last block M_n is right padded by the hexadecimal byte `80` followed by a sequence of 0 to 7 null bytes, hexadecimal `00` (if this not the case, decryption has failed, most probably a wrong key encryption key). Remove the byte(s) of padding of the bloc M_n . The decrypted data is the concatenation of the blocs $M_1 \dots M_n$

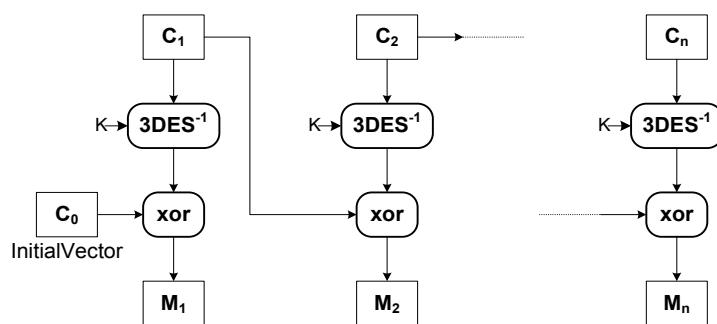


Figure 101 : Data Decryption Process

An example of data encryption is provided in *Annex B*.

3.4.6.6 MAC Protection

MAC protection uses the *AuthenticatedData* alternative of the *ContentInformationType* data structure with *ContentType* containing the value "id-ct-authData".

AuthenticatedData data structure contains:

- *Version* containing the value "v0",
- *Recipient* containing the encryption key, encrypted by the KEK, as described in the section 3.4.6.4, *Key Encryption*.
- *MACAlgorithm* which contains *Algorithm* with the value "id-retail-cbc-mac-sha-256" with the *Parameter.InitialisationVector* absent, a sequence of 8 null bytes being used for the CBC mode.
- *EncapsulatedContent* contains :
 - *ContentType* with the value "id-data" without *Parameter*.
- *MAC*, containing the result of the MAC computation described below, after generation and encryption of the session key, as described in the section 3.4.6.4, *Key Encryption*. For an exchange of messages, the generated key used for the MAC in the request must be reused in the message response.

MAC Computation Process:

- (i) Concatenate the header and the body of the message, and compute the SHA-256 digest D on the result of this concatenation.
- (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 00), to reach a length multiple of 8.
- (iii) The result D of the padded data is split in bloc of 8 bytes $D_1 \dots D_n$
- (iv) With the left part K_L of session key K previously generated, and initialising C_0 by 8 null bytes, compute the sequence $C_1 \dots C_{n-1}$, where

$$C_i = E_{KL} (C_{i-1} \text{ xor } D_i)$$

$$E_{KL}$$
 being the DES encryption with K_L
- (v) The MAC is the result of:

$$\text{MAC} = E_K (C_{n-1} \text{ xor } D_n)$$

$$E_K$$
 being the Triple-DES encryption with K

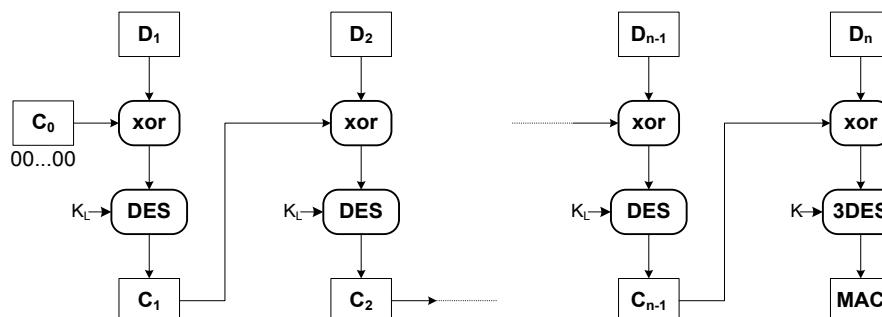


Figure 102 : MAC Computation Process

An example of MAC protection is provided in *Annex B*.

4 Application Protocol Specifications

This chapter starts with a presentation which provides a summary of the interface in terms of messages and functionalities, including implementation profiles.

The section 4.2, *Systems Synchronisation*, specifies the part of administrative services related to the synchronization of systems. It describes the management of session between systems, specifies the synchronization messages Login, Logout and Enable Service, and ends with a global state diagram of the synchronization.

The section 4.3, *Financial Services*, specifies all the financial services:

- The section starts with the section 4.3.1, *Standard Payment* which presents the payment messages, and specifies the standard payment service. It also specifies the identification of transactions.
- The section 4.3.2, *Other Payment Services* presents all the extensions of service for a payment transaction: cash-back, split payment, deferred sale, reservation, payments installment (by the merchant and by the issuer), recurring payments, tip, referral, refund, cash-advance, currency conversion, tokenisation and customer orders management. For each service, the additional data of the payment messages required by the service are presented, followed by the specification of the service.
- The section 4.3.4, *Loyalty Services* presents the loyalty part of the financial service. Loyalty service could be provided either inside a payment message for synchronization with the payment transaction, or with a dedicated loyalty message. The section presents an overview of the loyalty services, the loyalty message, the loyalty data of the payment messages, and the specification of the loyalty.
- The other sections present:
 - The card acquisition messages, allowing the reading and the analysis of the card used by the customer, before the process of the transaction.
 - The stored value messages to activate, load or reload stored value cards.
 - The reversal messages to process a manual reversal of a financial transaction.
 - The batch messages to process transactions without real time contact with the Sale system, and to get the result of these transactions.
 - Use cases providing additional payment services: pay-at-the-table.

The section 4.4, *Administrative Services*, specifies the part of administrative services related to the financial services. The section describes:

- The reconciliation messages to exchange the totals of the transactions during a certain period,
- The balance inquiry of an account attached to a payment, loyalty or stored value card;
- The get totals messages, to exchange the actual totals of the transactions since the beginning of a period.

The section 4.5, *Devices Services*, specifies the low level services. First the section describes the format of information to display or print, which is used by most of the device services. Then the section specifies the protocol for each of the nine devices or services: card reader, display, input, input update, print, PIN, sound and transmit . Most of these services define first the logical devices, present the services, describe the related messages and specify the processing of the various related services.

The section 4.6, *Error Reporting* and *Error Cases*, presents all the error cases of the protocol by category, and the standard framework to report the details of errors or warnings.

The section 4.7, *Error Management*, presents first the messages of the protocol that are used for error resolution: transaction status, abort, event notification and diagnosis. Then, the section 4.7.5, *Error Resolution* and Error Situations, presents first the general specification of error resolution on the two sides of the protocol. This section is completed by a set of typical error situations with the way to resolve these errors.

4.1 Overview

4.1.1 Presentation of the Interface

The set of messages of the Sale to POI protocol provides three categories of services:

1. The *Financial Services*, the POI System offers for the processing of the Sale System financial transactions.
2. The *Device Services*, which could be provided both by a Sale Terminal and a POI Terminal to share components or low level services between Sale and POI Systems.
3. The *Administrative Services*, to manage the interface between the POI System and the Sale System.

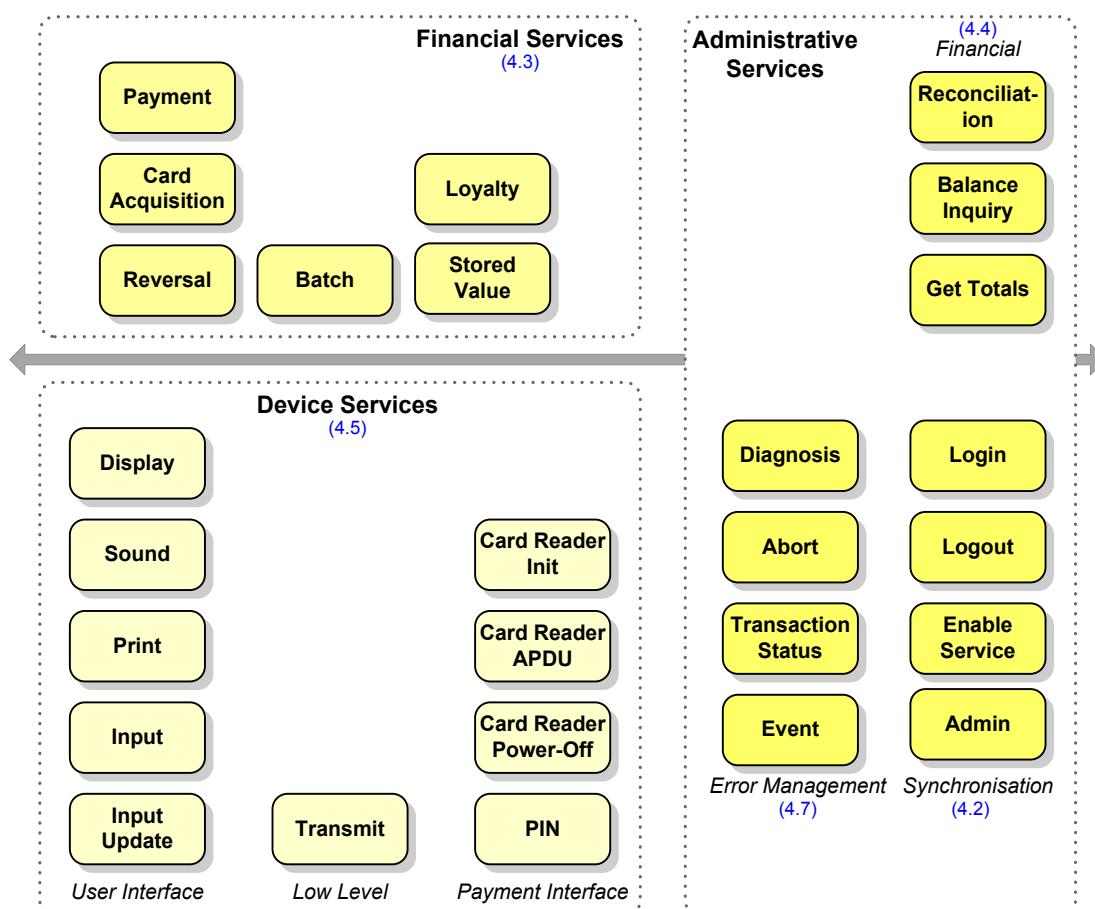


Figure 103: Presentation of the Interface

This organisation provides two levels of interface, which might be mixed and interleaved for a maximum of flexibility without removing any interoperability.

Financial Services are provided by the following messages:

- Payment** Realises the purchase payment of goods or services by the customer with a payment card. In addition to the standard payment, various kinds of payment services could be requested with this message: refund, deferred sale or reservation.
- One or several loyalty transactions, associated with the payment, could be

	realised at the same time. The kind of loyalty transaction is determined during the processing of the message (award, redemption...)
Loyalty	Realises a loyalty transaction separately, because there is no payment or the payment is made by another way (e.g. cash, another command...). The message processes loyalty award and redemption transactions, or their refund.
Card Acquisition	Reading and analysis of card to be used by a payment or loyalty transaction. It allows the Sale to make process dedicated to the card used, before the payment or loyalty transaction has started, or allows using the same card for several transactions.
Stored Value	Realises both administration and services of stored value cards: the activation, loading or reloading, and the payment. (The Payment message can use any type of card, including stored value cards)
Reversal	Cancels a previous Payment, Loyalty or Stored Value transaction.
Batch	When the POI terminal cannot be connected to the Sale System (e.g. payment at delivery), it allows to get the outcome of the transactions, and potentially to preload the transactions to perform.

Device Services offer *User Interface* services with the following messages:

Display	Outputs of information to the cashier or the customer, on logical devices of the POI Terminal or the Sale Terminal. Formats of information include referenced messages, text, XHTML or barcode.
Sound	To produce various forms of sounds to a customer or an operator interface.
Print	Printing of information, on logical devices of the POI Terminal or the Sale Terminal. The message offers some characteristics associated to the payment receipt printing.
Input	Request of information to the cashier or the customer, on logical devices of the POI or Sale Terminal. The message could contain information to display before the information entry. Input is requested by a command asking an answer (yes or no) e.g. for requesting the verification of the cardholder signature by the cashier as authentication method, some form of entry (text, numeric, function key, any key, selection of an item on a menu, customer assistance), and the confirmation of the site manager on the Terminal.
Input Update	When an event modifies the information presented to the user through an Input in progress, it allows updating the display without aborting the Input.

Device Services offer *Payment Interface* service with the following messages:

PIN	Performs for private cards PIN entering, offline verification of the cardholder PIN or both.
Card Reader Init	Enables the entering of a card in a card reader, and initialises a card when inserted in the reader.
Card Reader APDU	Exchanges of ISO 7816 APDU-Q and APDU-R with the chip of an initialised smart card (See 4.6.4.1.8: The POI system must not allow all possible APDUs e.g. the commands "Verify PIN" or "Generate Transaction Certificate" to protect a payment card against misuse).
Card Reader Power-Off	Processes a power-off to the chip of an initialised smart card, and processes the removal of the card.

Device Services offer *Low Level* service with the following message:

Transmit	To use the other party as a communication gateway by a terminal to exchange messages with a host.
-----------------	---

Administrative Services offer *Financial* administrative services with the following messages:

Reconciliation	Performs the balance of the computed transactions during a certain period. It could be an already closed period or the current period which is closed by the request. It could concern transactions between the Sale System and the POI System, the POI System and an Acquirer, or both. Totals are computed conforming to some criteria configured in the POI System.
Balance Inquiry	Realises the reading and the analysis of a payment card, a loyalty card, or a stored value card, and provide the balance or other information of the associated account.
Get Totals	Performs the totals of the transactions realised by the POI System since the beginning of the current reconciliation period. The Sale System could request all the transactions, or only those performed by a particular POI Terminal, Sale Terminal, cashier, shift, or a particular category of transactions identified by a group ID. Totals are computed per Acquirer and card brand, and might also be presented according to other criteria.

Administrative Services offer *Synchronisation* services with the following messages:

Login	Accomplishes the initialisation of the interface between two components of the Sale System and the POI System. The environment, capabilities and profile of the two components are exchanged, with the default value of some information that will be used for the next exchanges.
Logout	Ends the association between two components of the Sale System and the POI System, which was previously realised by a Login. After this exchange, only basic administrative exchange could be requested.
Enable Service	Enables starting of a financial transaction before the request of the Sale System, for the so-called swipe-ahead transaction. In addition, this message could request the termination of a transaction started by a Card Acquisition or a swipe-ahead.
Admin	Select and start customised administrative services provided by the POI, with possible usage a "menu" for an interactive or software interface.

Administrative Services offer *Error Management* services with the following messages:

Diagnosis	Requests the status of a POI Terminal and its components. The message could request, when available, the reachability of the hosts.
Abort	Request from the Sale System to abort a current financial transaction or a device command.
Transaction Status	Request from the Sale System to get the result of a previous transaction (e.g. Payment, Loyalty), because the response message was not received by the Sale System.
Event	This is an unsolicited notification of an event from the POI System, to report a change of state or an error external to the processing of a message request. The message conveys information related to the event, and possible action to take.

4.1.2 Profiles

The Sale to POI protocol could be used by various environments in many different domains. Implementations may contain the part of the protocol that satisfies their needs in term of product. Services that an implementation offers are provided as a functional profile, which defines the messages that the product has implemented and extension to these messages in term of service.

Profiles define globally the services which are implemented by the Sale system of the Payment system. The details of services which are implemented are consigned in the Implementation Conformance Statement (ICS) attached to the implementation. ICS are described in the Test Cases document of the EPAS Sale to POI protocol.

Profile is composed of:

- A *Generic Profile* which indicates what group of messages of the Sale to POI protocol are used or provided,
- A list (possibly empty) of *Service Profiles* which are implemented either in term of messages, either in term of an optional part of the messages.

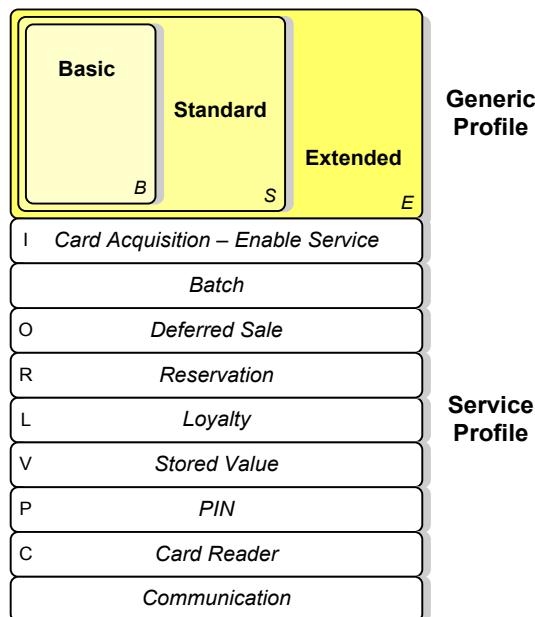


Figure 104: Functional Profile

There are three *Generic profiles* defined by the protocol:

1. The *Basic* profile, with contains the minimum of messages to make payments and to manage the POI System. It contains:
 - a. The Login/Logout messages to synchronise the Systems and start a session.
 - b. The basic Payment message without loyalty or additional payment features and the Reversal to cancel a payment.
 - c. The messages Abort, Transaction Status and Event, to manage the errors in an interoperable way.
 - d. The Reconciliation message to synchronise the transaction totals between Systems, according to various parameters if needed.
2. The *Standard* profile, which allows interactions between Sale and POI, and device sharing. In addition to the Basic profile messages, it contains:
 - a. The Print, Display and Input messages allowing the POI to interact with the Sale Terminal devices.

- b. Diagnosis messages as enhanced management functions of the POI System.
- 3. The *Extended* profile, which provides a more complete interface. In addition to the Standard profile messages, it contains:
 - a. The Get Totals for reporting.
 - b. The Print, Display and Input messages requested by the Sale Terminal to the POI Terminal devices.

The *Service Profiles* that could be defined in addition to the generic profile are:

- 1. *Card Acquisition* or *Enable Service* which could be requested only by the Sale System.
- 2. *Batch* which could be requested also by the Sale System only.
- 3. *Deferred Sale* service which could be requested by the Sale System, as for outdoor petrol. It allows the service in the Payment message, and could use Input messages. In the Payment message PaymentData.PaymentType could have the value “OneTimeReservation” only for this profile, which uses also the value “Completion”.
- 4. *Reservation* service which could be requested by the Sale System with the Payment message. PaymentData.PaymentType could have the value “FirstReservation” and “UpdateReservation” only for this profile, which uses also the value “Completion”.
- 5. *Loyalty* service which could be requested by the Sale System. It allows the service in the Payment message, and could use Loyalty, Balance Inquiry and Get Totals messages.
- 6. *Stored Value* service which could be requested by the Sale System. It could use Stored Value, Balance Inquiry and Get Totals messages.
- 7. *Card Reader* service, if the Sale System could request use of the POI Terminal card reader with the Card Reader messages.
- 8. *PIN* service, the Sale System could request the PIN validation with the PIN message.
- 9. *Communication*, when the terminal does not own communication device.

	Basic	Stand.	Extend	Card Acqu. Enab.	Batch	Defe rred	Res erv.	Loy.	Store Value	PIN	Card Read.	Com m.
Notation	B	S	E	I		O	R	L	V	P	C	
Card Acquisition				X								
Payment	X	X	X			X	X					
Batch					X						X	
Loyalty									X			
StoredValue										X		
Reversal	X	X	X									
Display		X*	X									
Sound		X*	X									
Print		X*	X									
Input		X*	X				X					
Input Update		X*	X				X					
PIN											X	
Transmit										X		X
Card Reader												X
Login/Logout	X	X	X									
EnableService					X							
Reconciliation	X	X	X									
Balance Inquiry									X	X		
GetTotals			X						X	X		
TransactionStatus	X	X	X									
Abort	X	X	X									
Diagnosis		X	X									
Event	X	X	X									
Admin	X	X	X									

* These device services are available from the POI to Sale direction only.

Table 15: Generic and Services Profiles per Messages

Profiles of the Sale System and POI System component are configuration parameters which are exchanged during the Login request and response messages.

Each Sale Terminal and POI Terminal has the following parameters:

Name	Generic Profile
Definition	Type of interface the Sale or the POI Terminal can manage.
Usage	Indicates what group of messages of the protocol are used or provided
Specification	4.1.2 Profiles

Name	Service Profile
Definition	List of optional services the Sale or the POI Terminal can manage.
Usage	List (possibly empty) of services which are implemented either in term of messages, either in term of optional part of the messages.
Specification	4.1.2 Profiles

Configuration 8: Sale and POI Terminal Profiles

4.2 Systems Synchronisation

The section specifies the part of administrative services related to the synchronization of systems.

It describes the management of session between systems, specifies the synchronization messages Login, Logout and Enable Service, and ends with a global state diagram of the synchronization.

4.2.1 Session Management

This section presents the management of the synchronisation between two components of the Sale System and the POI System.

The two following sections present the specifications of messages which are used to synchronise this synchronisation: the Login and Logout messages.

The SaleToPOI protocol is built on the set of messages presented on the section *Presentation of the Interface*, which are exchanged between two components of the Sale and the POI System.

To exchange messages, two components have to be previously logged together with the help of *Login* request and response messages exchange. The link is broken by one of the following events:

- The exchange of a *Logout* request and response messages.
- The exchange of a *Login* request and response messages from the same Sale component.
- When the state of the *Login* exchange is lost by the Sale or the POI component, depending on the implementation of these Systems (e.g. stopping of the component application).

The period of time between a *Login* and a *Logout* exchange is called a *Session*.

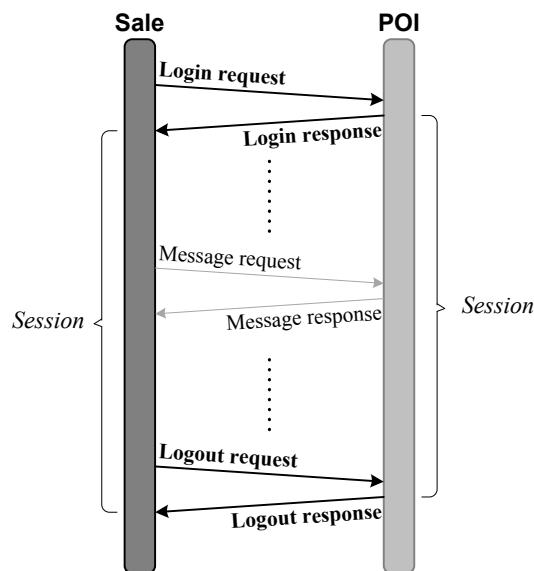


Figure 105: Session Between two Components (Servers or Terminals)

There are two levels of Login, or two types of sessions:

- A general from the Sale System to the POI System, realised by the Sale Server to the POI Server. This session allow exchange of messages between the Servers, and declares the availability of the Sale System and the POI System to work together.
- A login for every pair of Sale-POI Terminals which have to work together.

Rule 1: The login between Systems is optional (depending on the configuration), the login between Terminals is mandatory.

Rule 2: Outside a session, only *Event Notification* messages may be sent by the POI.

Rule 3: If the POI Terminal has not received a Login request since the last Logout exchange or starting-up of the Terminal, the POI Terminal put the value “LoggedOut” in the data component *Response.ErrorCondition* of any message response (see section 4.6.4.2 *LoggedOut Error*). This error has to be logged by the Sale system, as all errors in response messages.

4.2.2 Interface State Diagram

The figure below presents the state diagram of the global process flow for the Sale to POI interface, with the messages defined in this chapter.

This framework is attached to a couple of Sale Terminal and POI Terminal which have to be linked by a Login session.

After the start-up of the Sale or the POI Terminal, or after the Logout service process, the Terminals are considered as *Closed*, and cannot be used to request or process messages on the Sale to POI interface. Excluding specific error cases, Terminals are shut-down on this state.

As described in the previous sections, establishment of a session is realised by the Login message exchange. The Sale System associates a Sale Terminal to a POI Terminal sending a Login request message, and then both Terminals enter to the *Opening* state.

After the processing of the Login, the POI Terminal sends a successful Login response, and the interface goes to the *Open* state. In this state, the Sale Terminal can close the session sending a Logout request message to go back to the *Closed* state through the *Closing* state, or sends a Login request message to open a new session passing again through the *Opening* state.

The Sale Terminal can send to the POI Terminal a Device request message, entering in the *Processing Device* state. At the end of the POI process, the device operation is complete, and they go back to the *Open* state.

In the *Open* state, the Sale Terminal can also send to the POI Terminal a Service request message, entering in the *Processing Transaction* state. At the end of the POI process, the transaction is complete, and they go back to the *Open* state.

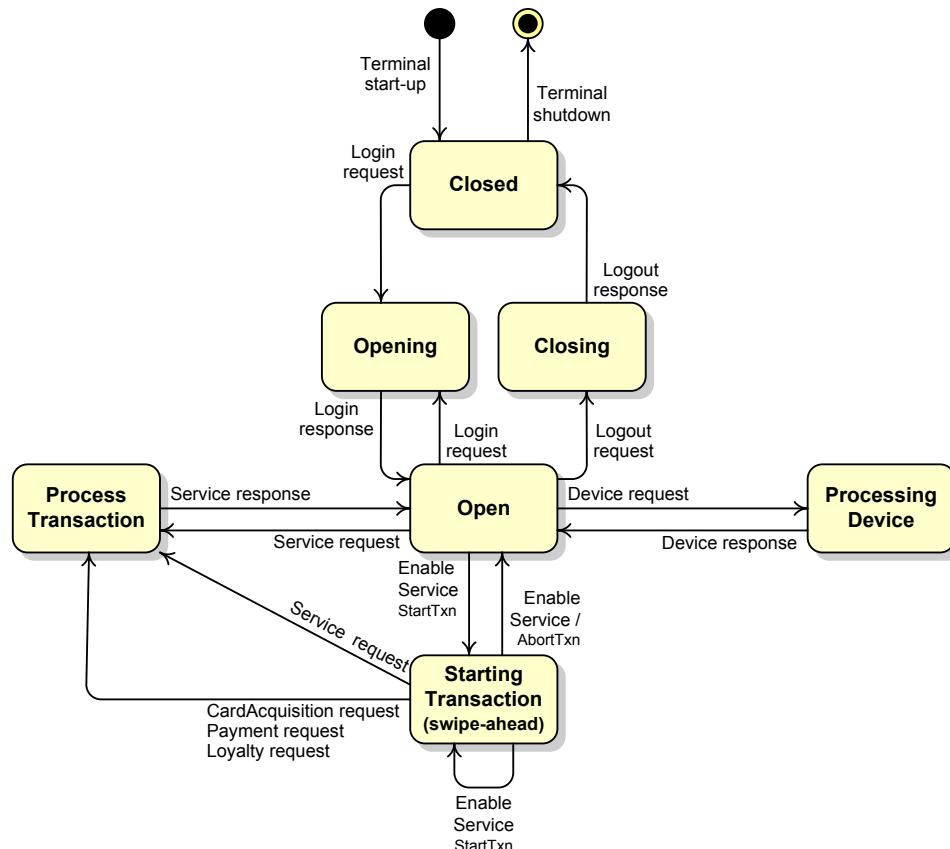


Figure 106: Sale to POI Interface State Diagram

To start the transaction with a “swipe-ahead” mechanism, i.e. to process in parallel the POI transaction and the Sale transactions, the Sale Terminal sends an EnableService request message (with *TransactionAction* =“StartTransaction”), entering in the *Starting Transaction* state to start-up the transaction if requested by the Cardholder. When the Sale System requests the Service, the POI System synchronises and processes the service, and the Terminals enter in the *Processing Transaction* state¹⁹ .

Because of the additional complexity of the error management (in particular error handling depends on the type of Service or Device exchanges), error states and transitions are not presented in the state diagram. See the section 4.7.5.1 *Error Resolutions Specifications on the POI System* for the POI error management, and see section 4.7.5.2 *Error Resolutions Specifications on the Sale System* for the Sale error management.

¹⁹ See section 4.2.5 *Enable Service Messages* for detailed processing of EnableService and “swipe ahead”.

4.2.3 Login

4.2.3.1 Login Processing

Login message is initiated by the Sale System after the occurrence of some event like the change of Cashier, the end of initialisation of the Sale system or the POI System, and because request of service to the POI Terminal have to be made by the Sale System.

Authentication of a Cashier or a supervisor is not the purpose of the *Login*, because these kinds of authentication are made locally on the Sale System and before the association to the POI Terminal.

Rule 1: If configured, the Sale Server has to be successfully logged to the POI Server, before a Sale Terminal may be logged to a POI Terminal.

If the Sale System is not logged, the POI Terminal has to send back a Login response with *Result*="Failure" and *ErrorCondition*="LoggedOut" (see section 4.6.4.2 *LoggedOut Error*).

Rule 2: Several Sale Terminals could be logged in to one POI Terminal, and one Sale Terminal could be logged in to several POI Terminals.

Rule 3: The POI may send *Display*, *Input* or *Print* messages during the processing of the *Login* request message, before sending the *Login* response.

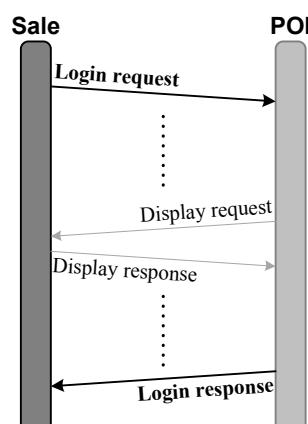


Figure 107: Login Exchange

Display could be used to display the status, at each step of the login process, or to display some warning or errors. *Input* could be used to ask a question to the Cashier in case of interactive login.

Rule 4: Logout exchange can be skipped by a new Login. The POI shall accept a Login request message from the same Sale Component already logged in. All information of the new Login replaces those stored during the previous Login.

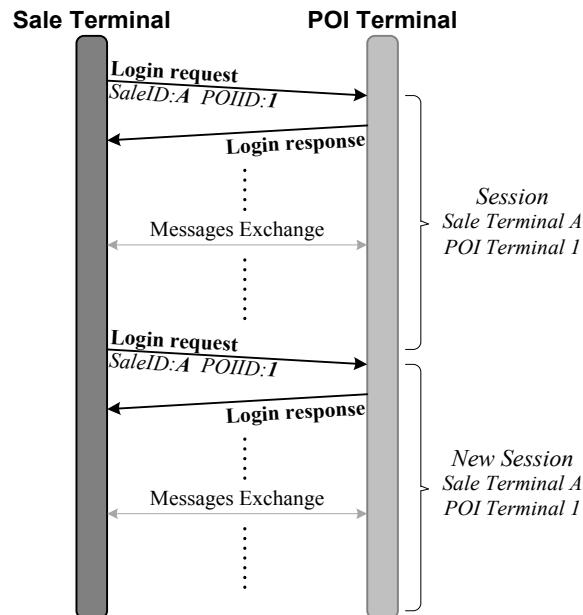


Figure 108: Login without Logout

Rule 5: Login cannot be aborted, but supports retry. If the Sale System does not receive a response, it can send another Login with a new message identifier *ServiceID*. Then the POI has not to answer to the first Login message, but only to the second one.

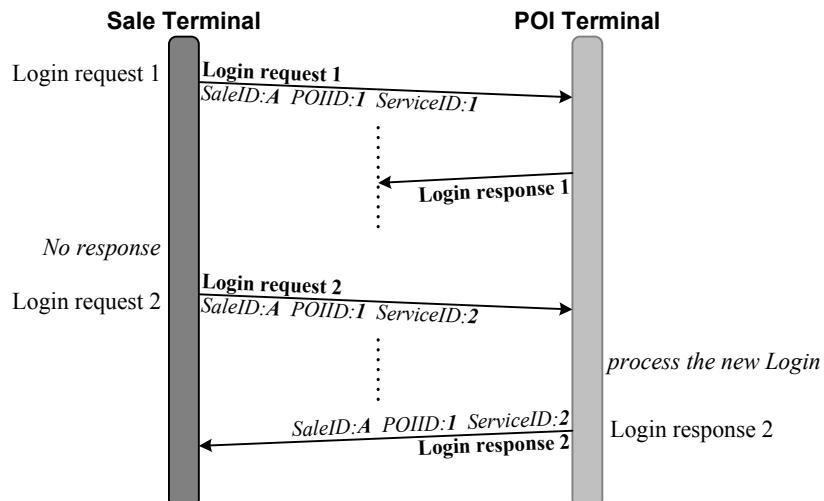


Figure 109: Login Retry

- Rule 6: Another error resolution for no reception of Login response is to send a Logout. Then the POI has not to answer to the first Login message, but only to the Logout, after stopped the Login process and completed the Logout process.

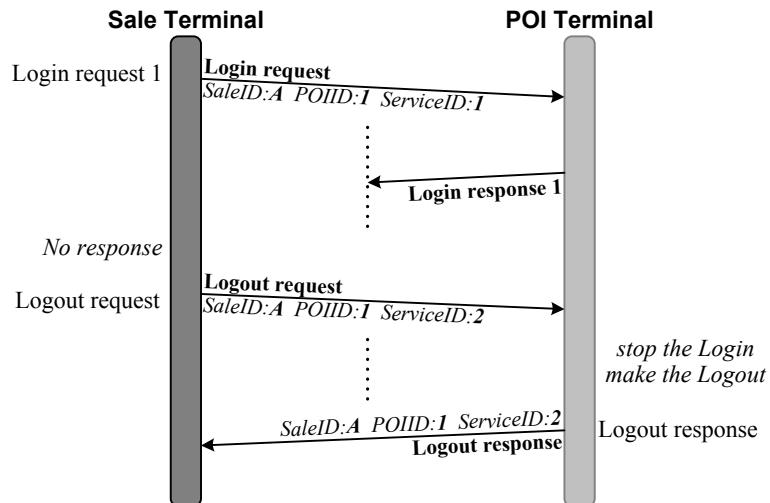


Figure 110: Logout after no Login Response

- Rule 7: The POI receives from the same Sale terminal, a Login request during the processing of the previous Login request which is not completed. The POI must continue the Login processing and answer to the last Login.

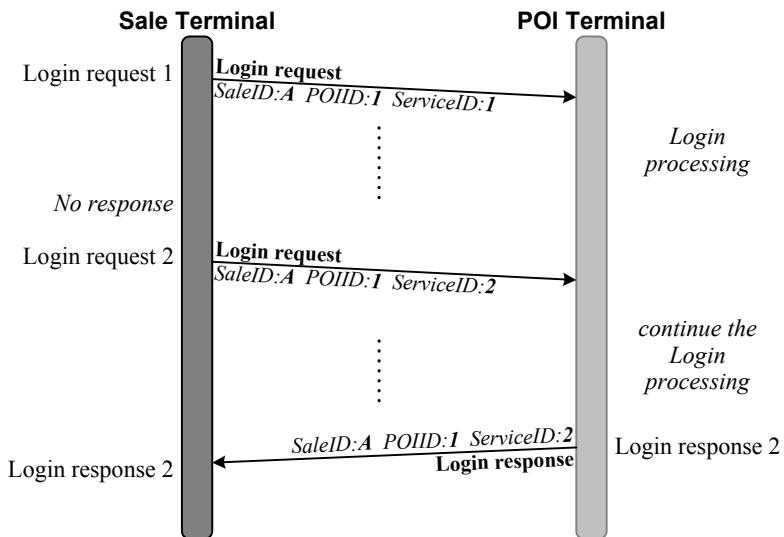


Figure 111: Double Login Processing

- Rule 8: When the cash handling machine managed by the POI does not have any more coins or bills of a certain value, the Login response must contain the related occurrence of *CoinsOrBills* with *Number* equal to 0 (in order to have the possible type of coins or notes accepted by the cash machine).

The Sale System and the POI System have the following configuration parameters:

Name	Server Login
Definition	The Sale System sends a Login to the POI System before any Terminal Login.
Usage	Allows a dialogue between the Sale Server and the POI Server.
Specification	4.2.1 Session Management and 4.2.3 Login

Configuration 9: Server Login

Name	Time Synchronisation
Definition	The POI System and POI Terminals synchronise their clock at the Login.
Usage	
Specification	4.2.1 Session Management and 4.2.3 Login

Configuration 10: Time Synchronisation

4.2.3.2 Presentation of the Messages

The Login request message body *LoginRequest*, contains the following information:

1. The date and time of the Sale System or the Sale Terminal: *DateTime*. It allows the POI to know the time of the Sale Server or Terminal.
2. The data structure *SaleSoftware*, containing information on the Sale software product: the product's name, vendor's name, the software version, and the certification code.
3. The data structure *SaleTerminalData*, containing characteristics of the Sale Terminal: the attendance environment, the capabilities in term of devices, the profile, and a possible group ID to classify the Terminal. This information is not used for a server login.
4. Some default values of Sale data elements used during the session, avoiding their insertion in all messages: the cashier ID and language, shift number, the training characteristic of the session.

The request contains also the serial number of the POI Terminal provided in the response of the last login for the POI Terminal attached to the identifier *POIID*.

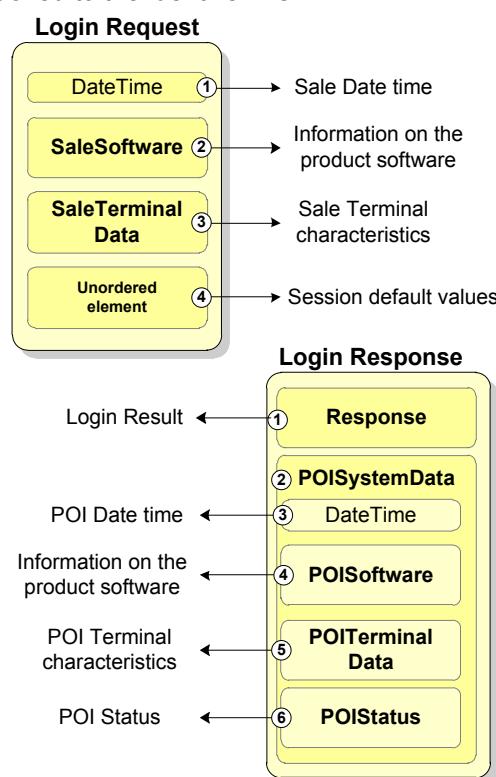


Figure 112: Login Request/Response Information

The Login response message body *LoginResponse*, contains the following information:

1. The result of the Login: *Response*.
2. The data structure *POISystemData*, containing the POI information presented below if the response is positive.
3. The date and time of the POI System or the POI Terminal: *DateTime*. It allows the Sale to know the time of the POI Server or Terminal, with possible synchronisation.
4. The data structure *POISoftware*, containing information on the POI hardware / software product: the product's name, vendor's name, the software version, and the certification code.
5. The data structure *POITerminalData*, containing characteristics of the POI Terminal: the copy of the attendance environment, the capabilities in term of devices, the profile, and the serial number of the POI Terminal. This information are not used for a server login.

6. The data structure *POITerminalData*, containing the global status of the POI Terminal and the status of its hardware and software modules.

4.2.3.3 Login Request Layout

<i>LoginRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
ProtocolVersion	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Login
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
LoginRequest	[1..1]		
DateTime	[1..1]		
SaleSoftware	[1..1]		
ManufacturerID	[1..1]		
ApplicationName	[1..1]		
SoftwareVersion	[1..1]		
CertificationCode	[1..1]		
SaleTerminalData	[0..1]		Present if the login involves a Sale Terminal
TerminalEnvironment	[1..1]		
SaleCapabilities	[1..1]		
SaleProfile	[0..1]		If at least one element is present
GenericProfile	[0..1]		default Standard
ServiceProfiles	[0..1]		If a service profile could be requested
TotalsGroupID	[0..1]		If present, default value for all transactions.
TrainingModeFlag	[0..1]	S	default False
OperatorLanguage	[1..1]		Default value for Device type displays
OperatorID	[0..1]	S	Four conditions to send it: a) the Sale System wants the POI log it in the transaction log b) because of reconciliation with total on OperatorID c) because the POI needs it d) acquirer or issuer need it
ShiftNumber	[0..1]	S	Same as OperatorID
TokenRequestedType	[0..1]		If a token is requested during the session.
CustomerOrderReq	[0..1]		If customer orders must be listed in Card Acquisition, Reversal and Payment response messages during the session.
POISerialNumber	[0..1]	S	If the login involve a POI Terminal and not the first Login to the POI System

The components *SaleTerminalData* and *POISerialNumber* are absent for Login between Servers.

The *SaleProfile* is absent if the Generic Profile is the default ("Standard"), and the Sale has no additional Services.

4.2.3.4 Login Response Layout

<i>LoginResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
ProtocolVersion	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Login
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
LoginResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>Same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>Same as PaymentResponse</i>
POISystemData	[0..1]		if Response.Result is Success
DateTime	[1..1]		
POISoftware	[1..1]		
ManufacturerID	[1..1]		
ApplicationName	[1..1]		
SoftwareVersion	[1..1]		
CertificationCode	[1..1]		
POITerminalData	[0..1]		Present if the login involves a POI Terminal
TerminalEnvironment	[1..1]		Copy
POICapabilities	[1..1]		
POIProfile	[0..1]		If at least one element is present. The Sale System decides if it can continue or not.
GenericProfile	[0..1]		default Standard
ServiceProfiles	[0..1]		If a service profile could be requested
POISerialNumber	[1..1]		
POIstatus	[0..1]		if Response.Result is Success
GlobalStatus	[1..1]		
SecurityOKFlag	[0..1]		If security module present
PEDOKFlag	[0..1]		If PED present
CardReaderOKFlag	[0..1]		If card reader device present
PrinterStatus	[0..1]		If printer device present
CommunicationOKFlag	[0..1]		If communication infrastructure present
CashHandlingDevice	[0..n]		If cash handling devices present.
CashHandlingOKFlag	[1..1]		
Currency	[1..1]		
CoinsOrBills	[1..n]		
UnitValue	[1..1]		
Number	[1..1]		
FraudPreventionFlag	[0..1]		Default False
TokenRequestStatus	[0..1]		If token is managed by the POI

<i>LoginResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
CustomerOrderStatus	[0..1]		If the POI supports the CustomerOrder functionnality

The component *POISystemData* is present if the processing is successful (*Response.Result* has the value “Success”).

The component *POITerminalData* is absent for Login between Servers.

The *POIProfile* is absent if the Generic Profile is the default (“Standard”), and the POI has no additional Services.

4.2.3.5 Error Cases

When the Login request is successfully processed, the Login response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of an error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved or before to try another Login attempt.

UnavailableService

The *ProtocolVersion* sent by the Sale System is too old to be manageable by the version of protocol implemented in the POI (see section 4.6.4.3.1 *Too Old Protocol Version*).

This is a permanent error, which has to be resolved or before to try another Login attempt.

Busy

The POI System or the POI Terminal is temporary not available. Reason could be initialisation in progress, maintenance in progress (see section 4.6.5.1 *Busy Error*).

The Sale System has to try later another Login when the POI component becomes available.

DeviceOut

The POI element cannot start to work, because of a temporary error on a device (e.g. printer without paper), see section 4.6.3.1.1 *POI Temporary Unavailable*.

The POI element cannot start to work, because of a permanent error on a device (e.g. card reader out of order, pin-pad disconnected), see section 4.6.3.1.1 *POI Permanent Unavailable*.

4.2.3.6 Example

This is the Login of the Sale Terminal “SaleTermA” which is an attended cash register, requesting some loyalty processing to the POI Terminal “POITerm1”, and some *PIN* requests or *Card Reader* to manage payment cards local to the store.

MessageHeader		(message example 1)
ProtocolVersion	3.0	
MessageClass	Service	
MessageCategory	Login	
MessageType	Request	
ServiceID	498	
SaleID	SaleTermA	
POIID	POITerm1	
LoginRequest		
Date Time	2015-03-08T09:13:51.0+01:00	
SaleSoftware		
ManufacturerID	PointOfSaleCo	
ApplicationName	SaleSys	
SoftwareVersion	01.98.01	
CertificationCode	ECTS2PS001	
SaleTerminalData		
TerminalEnvironment	Attended	
SaleCapabilities	PrinterReceipt CashierStatus CashierError CashierDisplay CashierInput	
SaleProfile		
GenericProfile	Extended	
ServiceProfiles	Loyalty PIN CardReader	
OperatorLanguage	sp	
OperatorID	Cashier16	
ShiftNumber	2	
POISerialNumber	78910AA46010005	

The payment system manages a cash handling machine with coins of 2, 1, 0.50, 0.20, 0.10 and 0.05 euros.

MessageHeader		<i>(message example 2)</i>
ProtocolVersion	3.0	
MessageClass	Service	
MessageCategory	Login	
MessageType	Response	
ServiceID	498	
SaleID	SaleTermA	
POIID	POITerm1	
LoginResponse		
Response		
Result	Success	
POISystemData		
DateTime	2015-03-08T09:13:49.8+01:00	
POISoftware		
ManufacturerID	ElecFundsCo	
ApplicationName	POIsys	
SoftwareVersion	5.0.001	
CertificationCode	ECTS2PP001	
POITerminalData		
TerminalEnvironment	Attended	
POICapabilities	CustomerDisplay CustomerError MagStripe ICC CashHandling	
POIPprofile		
GenericProfile	Extended	
ServiceProfiles	Loyalty PIN CardReader	
POISerialNumber	78910AA46010005	
POIStatus		
GlobalStatus	OK	
SecurityOKFlag	True	
PEDOKFlag	True	
CardReaderOKFlag	True	
PrinterStatus	OK	
CommunicationOKFlag	True	
CashHandlingDevice		
CashHandlingOKFlag	True	
Currency	EUR	
CoinsOrBills		
UnitValue	2	
Number	8	
CoinsOrBills		
UnitValue	1	
Number	43	
CoinsOrBills		
UnitValue	0.50	
Number	19	
CoinsOrBills		
UnitValue	0.20	
Number	38	
CoinsOrBills		
UnitValue	0.10	
Number	27	
CoinsOrBills		
UnitValue	0.05	
Number	0	

4.2.4 Logout

4.2.4.1 Logout Processing

Logout exchange is initiated by the Sale System to put an end to the association (the session) between a Sale Terminal to a POI Terminal, or between the Sale System and the POI System.

- Rule 1: The Sale System has to wait for the response of the messages in progress before to send the *Logout* request. If there are transactions in progress on the POI Terminal or the Sale Terminal, the POI Terminal refuses the *Logout*, and a *Logout* response is sent with *Result*=“Failure” and *ErrorCondition*=“Busy” (see section 4.6.5.1.2 *POI Busy*).
- Rule 2: The POI may send *Display*, *Input* or *Print* message during the processing of the *Logout* request message, before to send the *Logout* response.

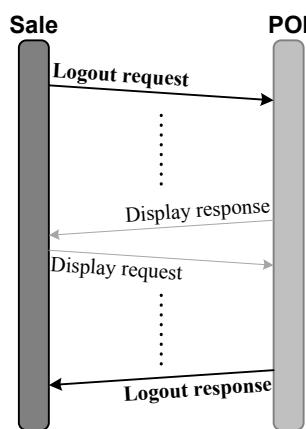


Figure 113: Logout Exchange

- Rule 3: The POI must accept *Logout* request messages if the Sale Terminal (or Sale System) was not logged in.

4.2.4.2 Presentation of the Messages

The *Logout* request message body *LogoutRequest* may contains:

1. The flag to allow maintenance after the closing of the session: *MaintenanceAllowed*.

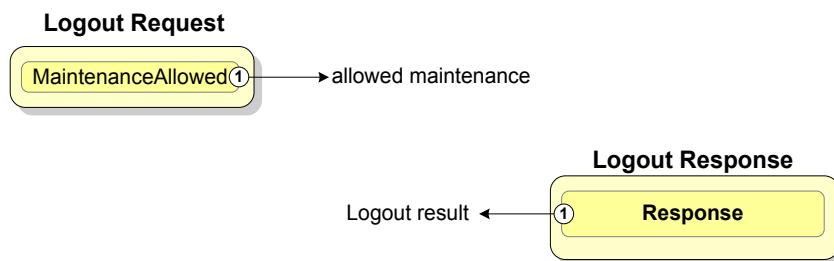


Figure 114: Logout Request/Response Information

The *Logout* response message body *LogoutResponse* contains:

2. The result of the *Logout*: *Response*.

4.2.4.3 Logout Request Layout

<i>LogoutRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Logout
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
LogoutRequest	[1..1]		
MaintenanceAllowed	[0..1]		default False

4.2.4.4 Logout Response Layout

<i>LogoutResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Logout
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
LogoutResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		Same as PaymentResponse
AdditionalResponse	[0..1]		Same as PaymentResponse

4.2.4.5 Error Cases

When the Logout request is successfully processed, the Logout response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved or before to try another Logout attempt.

Busy

There are transactions in progress in the POI Terminal or the Sale Terminal (see section 4.6.5.1 *Busy Error*).

The Sale System has to wait, terminate, or abort these transactions, and try later another Logout later, after the termination of these transactions.

4.2.4.6 Example

This is the Logout of the pair of Terminals “SaleTermA” and “POITerm1”.

MessageHeader		(message example 3)
MessageClass	Service	
MessageCategory	Logout	
MessageType	Request	
ServiceID	613	
SaleID	SaleTermA	
POIID	POITerm1	
LogoutRequest		
MaintenanceAllowed	True	

MessageHeader		(message example 4)
MessageClass	Service	
MessageCategory	Logout	
MessageType	Response	
ServiceID	613	
SaleID	SaleTermA	
POIID	POITerm1	
LogoutResponse		
Response		
Result	Success	

4.2.5 Enable Service Messages

4.2.5.1 Presentation of the Messages

The Enable Service request message body *EnableServiceRequest*, contains the following information:

1. The indicator *TransactionAction*, to start a swipe-ahead transaction or to abort the current transaction.
2. *ServicesEnabled*, to specify the financial services which are enabled by the message: Payment, Loyalty, CardAcquisition.
3. *DisplayOutput*, to display a message to the Customer on the POI Terminal.

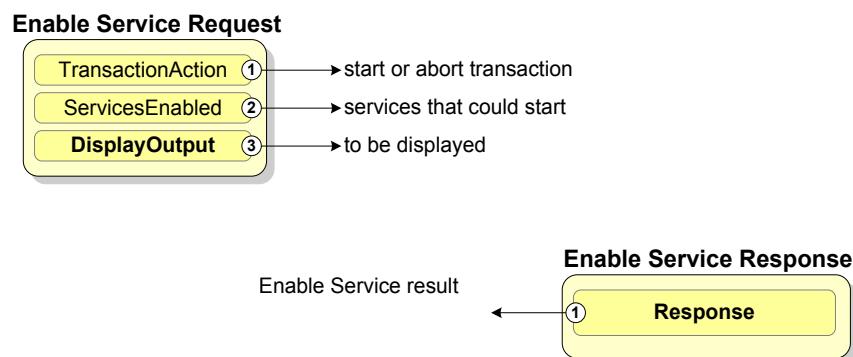


Figure 115: Enable Service Request/Response Information

The Enable Service response message body *EnableServiceResponse*, contains only one data structure:

1. The result of the Enable Service transaction: *Response*.

4.2.5.2 Enable Service Request Layout

<i>EnableServiceRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		EnableService
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
EnableServiceRequest	[1..1]	I	
TransactionAction	[1..1]		
ServicesEnabled	[0..1]		Mandatory if TransactionAction is "StartTransaction", absent if not.
DisplayOutput	[0..1]		Prompt or welcome message
Device	[1..1]		CustomerDisplay
InfoQualify	[1..1]		Display
OutputContent	[1..1]		
OutputFormat	[1..1]		
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartRow	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display.

4.2.5.3 Enable Service Response Layout

<i>EnableServiceResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		EnableService
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
EnableServiceResponse	[1..1]	I	
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>

4.2.5.4 Enable Service Processing

The Enable Service message request is sent for two purposes:

1. To enable the start of a payment, loyalty or stored-value transaction before the request of the Sale System, i.e. enable the starting of a swipe-ahead. The service is enabled until the next service request from the Sale System. This command always reinitialises the transaction context, i.e. information of an uncompleted previous swipe-ahead transaction is lost.
2. To reinitialise the POI context of a transaction after the processing of a CardAcquisition or EnableService message pair which has not been used for a transaction.

Rule 1: A swipe-ahead transaction starts by the Enable Service request message with *TransactionAction* set to "StartTransaction" and *ServicesEnabled* providing the accepted service from the Sale Terminal.

Rule 2: A swipe-ahead transaction is completed either by

- a) The CardAcquisition, Payment or Loyalty message exchange, as specified in the *ServicesEnabled* component of the Enable Service request message, or
- b) Enable Service request message with *TransactionAction* set to "AbortTransaction" and *ServicesEnabled* absent.

The case a) is the standard completion of a swipe-ahead transaction:

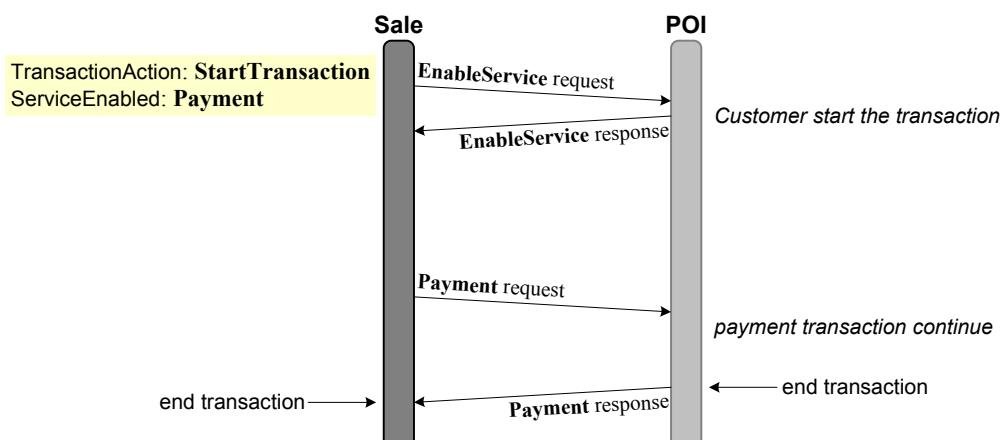


Figure 116: Standard Swipe-Ahead Completion

The case b) is used if a started swipe-ahead transaction is not completed by a Service request from the Sale Terminal, to avoid using the card of the previous Customer:

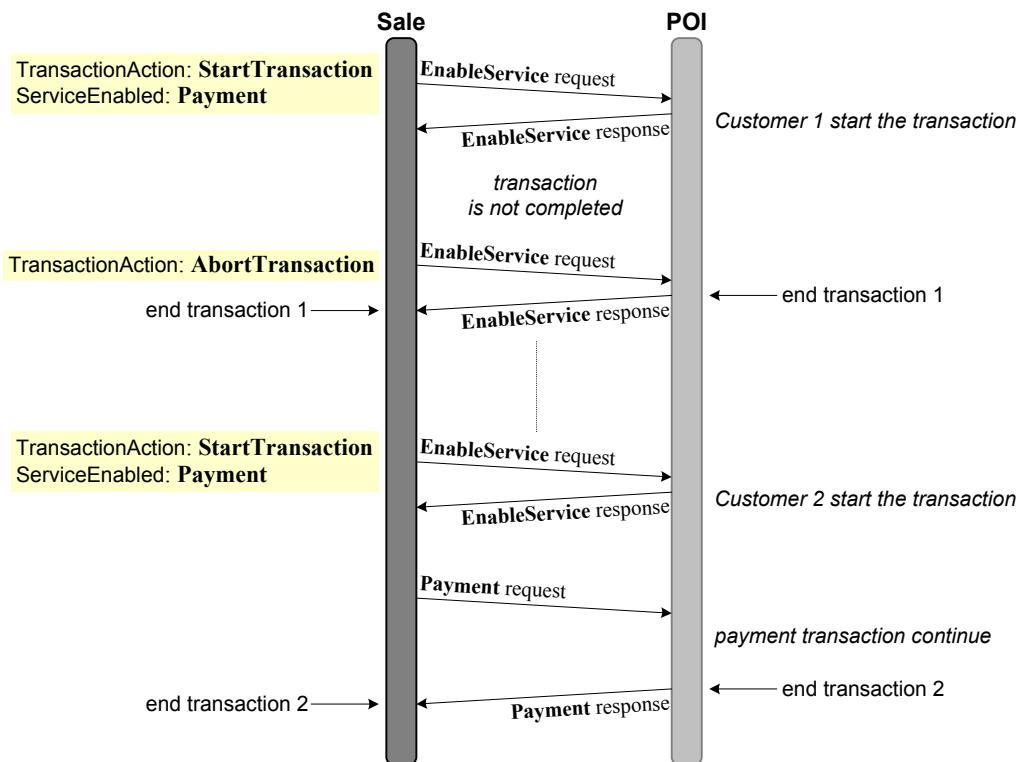


Figure 117: Swipe-Ahead Abort

Rule 3: The reception of an Enable Service request message with *TransactionAction* set to "StartTransaction" on an already started swipe-ahead transaction before the reception of a Service request closes the old transaction and starts a new one.

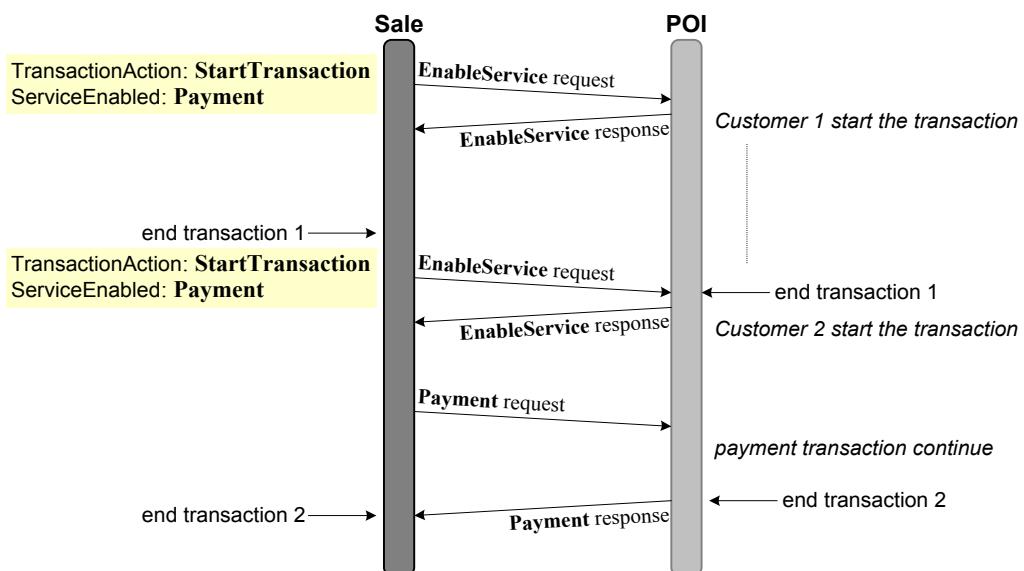


Figure 118: Swipe-Ahead Repetition

Rule 4: The reception of an Enable Service request message during a Service or Device dialogue produces the sending of Enable Service response with *Result*= "Failure" and *ErrorCondition*= "NotAllowed" (see section 4.6.4.1.1 *Forbidden Dialogue*):

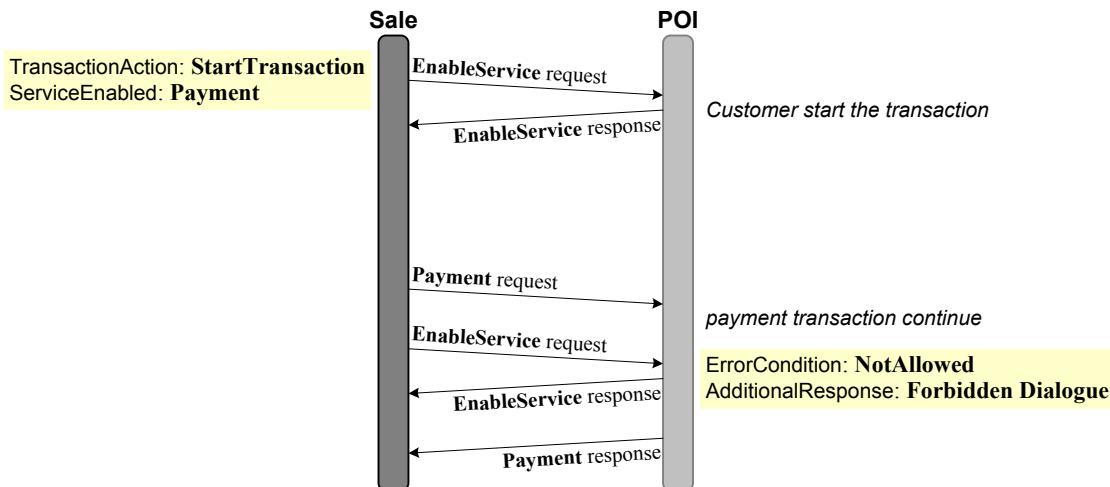


Figure 119: EnableService on a Transaction in Progress

Rule 5: The reception of an Enable Service request message with *TransactionAction* set to "AbortTransaction" on an already finished transaction produces the sending of Enable Service response with *Result*= "Failure" and *ErrorCondition*= "NotAllowed" (see section 4.6.4.1.3 *Completed Transaction*):

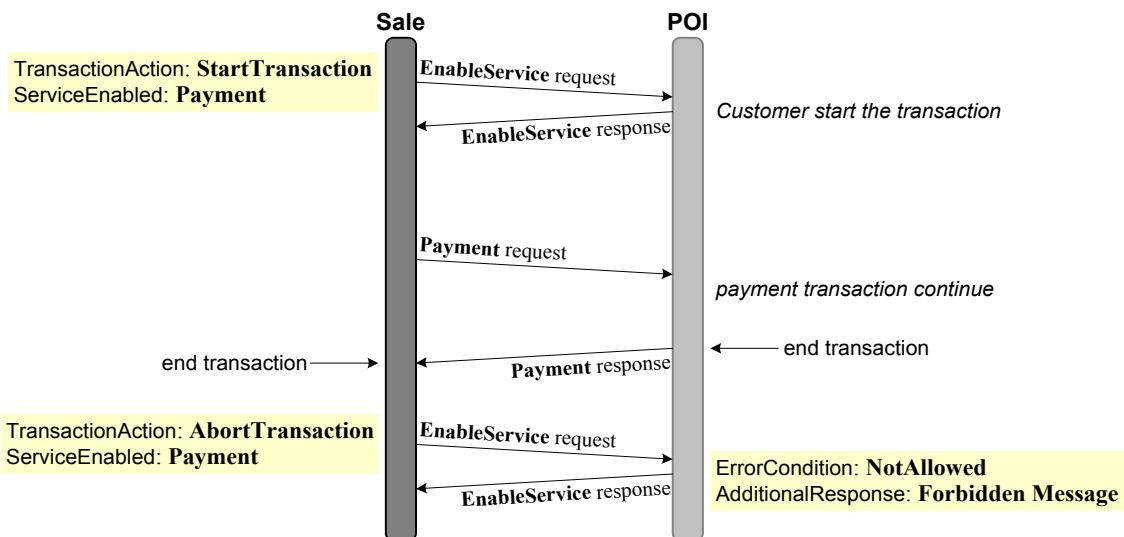


Figure 120: Close a Completed Transaction

Rule 6: A swipe-ahead transaction is in progress, but the Service request among the *ServicesEnabled* defined in the Enable Service request is not received yet.
 The reception of a Device request message or a Service request not related to the current *ServicesEnabled* closes implicitly the transaction in progress.

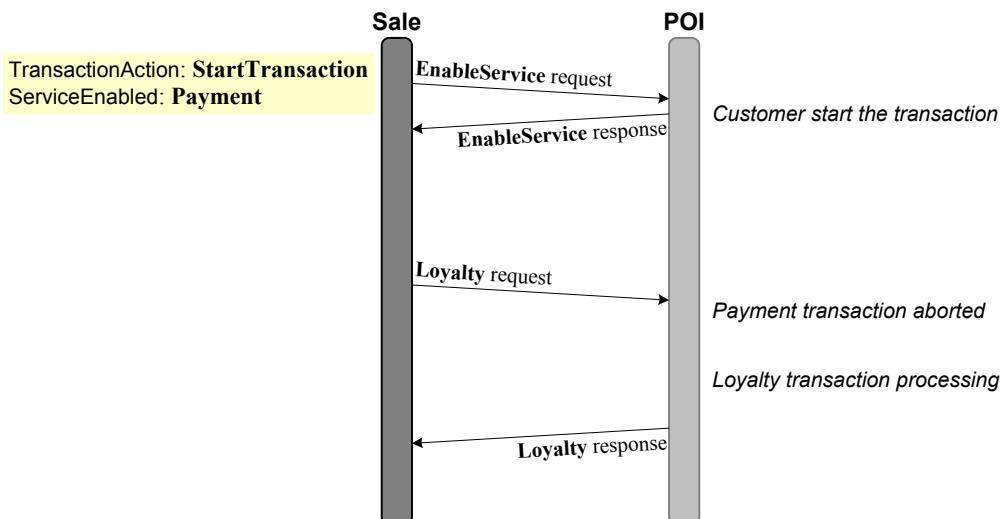


Figure 121: Swipe-Ahead Aborted by a Device or Service Request

Rule 7: If the Enable Service response is not received by the Sale System for some reason, the Sale System must follow the error resolution of idempotent messages (see section 4.7.5.1 *Error Resolutions Specifications*):

If there is no response to the Enable Service request:

- If the previous transaction was not completed, it could take some time to terminate the transaction because of external action (e.g. removing the card). But there is no incidence for the transaction to send another Enable Service request.
- If the previous transaction was completed the response will be sent quite soon. In this case there is no incidence for the transaction to send another Enable Service request as well.

4.2.5.5 Error Cases

When the Enable Service request is successfully processed, the Enable Service response message gets the value "Success" in the data element *Response.Result*, and the value "Failure" in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

UnavailableService

The Enable Service administrative service is not available in the POI System (see section 4.6.4.3.3 Unavailable Administrative Service).

NotAllowed

The reception of an Enable Service request message during a Service or Device dialogue (see section 4.6.4.1.1 *Forbidden Dialogue*).

The reception of an Enable Service request message with *TransactionAction* set to "AbortTransaction" on an already finished transaction (see section 4.6.4.1.3 *Completed Transaction*).

4.2.5.6 Example

Here is a simple example of EnableService exchange to allow the processing of a payment transaction before the Payment request of the Sale system (the famous “swipe ahead” situation).

MessageHeader		(message example 5)
MessageClass	Service	
MessageCategory	EnableService	
MessageType	Request	
ServiceID	640	
SaleID	SaleTermA	
POIID	POITerm1	
EnableServiceRequest		
TransactionAction	StartTransaction	
ServicesEnabled	Payment	
DisplayOutput		
Device	CustomerDisplay	
InfoQualify	Display	
OutputContent		
OutputFormat	Text	
OutputText		
Text	Plese, enter your card	

MessageHeader		(message example 6)
MessageClass	Service	
MessageCategory	EnableService	
MessageType	Response	
ServiceID	640	
SaleID	SaleTermA	
POIID	POITerm1	
EnableServiceResponse		
Response		
Result	Success	

4.2.6 Admin

4.2.6.1 Admin Processing

Admin exchange is initiated by the Sale System to select and start customised administrative services provided by the POI, using a "menu" for an interactive or software interface.

Rule 1: The standard procedure of Admin processing without the *ServiceIdentification* data element in the request is:

1. The Sale System sends an *Admin* request message to the POI.
2. The POI sends a sequence of some *Input* message with *InputCommand* "GetMenuItem", to allow the Cashier or the Sale software to select an administrative service.
3. The POI may send some *Input* message to request some additional information to process the selected service, or a password.
4. The POI processes the selected service, and might send some *Display*, *Input* or *Print* message to present the result or ask additional options.
5. After processing the selected service, the POI might:
 - a. End the service by sending the *Admin* response message to the Sale System, or
 - b. Continue the Admin command by proposing another menu item (going to step 2).

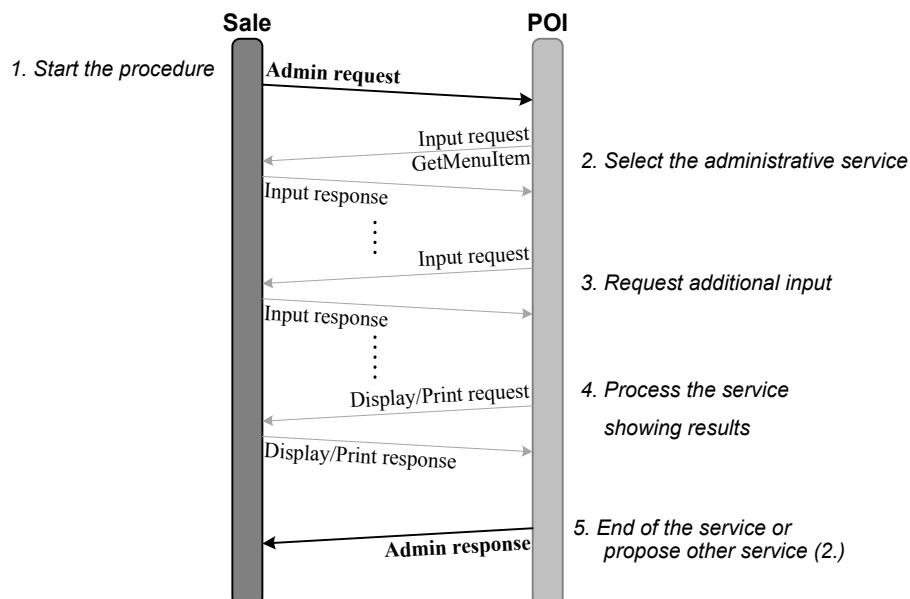


Figure 122: Admin Standard Process

Rule 2: The procedure of Admin when the request contains *ServiceIdentification* is:

1. The Sale System sends an *Admin* request message to the POI.
2. The POI processes the service identified by *ServiceIdentification*, and may send some *Input* message to request some additional information or a password.
3. The POI processes the selected service, and might send some *Display*, *Input* or *Print* message to present the result or ask additional options.
4. After processing the selected service, the POI ends the service by sending the *Admin* response message to the Sale System.

Rule 3: If the service related to the chosen menu item or *ServiceIdentification* is not allowed or not found, the POI must send an Admin response with *Result*="Failure" and *ErrorCondition*="UnavailableService" (see section 4.6.4.3.3 *Unavailable Administrative Service*)

4.2.6.2 Presentation of the Messages

The Admin request message body *AdminRequest* contains:

1. The identification of a service, if a service is requested directly without a menu:
ServiceIdentification.

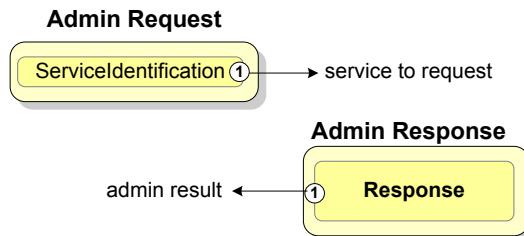


Figure 123: Admin Request/Response Information

The Admin response message body *AdminResponse* contains:

1. The result of the Admin request: *Response*.

4.2.6.3 Admin Request Layout

<i>AdminRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Admin
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
AdminRequest	[1..1]		AdminRequest
ServiceIdentification	[0..1]		If direct service to process

4.2.6.4 Admin Response Layout

<i>AdminResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Admin
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
AdminResponse	[1..1]		AdminResponse
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>Same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>Same as PaymentResponse</i>

4.2.6.5 Error Cases

When the Admin request is successfully processed, the Admin response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved or before to try another Logout attempt.

Busy

There are transactions in progress in the POI Terminal or the Sale Terminal (see section 4.6.5.1 *Busy Error*).

The Sale System has to wait, terminate, or abort these transactions, and try later another Admin later, after the termination of these transactions.

UnavailableService

The service related to the chosen menu item or *ServiceIdentification* is not allowed or not found.

4.2.6.6 Example

This is an Admin exchange of messages, on the pair of Terminals “SaleTermA” and “POITerm1” where the POI system provides a daily payment report which can be transferred to the Sale by the Admin exchanges (this is a dedicated administration service offer by the POI system product out of scope of the protocol).

The operator, or the Sale system through the sale software, wants to get the “Daily Payment Report” of the POI System:

1. The operator sends an *Admin* request message to the POI to have the administrative menu.
2. The POI sends an Input request message displaying the menu in *OutputContent* and *InputCommand*=“GetMenuItem” to have selection of the menu item (see example c) *Menu Item Selection* page 392).
3. The operator selects the item “1: Daily Payment Report” and sends the Input response message (see example c) *Menu Item Selection* page 392).
4. The POI performs the daily payment report, delivers it to the operator, and sends the Admin response message. The delivery is not presented in the example.

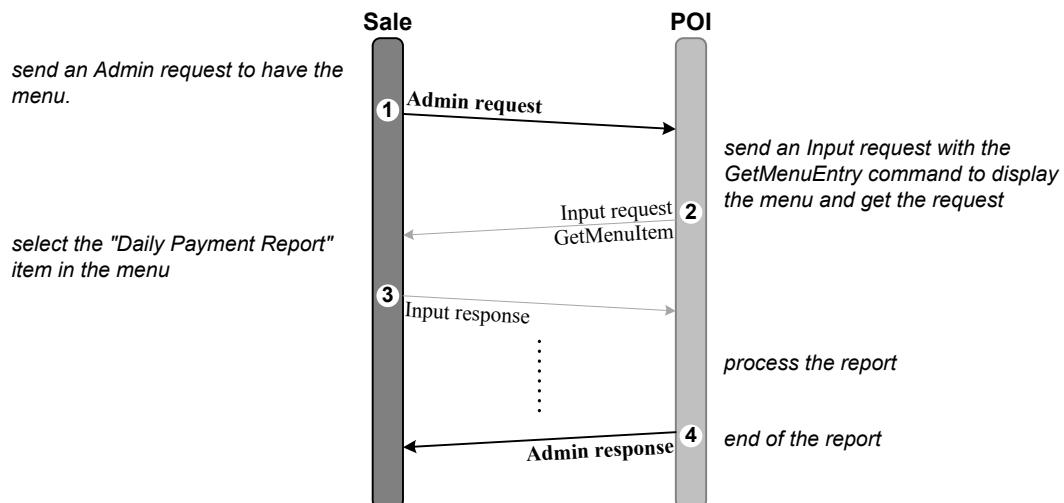


Figure 124: Daily Payment Report Example

MessageHeader		(message example 7)
MessageClass	Service	
MessageCategory	Admin	
MessageType	Request	
ServiceID	613	
SaleID	SaleTermA	
POIID	POITerm1	
AdminRequest		

MessageHeader		(message example 8)
MessageClass	Service	
MessageCategory	Admin	
MessageType	Response	
ServiceID	613	
SaleID	SaleTermA	
POIID	POITerm1	
AdminResponse		
Response		
Result	Success	

4.3 Financial Services

4.3.1 Standard Payment

4.3.1.1 Transaction Identification

Every transaction is identified by the Sale System and the POI System. Transaction identification is a data structure with the type *TransactionIdentificationType*, which includes two data elements:

1. An ID *TransactionID*, to identify the transaction for the Sale or the POI.
2. The date and time of the transaction *TimeStamp*, allowing the uniqueness of the identification.

In the Payment request, the Sale Terminal sends an identification of the Sale transaction in *SaleData.SaleTransactionID*, in the Payment response the POI Terminal sends the identification of the payment transaction in *POIData.POITransactionID*.

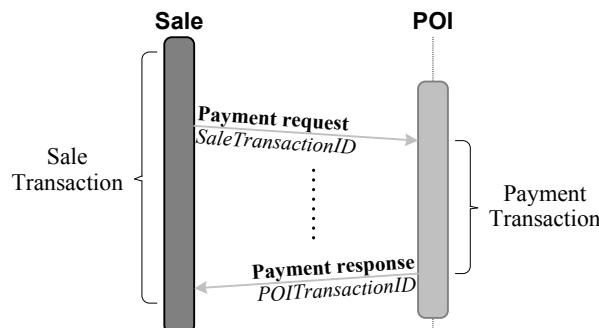


Figure 125: Transaction Identifiers

Rule 1: The payment identification of the transaction *POITransactionID* sent in the response of a Payment message is unique for the POI Terminal identified by *POIID*. This is mandatory in every message response, even with a format error.

Rule 2: A new POI transaction identification *POITransactionID* has to be generated by the POI for every Payment message exchange. Exceptions to this rule are the Payment response with the *PaymentResult.PaymentType* set to “UpdateReservation” or “Completion”, or when the Payment exchange is linked to the previous CardAcquisition exchange.

The POI transaction identification *POITransactionID* is used by the Sale System to link a Service message request to a previous Service messages exchange.

If the transaction related to the link has been realised on another POI, in addition to the *POITransactionID*, the link must contains the identification of the POI.

Rule 3: The sale transaction identification *SaleTransactionID* sent in the Payment request message is not necessarily unique. This is only an informative element to be logged by the POI System.

A Sale transaction may not have the same scope as a POI or payment transaction:

- A Sale transaction could cover several Payment transactions in case of split payments.
- Some Sale Terminal generates new transaction identification in case of successful transaction.
- A Sale transaction could make a payment without the request to the POI, e.g. by cash.

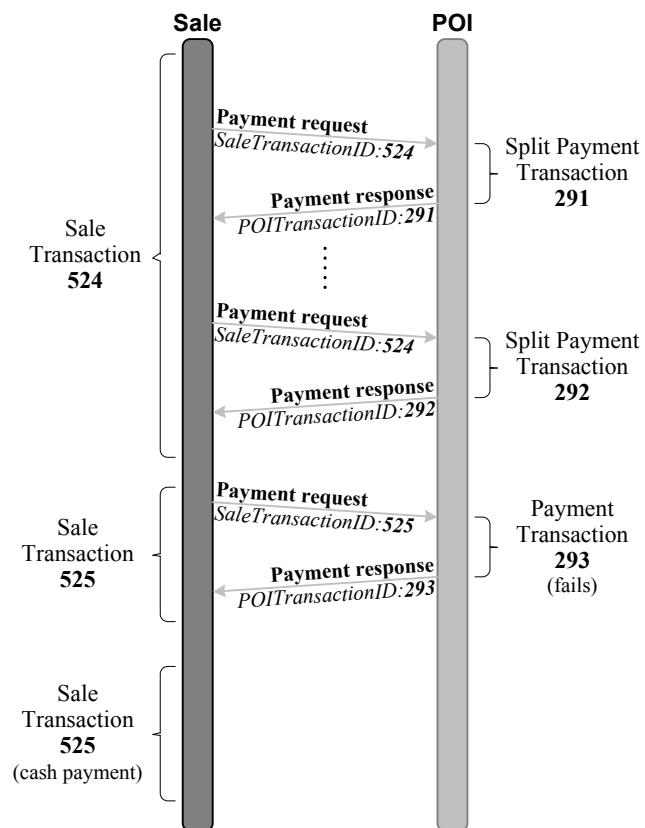


Figure 126: Sale and POI Transactions

4.3.1.2 Presentation of the Payment Messages

The Payment request message body *PaymentRequest*, contains the following information:

1. The data structure *SaleData*, containing the context of the Sale Terminal specific to this payment request:
 - a. The information which has changed since or was absent from the last Login request (*OperatorID*, *OperatorLanguage*...).
 - b. The identification of the transaction for the Sale System *SaleTransactionID*, containing the identification completed by a time stamp to ensure the uniqueness of the identification. It could be completed by a global identification *SaleReferenceID*, for some kind of payment as reservation.
 - c. Information not interpreted by the POI, for the log, the Acquirer or the Issuer.
2. The data structure *PaymentTransaction*, containing the information related to the transaction, global to the payment or the loyalty:
 - a. The amounts and the currency in the data structure *AmountsReq*.
 - b. Link to a previous transaction *OriginalPOITransaction*.
 - c. The data structure *TransactionConditions*, containing possible conditions or restrictions on the payment and loyalty: card brands, type of allowed loyalty transaction, merchant category...
 - d. The data structure *SaleItem*, containing sold items if they could be necessary for the payment card or the loyalty.
3. The data structure *PaymentData*, containing specific information to the payment:
 - a. The payment type *PaymentType*, which identifies the payment service, "Normal" for a standard payment.
 - b. Link to the previous card acquisition *CardAcquisitionReference*.
 - c. Information on the payment instrument of the generic data structure : *PaymentInstrumentData* containing *CardData*, *CheckData*, *MobileData*, or *StoredValueAccountID*, if the card is read by the Sale Terminal.
4. The repeated data structure *LoyaltyData*, containing specific information to the loyalty:
 - a. Link to the previous card acquisition *CardAcquisitionReference*.
 - b. The data structure *LoyaltyAccountID*, if the card is read by the Sale Terminal.
 - c. The data structure *LoyaltyAmount*, to store an amount on the loyalty account independently of the payment transaction.

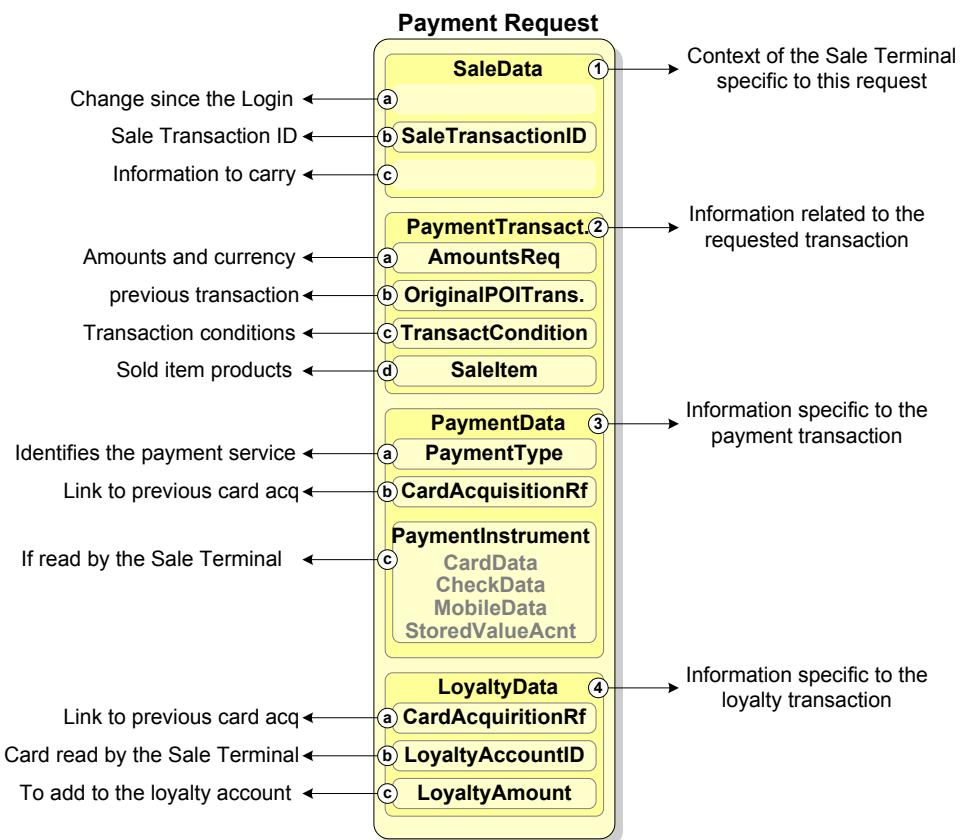


Figure 127: Payment Request Information

The Payment response message body *PaymentResponse*, contains the following information:

1. The result of the Payment: *Response*.
2. The data structure *SaleData*, containing the Sale transaction identification.
3. The data structure *POIData*, containing the POI transaction identification and the reconciliation identification.
4. The data structure *PaymentResult*, containing the result of the payment:
 - a. Information on the payment instrument of the generic data structure : *PaymentInstrumentData* containing *CardData*, *CheckData*, *MobileData*, or *StoredValueAccountID*.
 - b. The amounts of the response: *AmountsResp*.
 - c. The sold products which can be paid by the card: *AllowedProductCode*.
 - d. Identification of the transaction for the payment Acquirer: *PaymentAcquirerData*.
5. The repeated data structure *LoyaltyResult*, containing the result of the loyalty transactions:
 - a. Information on the loyalty account: *LoyaltyAccount*.
 - b. The amounts of the response: *LoyaltyAmount*.
 - c. Identification of the transaction for the loyalty host: *LoyaltyAcquirerData*.
 - d. Information on the rebates, related to the total or per sold items: *Rebates*.

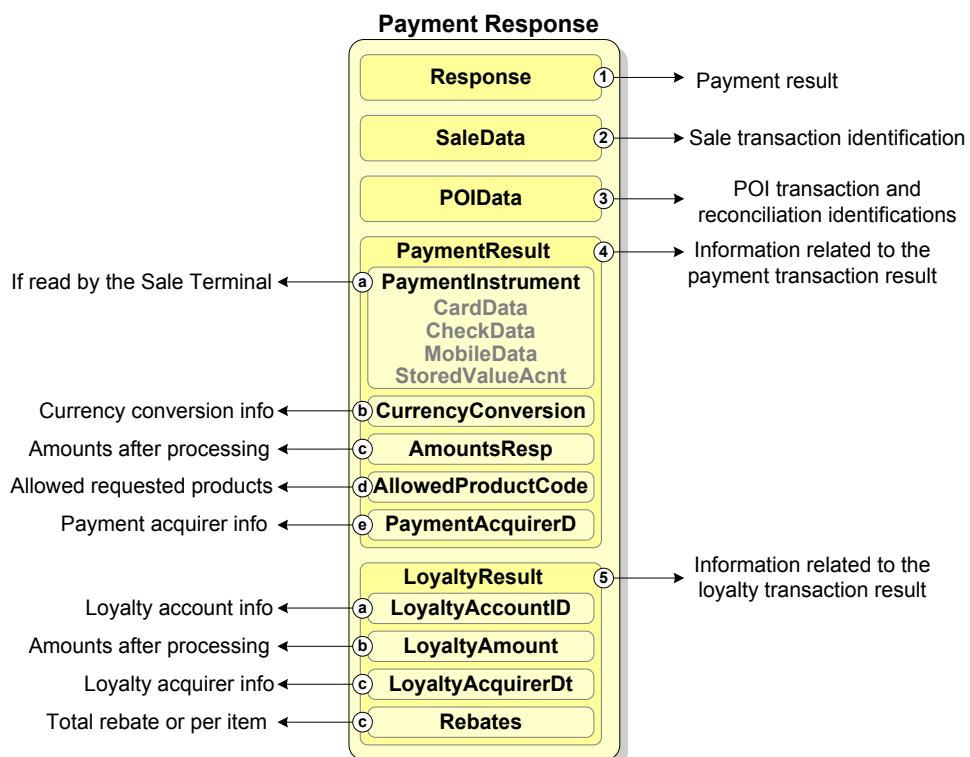


Figure 128: Payment Response Information

4.3.1.3 Payment Request Layout

<i>PaymentRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Payment
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
PaymentRequest	[1..1]		
SaleData	[1..1]		
OperatorID	[0..1]		if different from the Login and see <i>Login .SaleData</i>
OperatorLanguage	[0..1]		if different from the Login
ShiftNumber	[0..1]		if different from the Login and see <i>Login .SaleData</i>
SaleTransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
SaleReferenceID	[0..1]	R	Mandatory if payment reservation or if CustomerOrder is present
SaleTerminalData	[0..1]		If content is not empty
TerminalEnvironment	[0..1]		If modified since the login
SaleCapabilities	[0..1]		If modified since the login (devices failures)
TotalsGroupID	[0..1]		if modified since, or not in the login and used by the Sale System
TokenRequestedType	[0..1]		If a token is requested.
CustomerOrderReq	[0..1]		If customer orders must be listed in the response message.
SaleToPOIData	[0..1]		Stored with the transaction
SaleToAcquirerData	[0..1]		Send to the Acquirer if present
SaleTolssuerData	[0..1]		Send to the Acquirer if present
StatementReference	[0..1]		Information to print on the bank statement
PaymentTransaction	[1..1]		
AmountsReq	[1..1]		
Currency	[1..1]		
RequestedAmount	[0..1]	mandatory if not O	Absent only if the maximum amount is unknown for a OneTimeReservation. The value has to be greater than 0.
CashBackAmount	[0..1]	O	If payment with cash back requested by the Sale System.
TipAmount	[0..1]		If payment with tip requested by the Sale System.
PaidAmount	[0..1]		if SplitPaymentFlag is True
MinimumAmountToDeliver	[0..1]	O	if unknown maximum amount for a OneTimeReservation or minimum amount requested by the Sale System
MaximumCashBackAmount	[0..1]		Maximum amount which could be requested for

<i>PaymentRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
			cash-back to the Sale System.
MinimumSplitAmount	[0..1]		Minimum amount of a split, which could be requested.
OriginalPOITransaction	[0..1]	BOR	if Completion or Refund, optional for a reservation transaction.
POITransactionID	[0..1]		If SaleReferenceID is insufficient to identify the transaction.
TransactionID	[1..1]		
TimeStamp	[1..1]		
POIID	[0..1]		If original transaction is coming from another POI
ReuseCardDataFlag	[0..1]	BOR	default True
ApprovalCode	[0..1]		If referral
TransactionConditions	[0..1]		If one data element is present
AllowedPaymentBrand	[0..n]		Restrict brand if present
AcquirerID	[0..n]		Restrict to these Acquirer if present
DebitPreferredFlag	[0..1]		default False The preferred type of payment is a debit transaction rather than a credit transaction.
AllowedLoyaltyBrand	[0..n]	L	Restrict brand if present
LoyaltyHandling	[0..1]	L	default Forbidden
CustomerLanguage	[0..1]		If the language is selected by the Customer on the Sale System before the request to the POI.
ForceOnlineFlag	[0..1]		default False. Go online if data sent
ForceEntryMode	[0..n]		Restrict entry mode if sent and no PaymentData or LoyaltyData.CardAcquisitionReference
MerchantCategoryCode	[0..1]		The payment implies a specific MCC.
SaleItem	[0..n]	S L	If purchased products are required for the payment or loyalty
ItemID	[1..1]		
ProductCode	[1..1]		
EanUpc	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it
UnitOfMeasure	[0..1]		
Quantity	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it
UnitPrice	[0..1]		
ItemAmount	[1..1]		
TaxCode	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it
SaleChannel	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it
ProductLabel	[0..1]		
AdditionalProductInfo	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it
PaymentData	[0..1]		If one data element is present
PaymentType	[0..1]		default Normal
SplitPaymentFlag	[0..1]		default False if split of the amount with possible fleet cards.
RequestedValidityDate	[0..1]	OR	If time period of the OneTimeReservation, FirstReservation or UpdateReservation is requested

<i>PaymentRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
CardAcquisitionReference	[0..1]	E	if the card data comes from the previous CardAcquisition
TransactionID	[1..1]		
TimeStamp	[1..1]		
Instalment	[0..1]		If PaymentType is "Instalment" or "IssuerInstalment"
InstalmentType	[0..1]		
SequenceNumber	[0..1]		
PlanID	[0..1]		
Period	[0..1]		
PeriodUnit	[0..1]		
FirstPaymentDate	[0..1]		Mandatory if InstalmentType= "DeferredInstalments".
TotalNbOfPayments	[0..1]		
CumulativeAmount	[0..1]		
FirstAmount	[0..1]		If the first amount is different from the others (InstalmentType= "InequalInstalments")
Charges	[0..1]		
CustomerOrder	[0..1]		If related to a customer order
CustomerOrderID	[0..1]		
StartDate	[1..1]		
ForecastedAmount	[1..1]		
OpenOrderState	[0..1]		True by default
Currency	[0..1]		If multiple currencies are allowed.
AdditionalInformation	[0..1]		
PaymentInstrumentData	[0..1]		If payment instrument data is read by the Sale System
PaymentInstrumentType	[1..1]		
ProtectedCardData	[0..1]		SensitiveCardData protected by CMS EnvelopedData
CardData	[0..1]		If PaymentInstrumentType is "Card"
EntryMode	[1..1]		not (ICC or SynchronousICC)
SensitiveCardData	[0..1]		If structure non empty (could be CMS protected EnvelopedData)
PAN	[0..1]		if EntryMode is File, Keyed or Manual
CardSeqNumb	[0..1]		if EntryMode is File, Keyed or Manual and data available on the card
ExpiryDate	[0..1]		if EntryMode is File, Keyed or Manual and data available on the card
TrackData	[0..3]		if EntryMode is MagStripe or RFID
TrackNumb	[0..1]		default 2
TrackFormat	[0..1]		default ISO
TrackValue	[1..1]		
CheckData	[0..1]		If PaymentInstrumentType is "Check"
BankID	[0..1]		Mandatory if TrackData absent
AccountNumber	[0..1]		Mandatory if TrackData absent
CheckNumber	[0..1]		Mandatory if TrackData absent
TrackData	[0..1]		Mandatory if CheckNumber absent

<i>PaymentRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
TrackNumb	[0..1]		default 2
TrackFormat	[1..1]		"E-13B" or "CMC-7"
TrackValue	[1..1]		
CheckCardNumber	[0..1]		If provided by the customer
TypeCode	[0..1]		default Personal
Country	[0..1]		Absent if country of the Sale system
MobileData	[0..1]		If PaymentInstrumentType is "Mobile"
MobileCountryCode	[0..1]		If data available
MobileNetworkCode	[0..1]		If data available
MaskedMSISDN	[0..1]		If data available
Geolocation	[0..1]		If data available
GeographicCoordinates	[0..1]		
Latitude	[1..1]		
Longitude	[1..1]		
UTMCoordinates	[0..1]		
UTMZone	[1..1]		
UTMEastward	[1..1]		
UTMNorthward	[1..1]		
ProtectedMobileData	[0..1]		<i>SensitiveMobileData</i> protected by CMS EnvelopedData
SensitiveMobileData	[0..1]		If unprotected mobile data
MSISDN	[1..1]		
IMSI	[0..1]		If data available
IMEI	[0..1]		If data available
StoredValueAccountID	[0..1]		If PaymentInstrumentType is "StoredValue"
StoredValueAccountType	[1..1]		
StoredValueProvider	[0..1]		If available for the card or account.
OwnerName	[0..1]		If available for the card or account.
ExpiryDate	[0..1]		If required for the card or account.
EntryMode	[1..1]		
IdentificationType	[1..1]		
StoredValueID	[1..1]		
LoyaltyData	[0..n]	L	Loyalty cards used with the payment transaction and read by the Sale System
CardAcquisitionReference	[0..1]	E	see <i>Loyalty request</i>
TransactionID	[1..1]		
TimeStamp	[1..1]		
LoyaltyAccountID	[0..1]		see <i>Loyalty request</i>
EntryMode	[1..1]		
IdentificationType	[1..1]		
LoyaltyID	[1..1]		
LoyaltyAmount	[0..1]		When the Sale System want to make an additional award the Loyalty account
LoyaltyUnit	[0..1]		see <i>Loyalty request</i>
Currency	[0..1]		see <i>Loyalty request</i>
AmountValue	[1..1]		

4.3.1.4 Payment Response Layout

<i>PaymentResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Payment
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
PaymentResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		If Result is not Success, or on warning condition
AdditionalResponse	[0..1]		If present, the POI logs it for further examination
SaleData	[1..1]		
SaleTransactionID	[1..1]		Copy
TransactionID	[1..1]		
TimeStamp	[1..1]		
SaleReferenceID	[0..1]	R	Mandatory if payment reservation or if CustomerOrder is present
POIData	[1..1]		
POITransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
POIReconciliationID	[0..1]		If Result is Success or Partial
PaymentResult	[0..1]		If one data element is present
PaymentType	[0..1]		Copy, default Normal
PaymentInstrumentData	[0..1]		If a payment instrument is analysed by the POI.
PaymentInstrumentType	[1..1]		
CardData	[0..1]		If PaymentInstrumentType is "Card"
PaymentBrand	[0..1]		If card PAN is readable
MaskedPAN	[0..1]		If required for the card, instead of clear PAN
PaymentAccountRef	[0..1]		Mandatory if available.
EntryMode	[1..1]		Copy if present in the request
CardCountryCode	[0..1]		If available in the card
ProtectedCardData	[0..1]		SensitiveCardData protected by CMS EnvelopedData
SensitiveCardData	[0..1]		If structure non empty (unprotected)
PAN	[0..1]		If card PAN is readable
CardSeqNumb	[0..1]		If data available on the card
ExpiryDate	[0..1]		If data available on the card
TrackData	[0..4]		if configured to be sent, and EntryMode is MagStripe or RFID
TrackNumb	[0..1]		default 2
TrackFormat	[0..1]		default ISO
TrackValue	[1..1]		

<i>PaymentResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
IMEI	[0..1]		If data available
StoredValueAccountID	[0..1]		If PaymentInstrumentType is "StoredValue"
StoredValueAccountType	[1..1]		
StoredValueProvider	[0..1]		If available for the card or account.
OwnerName	[0..1]		If available for the card or account.
ExpiryDate	[0..1]		If required for the card or account.
EntryMode	[1..1]		Copy if present in the request.
IdentificationType	[1..1]		
StoredValueID	[1..1]		
AmountsResp	[0..1]		If Result is Success or Partial
Currency	[0..1]		Mandatory for currency conversion.
AuthorizedAmount	[1..1]		
TotalRebatesAmount	[0..1]	L	If rebate on the total amount or rebate on individual products
TotalFeesAmount	[0..1]	S	If fees to be charged from a financial service
CashBackAmount	[0..1]	S	If cashback service was performed with the payment
TipAmount	[0..1]		If payment with tip requested by the Sale System.
Instalment	[0..1]		Absent if PaymentType is not "IssuerInstalment"
InstalmentType	[0..1]		Copy
SequenceNumber	[0..1]		
PlanID	[0..1]		
Period	[0..1]		
PeriodUnit	[0..1]		
FirstPaymentDate	[0..1]		Mandatory if InstalmentType= "DeferredInstalments".
TotalNbOfPayments	[0..1]		
CumulativeAmount	[0..1]		
FirstAmount	[0..1]		If the first amount is different from the others (InstalmentType= "InequalInstalments")
Charges	[0..1]		
CurrencyConversion	[0..n]		
CustomerApprovedFlag	[0..1]		Default True
ConvertedAmount	[1..1]		
AmountValue	[1..1]		
Currency	[1..1]		
Rate	[0..1]		Conversion rate of the target currency against the source currency.
Markup	[0..1]		Markup of the conversion.
Commission	[0..1]		Commission of the conversion.
Declaration	[0..1]		A declaration has to be presented to the customer.
MerchantOverrideFlag	[0..1]		default "False" If payment forced by the Cashier.
CapturedSignature	[0..1]		If handwritten signature is captured on the POI by a signature capture device.
AreaSize	[0..1]		
SignaturePoint	[1..n]		
X	[1..1]		

<i>PaymentResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
Y	[1..1]		
ProtectedSignature	[0..1]		Encrypted handwritten signature captured on the POI by a signature capture device.
CustomerLanguage	[0..1]		If the customer language is different from the default language or different from the CustomerLanguage of the PaymentRequest if any.
OnlineFlag	[0..1]		default "True" "True" if the payment transaction processing has required the approval of a host.
AuthenticationMethod	[0..1]		Methods for customer authentication.
ValidityDate	[0..1]	OR	If OneTimeReservation, FirstReservation or UpdateReservation and Result is Success
PaymentAcquirerData	[0..1]		If card is analysed and data available
AcquirerID	[0..1]		If several Acquirers
MerchantID	[1..1]		
AcquirerPOIID	[1..1]		
AcquirerTransactionID	[0..1]		If provided by the Acquirer and different from the POITransactionID.
TransactionID	[1..1]		
TimeStamp	[1..1]		
ApprovalCode	[0..1]		If available
HostReconciliationID	[0..1]		If provided by the Acquirer
LoyaltyResult	[0..n]	L	Loyalty cards used with the payment transaction. First the loyalty account present in the request in the same order if any.
LoyaltyAccount	[1..1]		
LoyaltyAccountID	[1..1]		
EntryMode	[1..1]		
IdentificationType	[1..1]		
IdentificationSupport	[1..1]		
LoyaltyID	[1..1]		
LoyaltyBrand	[0..1]		<i>see Loyalty response</i>
CurrentBalance	[0..1]		<i>see Loyalty response</i>
LoyaltyAmount	[0..1]		<i>see Loyalty response</i>
LoyaltyUnit	[0..1]		<i>see Loyalty response</i>
Currency	[0..1]		<i>see Loyalty response</i>
AmountValue	[1..1]		
LoyaltyAcquirerData	[0..1]		<i>see Loyalty response</i>
LoyaltyAcquirerID	[0..1]		<i>see Loyalty response</i>
ApprovalCode	[0..1]		<i>see Loyalty response</i>
LoyaltyTransactionID	[0..1]		<i>see Loyalty response</i>
TransactionID	[1..1]		
TimeStamp	[1..1]		
HostReconciliationID	[0..1]		<i>see Loyalty response</i>
Rebates	[0..1]		<i>see Loyalty response</i>
TotalRebate	[0..1]		<i>see Loyalty response</i>
RebateLabel	[0..1]		<i>see Loyalty response</i>
SaleItemRebate	[0..n]		<i>see Loyalty response</i>

<i>PaymentResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
ItemID	[1..1]		
ProductCode	[1..1]		
EanUpc	[0..1]		if present in the related SaleItem
UnitOfMeasure	[0..1]		see <i>Loyalty response</i>
Quantity	[0..1]		see <i>Loyalty response</i>
ItemAmount	[0..1]		see <i>Loyalty response</i>
RebateLabel	[0..1]		see <i>Loyalty response</i>
PaymentReceipt	[0..*]		If Basic profile implementation with no printer on the POI.
DocumentQualifier	[1..1]		SaleReceipt or CashierReceipt
IntegratedPrintFlag	[0..1]		same as <i>PrintRequest</i>
RequiredSignatureFlag	[0..1]		default False.
OutputContent	[1..1]		
OutputFormat	[1..1]		Text, XHTML
OutputText	[0..n]		same as <i>Display</i>
Text	[1..1]		same as <i>Display</i>
CharacterSet	[0..1]		same as <i>Display</i>
Font	[0..1]		same as <i>Display</i>
StartColumn	[0..1]		same as <i>Display</i>
Color	[0..1]		same as <i>Display</i>
CharacterWidth	[0..1]		same as <i>Display</i>
CharacterHeight	[0..1]		same as <i>Display</i>
CharacterStyle	[0..1]		same as <i>Display</i>
Alignment	[0..1]		same as <i>Display</i>
EndOfLineFlag	[0..1]		same as <i>Display</i>
OutputXHTML	[0..1]		same as <i>Display</i>
CustomerOrder	[0..n]		If the payment is related to a customer order in progress or if the list of customer orders has been requested.
CustomerOrderID	[0..1]		
SaleReferenceId	[1..1]		
OpenOrderState	[0..1]		default "True"
StartDate	[1..1]		
EndDate	[0..1]		If OpenOrderState = "False".
ForecastedAmount	[1..1]		
CurrentAmount	[1..1]		
Currency	[0..1]		If multiple currencies are allowed.
AccessedBy	[0..1]		If order process in progress.
AdditionalInformation	[0..1]		

4.3.1.5 Standard Payment Transaction

Payment is performed by the POI to pay the purchase of services or goods made by the customer using a payment card. This section presents the processing of the standard Payment message pair without using a CardAcquisition message to separate the payment mean selection.

The Sale system, at the end of the purchase, requests to the POI the payment of the purchase with the Payment request message described below:

Sale Terminal Information: SaleData

In addition to the Sale transaction identifier *SaleTransactionID* (see 4.3.1.1 *Transaction Identification*), the data structure *SaleData* contains the context of the Sale Terminal which could be factorised in the Login Request message, to avoid sending it at every transaction.

The value of the data elements *OperatorID*, *ShiftNumber*, and *TotalsGroupID* could be used to compute related transactions totals in the Reconciliation or GetTotals messages. The *TotalsGroupID* could be used by the Sale System to group transactions for its own purpose.

AmountsReq

Currency is a mandatory data element of the data structure *PaymentTransaction.AmountsReq*. The currency is the same for all amounts of the transaction.

Standard payment uses only the *RequestedAmount*. A Payment request with a *RequestedAmount* set to 0 is forbidden. The Payment response message is sent back with *Response* = “Failure” - “NotAllowed” (see section 4.6.4.1.4 *NotAllowed* Value).

Transaction Conditions

Conditions of the transaction required by the Sale System are:

- *AllowedPaymentBrand*, if present contains the set of payments brand that may be used by the Customer. It could be the result of a declaration by the Customer with a dedicated key pressed by the Cashier, or any restriction imposed by the Merchant for this transaction.
- *AcquirerID*, if present contains the set of Acquirers which are preferred for this transaction. The POI must use one of these Acquirers to realise the payment transaction acquisition.
- *AllowedLoyaltyBrand*, the same requirement, but for the loyalty cards.
- *LoyaltyHandling*, which has to be present with the value “Forbidden” or “Processed”, for a standard payment without loyalty.
- *CustomerLanguage*, if the Customer language is selected by the Sale System before the payment, for instance on a vending machine and unattended context (see *SaleTerminalData.TerminalEnvironment*)
- *ForceOnlineFlag*, to request to go online for this payment whatever the rules of the payment brand have decided.
- *MerchantCategoryCode*, if the Sale Terminal has several MCC related to the type of services or goods it provides.
- *ForceEntryMode*, to restrict the card entry mode for some reasons.

Information to Carry

Data element *SaleToPOIData*, *SaleToAcquirerData*, and *SaleToIssuerData*, which has to be stored on the POI Log, to be send to the Acquirer or to the Issuer without any interpretation of the POI System.

Sold Items

The sold items *SaleItem*, are sent by the Sale Terminal if a card with product restrictions could be used by the Customer for the payment transaction. They are used by the POI during the payment processing only on this case.

The *ItemID* is mandatory to be able to identify the sold item. Other mandatory data elements are *ProductCode*, which identify the product or the category of product, and *ItemAmount*, which is the amount of the item in the currency of the data structure *AmountsReq*.

Other data elements are optional, and sent to the Acquirer if present in the request.

Payment Information: PaymentData

The requested payment transaction is standard if the *PaymentType* of the data structure *PaymentData* has the value “Normal”. As this is the default value, the component *PaymentType* could be absent.

The component *CardData* is present if and only if the card data is read by the Sale Terminal. So if this is not the case, a standard payment do not use the data structure *PaymentData* in the Payment Request message.

Loyalty Information: LoyaltyData

Standard payment do not process loyalty, the data structure *LoyaltyData* is absent of the Payment Request message.

After sending the Payment request message, the POI Terminal verifies the validity of the message, realises the selection of a payment mean accepted by the POI for this transaction, processes the payment transaction with some dialogues with the Customer, and the Cashier if necessary, exchanges some messages with a payment Acquirer or other host if necessary, possibly prints a receipt, and finally answers to the Sale Terminal with a Payment response carrying the outcome of the transaction.

Depending on the POI Terminal profile, the environment (e.g. attended/unattended), the transaction processing, and, if Devices message exchanges have been implemented, the POI Terminal can send *Display Request* (e.g. to inform the Cashier the process status), *Input Request* (e.g. to ask the Cashier some confirmation), *Print Request* (e.g. to print the payment receipt),

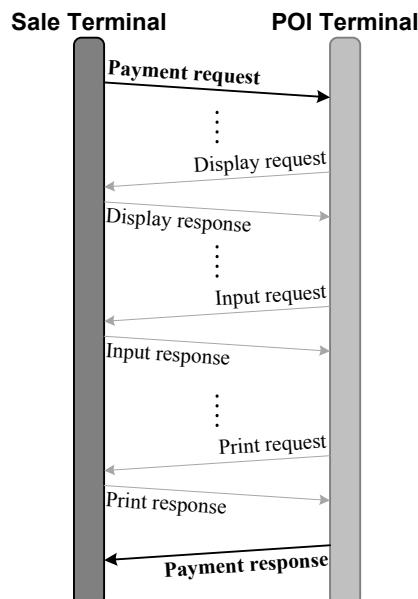


Figure 129: Standard Payment Exchange

With the result of the transaction *Response*, *POITransactionID* and *PaymentType* are the only mandatory component of the response message, whatever the result of the transaction.

Sale Terminal Information: SaleData

A copy of the Sale transaction identifier *SaleTransactionID* sent in the request is mandatory for verification.

POI Terminal Information: POIData

The POI transaction identifier *POITransactionID* (see 4.3.1.1 *Transaction Identification*) is mandatory, whatever the result of the transaction.

The data structure *POIData* contains the identification of the reconciliation period *POIReconciliationID*, if the POI Terminal has a reconciliation profile, and the *Result* of the transaction is not "Failure".

Payment Transaction Outcome: PaymentResult

A copy of the *PaymentType* sent in the request ("Normal" for standard payment), is mandatory²⁰ to recognise quickly the type of payment transaction result.

²⁰ As "Normal" is the default value, *PaymentType* could be absent in request and response for the standard payment.

The *PaymentInstrumentData* data structure contains the information relation to the payment mean (*PaymentInstrumentType*) choosen by the customer: *CardData*, *CheckData*, *MobileData*, *StoredValueAccountID*, or Cash.

Payment Instrument: CardData

The *CardData* data structure has to be present if a card is read by the POI Terminal, or provided by the Sale Terminal. In this case, *EntryMode* is mandatory. If the identifier of the card is present (*PAN*), *PaymentBrand*²¹ is mandatory (could be optional and configured to be sent or not), *MaskedPAN* is present when *SensitiveCardData* is protected and replaced by *ProtectedCardData*, *CardCountryCode*, *CardSeqNumb*, and *ExpiryDate* are present if available for the card, and *TrackData* if required to be delivered to the Sale for this type of card.

SensitiveCardData, which contains sensitive card data, could be protected. Then the complete *SensitiveCardData* data structure is encrypted with a cryptographic key, and is replaced by *ProtectedCardData*, using the CMS data structure *ContentInformationType* (EnvelopedData). Purpose to send sensitive card data in the response are:

- To print or display information about the payment transaction.
- To process some service related to the payment card. Some examples of this requirement are: find a loyalty account attached to the payment card, the loyalty transaction being realised by the Sale System; process some batch with an Acquirer or an Intermediary Agent.

Payment Instrument: CheckData

The *CheckData* data structure contains:

- The identification of the paper check which could read on the magnetic track (*TrackData*), or provided by another method (*BankID*, *AccountNumber*, *CheckNumber*),
- The type of paper check,
- Additional information specific to that payment instrument.

Payment Instrument: MobileData

The *MobileData* data structure contains information related to the mobile for the payment transaction, when a mobile phone is used as a payment instrument for the transaction.

Identification of the mobile and its subscription, *SensitiveMobileData*, could be protected in the data structure *ProtectedMobileData* which then replaces *SensitiveMobileData*.

Other information as *Geolocation* are provided if available.

Payment Instrument: StoredValueAccountID

The *StoredValueAccountID* data structure contains the identifications of the stored value account or the stored value card.

Amounts related to the Payment Response: AmountsResp

AuthorizedAmount represents the amount authorised which is the paid amount for a standard payment. *TotalFeesAmount* is an optional paid amount for the service provided with the card for the transaction.

AuthorizedAmount = *RequestedAmount* + *TotalFeesAmount*, if the complete requested amount is authorised by the card for the payment.

If the authorisation is partial: *AuthorizedAmount* < *RequestedAmount* + *TotalFeesAmount*. In this case the data element *Result* has the value "Partial"

Dynamic Currency Conversion: CurrencyConversion

²¹ This is a card brand name at the interface level.

CurrencyConversion contains information related to a dynamic currency conversion performed during the payment transaction.

When the payment is accepted because the Merchant has forced the result (for instance with an *Input* messages and a “SiteManager” *InputCommand*), the *PaymentResult* contains the data element *MerchantOverrideFlag* with the value “True”.

Handwritten Signature Capture: CapturedSignature

If the handled signature of the payment receipt is captured by the POI, the captured signature *CapturedSignature* can be included in the *PaymentResult* (or the encrypted signature *ProtectedSignature*).

See the Error Case “Payment Restriction Error” in the next section for the condition of presence of *AllowedProductCode* in the Payment response.

Payment Acquirer Informations: PaymentAcquirerData

If a card is read and the identifier of the card is present (*PAN*), *PaymentAcquirerData* is mandatory, containing at least *MerchantID* and *AcquirerPOIID*, the identifier of the POI for the Acquirer. *AcquirerID*, *AcquirerTransactionID*, *ApprovalCode*, and *HostReconciliationID* are present if available.

Payment Receipt to be printed: PaymentReceipt

PaymentReceipt contains the merchant and the customer payment receipts.

This data structure is present (and mandatory) when the Sale or POI system has not implemented the Print message exchange and the printer of the Sale terminal is used to print the merchant and customer payment receipts.

Loyalty Transactions Outcome: LoyaltyResult

Standard payment do not process loyalty, the data structure *LoyaltyResult* is absent of the Payment Request message.

4.3.1.6 Error Cases

When the Payment request is successfully processed, the Payment response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved or before to try another Payment attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The Payment request is received during a Device dialogue or another Service dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

Cancel

The user has aborted the transaction on the Customer interface (e.g. the POI Terminal keyboard), because he does not want to continue the payment (e.g., problem of PIN remembering, chooses another payment mean, stop the purchase on a vending machine), see section 4.6.6.2.1 *User Cancellation*.

Abort

The Sale System sent an Abort request message before the end of the Payment request processing. The POI has aborted the payment transaction, and sends the Payment response message with this *ErrorCondition*, to report the result of the aborted payment (see section 4.6.6.1 *Aborted Error*).

DeviceOut

The POI element cannot start the payment transaction, because of a temporary error on a device (e.g. printer without paper), see section 4.6.3.1.1 *POI Temporary Unavailable*.

The POI element cannot start payment transaction, because of a permanent error on a device (e.g. card reader out of order, pin-pad disconnected). This error is returned just after the problem occurs on the POI Terminal, during the payment processing or because the Sale has not taken into account the reception of the Event Notification reporting the occurrence of the problem (see section 4.6.3.1.2 *POI Permanently Unavailable*).

InvalidCard

The POI terminates the transaction because no card is entered by the Customer, or the inserted card is not configured in the system and cannot be processed (see section 4.6.7.1.1 *No Card Entered* and section 4.6.7.1 *InvalidCard Error*).

Wrong PIN

The Cardholder has entered its PIN on the PED keyboard and the verification fails (see section 4.6.7.2 *WrongPIN Error*).

PaymentRestriction

The Customer has used a card restricted on the products the card may pay. Some of the items provided in the *SaleItem* of the request are product than the card cannot pay. The products of the *SaleItem* sequence which could be paid by the card are reported in the *AllowedProductCode* of the Payment response (see section 4.6.8.3 *PaymentRestriction Error*).

UnreachableHost

The Payment Acquirer is unreachable or has not answered to an online request, so it is considered as temporary unavailable. The Cashier has not forced the transaction, and the payment cannot be accepted (see section 4.6.8.1.1 *Host Unreachable* and section 4.6.8.1.2 *No Host Answer*).

Refusal

The transaction is refused by the payment Acquirer or the rules associated to the card. The Cashier has not forced the transaction, and the payment cannot be repeated. A specific message is normally displayed to the Customer and the Cashier if they are presents, the information below could be logged for further information (see section 4.6.8.2 *Refusal Error*).

4.3.1.7 Examples

a) Simple Payment

This is a simple payment from SaleTermA on POITerm1, for an amount of 104.11 €.

Customer pays with a magstripe card, and the payment is approved for the requested amount.

MessageHeader		(message example 9)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	642	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	579	
TimeStamp	2009-03-10T23:08:42.4+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	104.11	
TransactionConditions		
LoyaltyHandling	Forbidden	
PaymentData		
PaymentType	Normal	

MessageHeader		(message example 10)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	642	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentResponse		
Response		
Result	Success	
SaleData		
SaleTransactionID		
TransactionID	579	
TimeStamp	2009-03-10T23:08:42.4+01:00	
POIData		
POITransactionID		
TransactionID	481	
TimeStamp	2009-03-10T23:08:42.4+01:00	
POIReconciliationID	200903101	
PaymentResult		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	CardPlus	
EntryMode	MagStripe	
SensitiveCardData		
PAN	0011014570541535	

ExpiryDate	0411
AmountsResp	
AuthorizedAmount	104.11
PaymentAcquirerData	
AcquirerID	400012
MerchantID	mer77-130209
AcquirerPOIID	963276433
ApprovalCode	8347

b) Protected Card Data and MAC

The request and response are authenticated by a MAC.

The Sale System restricts the card brands for the payment. Customer pays with a smartcard, the payment is approved for the requested amount.

MessageHeader		(message example 11)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	642	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	580	
TimeStamp	2010-06-10T22:53:12.6+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	31.00	
TransactionConditions		
LoyaltyHandling	Forbidden	
SecurityTrailer		
ContentType	id-ct-authData	
AuthenticatedData		
Version	v0	
KEK		
Version	v4	
KEKIdentifier		
KeyIdentifier	SpecV1TestMACKey	
KeyVersion	2010060715	
KeyEncryptionAlgorithm		
Algorithm	des-ede3-cbc	
EncryptedKey	9CF4E2DE12A260E45D082D5DCCE4D050	
MACAlgorithm		
Algorithm	id-retail-cbc-mac-sha-256	
EncapsulatedContent		
ContentType	id-data	
MAC	F4411AE44D2A717B	

The card data in the response, PAN 4978678252755678 and expiry date 04/11, are protected by encryption. The result of the encryption is computed with an XML data coded *SensitiveData* which is:

```
"<SensitiveCardData PAN="4978678252755678" ExpiryDate="0411"/>"
```

		<i>(message example 12)</i>
MessageHeader		
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	642	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentResponse		
Response		
Result	Success	
SaleData		
SaleTransactionID		
TransactionID	580	
TimeStamp	2010-06-10T22:53:12.6+01:00	
POIData		
POITransactionID		
TransactionID	482	
TimeStamp	2010-06-10T22:53:12.6+01:00	
POIReconciliationID	200903101	
PaymentResult		
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	VISA	
EntryMode	MagStripe	
ProtectedCardData		
ContentType	id-envelopeData	
EnvelopedData		
Version	v0	
KEK		
Version	v4	
KEKIdentifier		
KeyIdentifier	SpecV1TestDATKey	
KeyVersion	2010060715	
KeyEncryptionAlgorithm		
Algorithm	des-ede3-cbc	
EncryptedKey	9CF4E2DE12A260E45D082D5DCCE4D050	
EncryptedContent		
ContentType	id-data	
ContentEncryptionAlgorithm		
Algorithm	des-ede3-cbc	
Parameter		
InitialisationVector	A27BB46D1C306E09	
EncryptedData	FECC2C960F5525D82A82B9E641934545A68A556B01E05949C 888F752DC4296AEA377044E0E926F40842A84EEF5D6DC271C 28D7F7EDDB40CD2DCAF38D3E0238AA5811C879EF92BF32	
AmountsResp		
AuthorizedAmount	31.00	
PaymentAcquirerData		
AcquirerID	497867	
MerchantID	mer77-130209	
AcquirerPOIID	456	

ApprovalCode	9473
SecurityTrailer	
ContentType	id-ct-authData
AuthenticatedData	
Version	v0
KEK	
Version	v4
KEKIdentifier	
KeyIdentifier	SpecV1TestMACKey
KeyVersion	2010060715
KeyEncryptionAlgorithm	
Algorithm	des-ede3-cbc
EncryptedKey	9CF4E2DE12A260E45D082D5DCCE4D050
MACAlgorithm	
Algorithm	id-retail-cbc-mac-sha-256
EncapsulatedContent	
ContentType	id-data
MAC	C998B351E39FE2D0

c) Payment with Products

The Sale System sends the sold item products to be paid. The basket contains 2 bottles of mineral water and 38.23 litres of Diesel. The Customer pays with a fleet card.

MessageHeader		(message example 13)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	643	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	581	
TimeStamp	2015-04-06T10:42:34.1+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	38.52	
TransactionConditions		
LoyaltyHandling	Forbidden	
SaleItem		
ItemID	1	
ProductCode	673	
EanUpc	84116369	
Quantity	2	
UnitPrice	1.15	
ItemAmount	2.30	
ProductLabel	Mineral Water 1,5L	
SaleItem		
ItemID	2	
ProductCode	101	
UnitOfMeasure	Litre	
Quantity	38.23	
UnitPrice	0.869	
ItemAmount	33.22	
ProductLabel	Diesel Fuel	
PaymentData		
PaymentType	Normal	

MessageHeader		(message example 14)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	643	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentResponse		
Response		
Result	Success	
SaleData		
SaleTransactionID		
TransactionID	581	
TimeStamp	2015-04-06T10:42:34.1+01:00	
POIData		
POITransactionID		
TransactionID	479	
TimeStamp	2015-04-06T10:43:02.9+01:00	
POIReconciliationID	200903101	
PaymentResult		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	FleetUniverse	
EntryMode	MagStripe	
SensitiveCardData		
PAN	6900014570541535	
ExpiryDate	0411	
AmountsResp		
AuthorizedAmount	38.52	
PaymentAcquirerData		
AcquirerID	690001	
MerchantID	mer77-130209	
AcquirerPOIID	963276433	
ApprovalCode	6541	

d) Payment with a Local Card

The Sale Terminal reads the card with a bar-code and sends card data to the POI Terminal. For this card, additional fees are paid, and the payment is partial.

MessageHeader		(message example 15)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	644	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	581	
TimeStamp	2009-03-11T10:37:21.9+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	38.01	
TransactionConditions		
LoyaltyHandling	Forbidden	
PaymentData		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
EntryMode	Scanned	
SensitiveCardData		
PAN	7011014570541535	

MessageHeader		(message example 16)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	644	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentResponse		
Response		
Result	Partial	
SaleData		
SaleTransactionID		
TransactionID	581	
TimeStamp	2009-03-11T10:37:21.9+01:00	
POIData		
POITransactionID		
TransactionID	481	
TimeStamp	2009-03-11T10:42:52.4+01:00	
POIReconciliationID	200903101	
PaymentResult		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	MerRel	

EntryMode	MagStripe
SensitiveCardData	
PAN	7011014570541535
AmountsResp	
AuthorizedAmount	31.12
TotalFeesAmount	0.19
PaymentAcquirerData	
MerchantID	mer77-130209
AcquirerPOIID	65-POITerm1

4.3.2 Other Payment Services

4.3.2.1 Cash Back

Cashback is an additional service provided by the Merchant to the Customer, to obtain a small amount of cash which is added to the purchase amount paid by card.

There are three data elements in the protocol devoted to the cashback:

- *CashBackAmount*: the requested amount of cashback in *PaymentRequest.PaymentTransaction.AmountsReq*,
- *MaximumCashBackAmount*: the maximum amount of cash back that the Merchant allows for this payment in *PaymentRequest.PaymentTransaction.AmountsReq*,
- *CashBackAmount*: the paid amount of cash back in *PaymentResponse.PaymentResult.AmountsResp*.

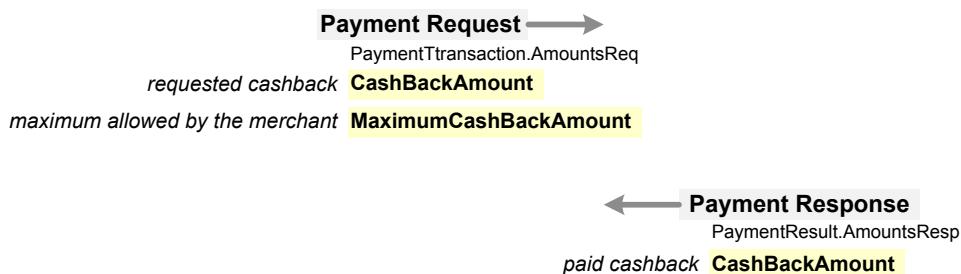


Figure 130: Cash Back Data Elements

Rule 1: When the *CashBackAmount* component of the Payment request is absent, the cashback amount could be proposed to the Customer by the POI during the payment transaction, regarding:

- The configuration parameter for this service with this type of card,
- The rules attached to the payment card,
- The *MaximumCashBackAmount* allowed by the Merchant configured or sent in the Payment request, and
- The decision of the Customer.

Configuration attached to the cashback function in the POI has to be consistent with the expected cash in the cash register.

Rule 2: When the *CashBackAmount* component of the Payment request is present, the cashback amount is negotiated between the Cashier and the Customer, and sent in the Payment Request message. Depending on the cash-back rules of the card chosen by the Customer and the response of the Acquirer, the requested cashback amount could be reduced in the Payment Response. If the payment is authorised, but the cash-back is refused, the *Response* component of the Payment response has the value "Partial" and the *CashBackAmount* component must have the value "0" in the Payment Response.

Rule 3: The *CashBackAmount* component is present in the Payment response if cashback has been performed with the payment. The value of *CashBackAmount* in the response must be lower or equal to the value of *CashBackAmount* and *MaximumCashBackAmount* in the request.

Rule 4: Cashback is allowed only for Payment request with a “Normal” *PaymentType*. For other *PaymentType* values, the *Response* component of the Payment response has the value “Failure”-“UnavailableService” (see section 4.6.4.3.2 *Unavailable Service for the Card*).

4.3.2.2 Split Payment

Payment of a Sale transaction is called “split payment”, if the related sale transaction requires several payment transactions.

The interface between the Sale system and the POI system for split payments is based on the following principles:

- The POI system processes a single payment per request of the Sale system.
- The Sale Terminal has the responsibility to manage the split of the payment for the Sale transaction.
- The split of the payment could also be noticed at the POI system when a payment is partial, or after some negotiations between the POI and the Customer.
- Split payment could be refused by some card brands.

The following data elements are used in the protocol to manage split payments:

- *SplitPaymentFlag*: this indicator in *PaymentRequest.PaymentData*, “False” by default, is set to “True” if the payment transaction is a split payment.
- *MinimumSplitAmount*: the minimum amount of a split payment transaction in *PaymentRequest.PaymentTransaction.AmountsReq*,
- *PaidAmount*: the amount paid by the previous split payment transactions in *PaymentRequest.PaymentTransaction.AmountsReq*,
- The various amounts in *PaymentResponse.PaymentResult.AmountsResp*, which could result in a partial payment of the requested amount.

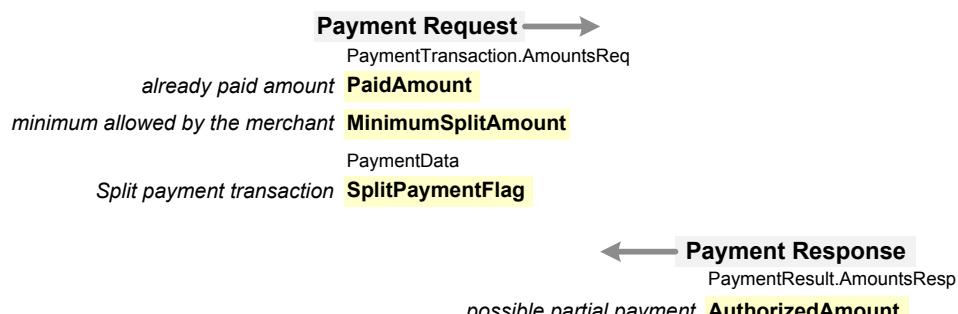


Figure 131: Split Payment Data Elements

Rule 1: When the *SplitPaymentFlag* component of the Payment request is present, the payment transaction is a split payment. *RequestedAmount* represents the amount to pay for this split transaction, *PaidAmount* is mandatory and represents the paid amount for the previous split transactions.

Most of the time, the amount of the Sale transaction is the sum of *RequestedAmount* and *PaidAmount*.

Rule 2: When the *SplitPaymentFlag* component of the Payment request is absent, the payment transaction is a split payment if:

$$\begin{aligned}
 & \text{AuthorizedAmount} + \text{TotalRebatesAmount} - (\text{TotalFeesAmount} + \\
 & \text{PaymentResponse.CashBackAmount} + \text{PaymentResponse.TipAmount}) \\
 < \\
 & \text{RequestedAmount} - (\text{PaymentRequest.CashBackAmount} + \text{PaymentRequest.TipAmount})
 \end{aligned}$$

- Rule 3: When the *MinimumSplitAmount* component of the Payment request is present, the amount *AuthorizedAmount* of the payment transaction has to be greater or equal than *MinimumSplitAmount* and lower than *RequestedAmount*.
- Rule 4: When a split payment is requested by the Sale Terminal or the Customer to the POI Terminal, and the Customer's card does not accept split transaction, either the POI proposes to select another card, either the POI Terminal answers with the value "Failure"- "UnavailableService" in the *Response* component of the Payment response (see section 4.6.4.3.2 *Unavailable Service for the Card*).
- Rule 5: Split is allowed only if the component *PaymentType* of the Payment request is set to "Normal". For other *PaymentType* values, the *Response* component of the Payment response has the value "Failure"- "NotAllowed" (see section 4.6.4.1.2 *Forbidden Combination of Service*).
- Rule 6: When the payment of the Sale transaction is split, the Sale Terminal is in charge of building an appropriate *SaleItem* list, which could be a fragment of the Sale transaction list (split basket), or the complete list (split amount only). The sum of the item amounts could then be more than the *RequestedAmount*.
- Rule 7: In any case, when the requested amount is not authorised, the *Response* component of the Payment response must have the value "Partial".
- Rule 8: The Sale identification *SaleTransactionID* of all the split payment transactions shall have the same value.

An example of typical split payments transactions is presented below, where the Customer that pays choose to limit the amount per card:

1. The Sale Terminal requests the payment of the Sale transaction for an amount of 500.12 €. The Customer empties a prepaid card with an amount of 35.65 €.
2. The Sale Terminal accept the payment split, and requests the payment of the remaining amount of 464.47 €, including the *SplitPaymentFlag* flag and the already paid amount of 35.65 €. The Customer uses another card with an amount of 300 €.
3. The Sale Terminal requests the payment of the remaining amount of 164.47 €, including the *SplitPaymentFlag* and the already paid amount of 335.65 €. The Customer uses a third card to complete the payment of the Sale transaction with the remaining amount of 164.47 €.

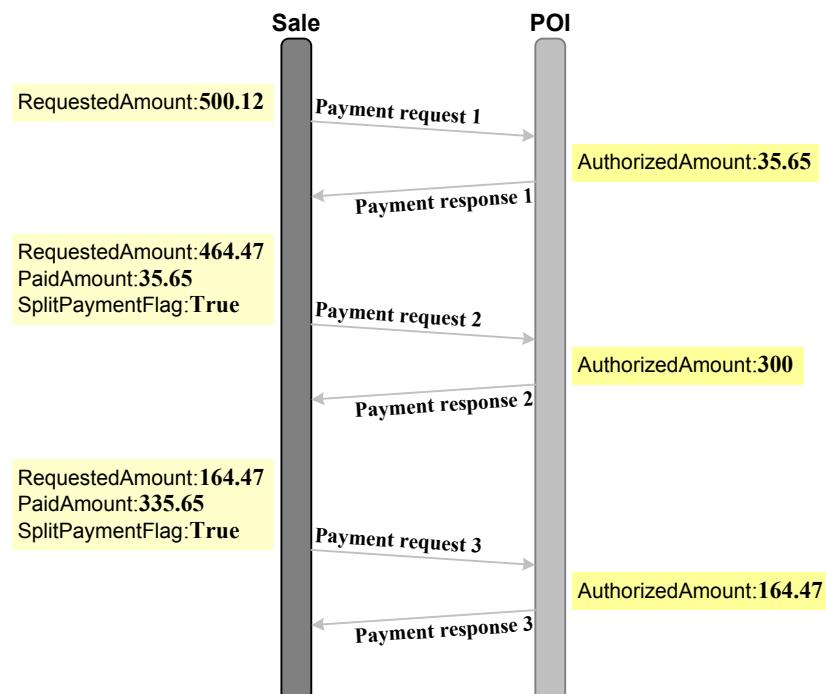


Figure 132: Typical Sequence of Split Payments

4.3.2.3 Deferred Sale

Deferred sale payment is used to separate the payment in two steps:

- Before the service consumption or the goods delivery, to get the payment authorisation for a maximum amount.
- At the end of the consumption or delivery, to complete the payment transaction with the actual amount.

The first Payment message pair uses the following data elements:

- *RequestedAmount*: this data could be absent or is the maximum amount of the purchase.
- *MinimumAmountToDeliver*: the minimum amount of the purchase if the *RequestedAmount* is absent,
- *PaymentType*: must be “OneTimeReservation”,
- *AuthorizedAmount*: the maximum amount of the purchase.

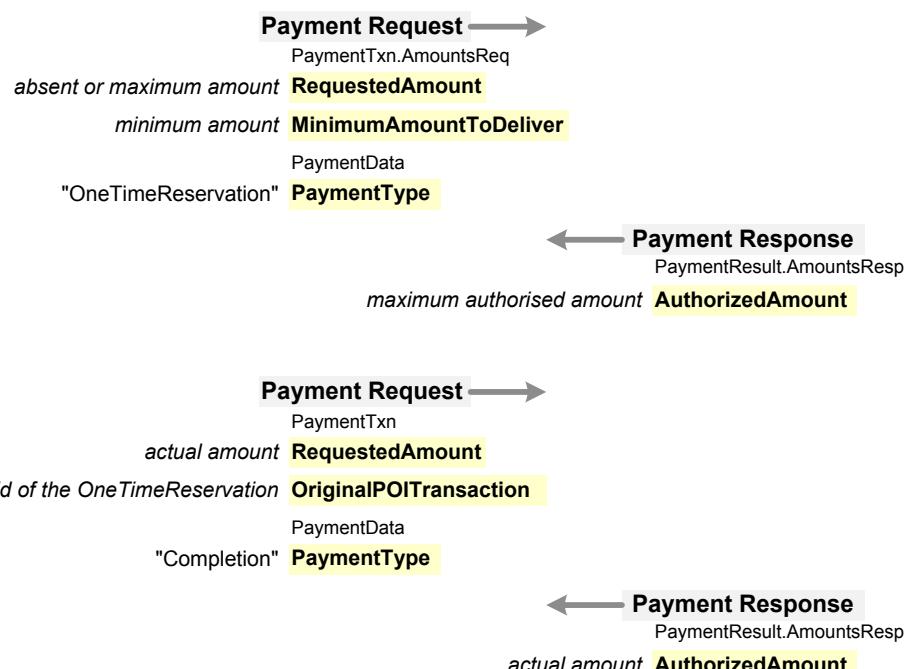


Figure 133: Deferred Sale Data Elements

The second Payment message pair uses the following data elements:

- *RequestedAmount*: this is the actual amount of the purchase.
- *PaymentType*: must be “Completion”,
- *OriginalPOITransaction*: this is a mandatory component of the Payment request containing the *POITransactionID* component of the first Payment response with “OneTimeReservation” *PaymentType*,
- *AuthorizedAmount*: the actual amount of the purchase, as in *RequestedAmount*.

Rule 1: The deferred sale transaction shall include a first Payment request with the *PaymentType* component set to “OneTimeReservation”, and after a successful Payment response and the end of the Sale transaction, a second Payment request the *PaymentType* component set to “Completion” and including in *OriginalPOITransaction* the *POITransactionID* of the first Payment request, and the *POIID* component, if the “OneTimeReservation” was made on another POI Terminal.

The *POITransactionID* of these two Payment responses must have the same value (see rule 9 if the two values are inconsistent).

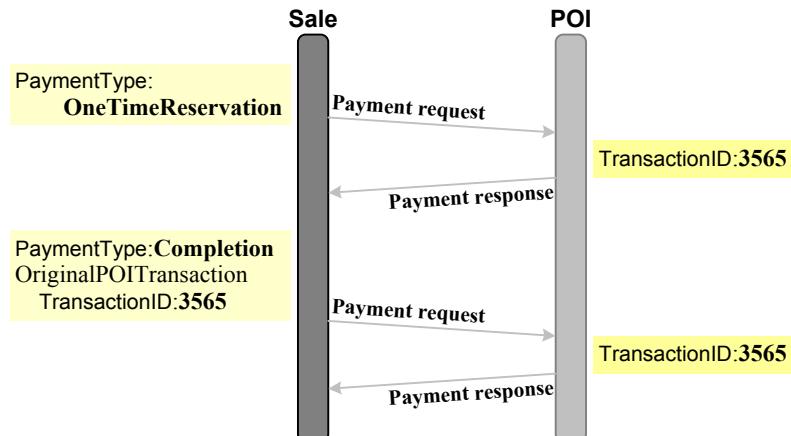


Figure 134: Successful Deferred Sale Sequence Flow

Rule 2: If the “OneTimeReservation” Payment response is unsuccessful (i.e. the *Response.Result* value “Failure”), the deferred transaction is complete, and a “Completion” Payment request is forbidden. The *Response* component of such “Completion” Payment response has the value “Failure” - “NotAllowed” (Invalid *OriginalPOITransaction* Value, see section 4.6.4.1.4 *NotAllowed* Value).

Rule 3: If “OneTimeReservation” Payment is successful, and the Sale transaction fails, the “Completion” Payment message pair must be replaced by a Reversal request to cancel the “OneTimeReservation” Payment.
The *POITransactionID* of the Reversal response must have a different value than the *POITransactionID* of the Payment responses.

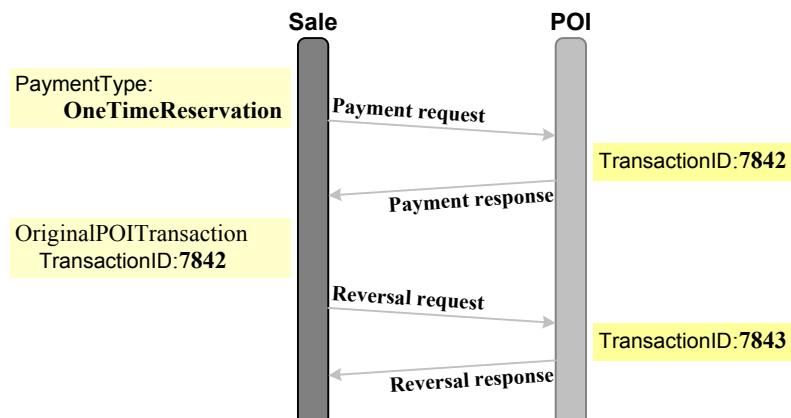


Figure 135: Unsuccessful Deferred Sale Sequence Flow

The “Completion” Payment request with a *RequestedAmount* set to 0 is forbidden. The *Response* component of such “Completion” Payment response has the value “Failure” - “NotAllowed” (see section 4.6.4.1.4 *NotAllowed* Value).

Rule 4: The *RequestedAmount* component of the “OneTimeReservation” Payment request is optional.

The *RequestedAmount* is absent, if the POI System configuration or an Intermediary Agent fixes the maximum amount, depending on the payment card and the context of the transaction.

Rule 5: If the *MinimumAmountToDeliver* component of the “OneTimeReservation” Payment request is present, the component *AuthorizedAmount* of the Payment response must be greater or equal.

Rule 6: The “Completion” Payment request must have the component *OriginalPOITransaction*. Card data of the “OneTimeReservation” Payment response must be used without reading of the card whatever the value of *ReuseCardDataFlag* set in the “Completion” Payment request.

Rule 7: The *RequestedAmount* component of the “Completion” Payment request must have a value lower or equal than the value of the *AuthorizedAmount* component of the “OneTimeReservation” Payment response. Otherwise, the *Response* component of the Payment response must have the value “Failure”-“NotAllowed” (see section 4.6.4.1.4 *NotAllowed Value*).

Rule 8: The following components of the “Completion” Payment request must have the same value as the “OneTimeReservation” Payment request: *SaleTransactionID*, *MerchantCategoryCode*, *SaleItem*, *LoyaltyAccountID*; but the *CardData* component must be absent. Otherwise, the *Response* component of the Payment response must have the value “Failure”-“NotAllowed” (see section 4.6.4.1.4 *NotAllowed Value*).

Rule 9: If the POI does not find the deferred payment transaction identified by the *OriginalPOITransaction* component of the “Completion” Payment request, or if the other elements of the transaction does not match (e.g. the transaction state shows that the completion is already performed), the *Response* component of the Payment response must have the value “Failure”-“NotFound” (see section 4.6.4.4.1 *Transaction Not Found*).

Rule 10: Depending on the environment, the “Completion” Payment request messages could be sent to a POI Terminal while the POI Terminal is processing another Service (e.g. Payment Request). The POI Terminal might accept to perform these two services in parallel, otherwise, the POI must send a Payment response result with the value “Failure”-“Busy” (see section 4.6.5.1.2 *POI Busy*). After this answer, the Sale System must retry to send the “Completion” Payment request message later, but with a limited number of retries.

This rule applies when no user interface is required for the processing of the “Completion” Payment request messages, and the Payment response is an acknowledgement to the Payment request, and not necessary the end of the Completion processing.

Rule 11: Only one “Completion” Payment request messages could be sent for the same Deferred Sale transaction (identified by *POITransactionID*). Otherwise, the POI must send a Payment response result with the value “Failure”-“NotAllowed” (see section 4.6.4.1.5 *Completed Transaction*). The standard error procedure is to send a *TransactionStatus* request to have the status of the pending request (see section 4.7.5.2 *Error Resolutions Specifications on the Sale System*).

4.3.2.4 Reservation

Reservation payment is used to make reservation of an estimated amount, for a certain time preceding the consumption of a service or delivery of goods and the actual payment:

- An initial reservation for an estimated amount of the purchase and a period of time to realise the purchase. This step could be followed by other reservations when the condition of the reservation is changing,
- After the consumption or delivery, to complete the payment transaction with the actual amount.

The Payment message pair uses the following data elements for the initial reservation:

- *SaleReferenceID*, the sale identifier for the whole reservation transaction,
- *RequestedAmount*: this is the estimated amount of the purchase to reserve,
- *PaymentType*: must be “FirstReservation”,
- *RequestedValidityDate*: the estimated time period of the purchase to reserve,
- *ValidityDate*: the validity period of the reservation,
- *AuthorizedAmount*: the reserved amount.

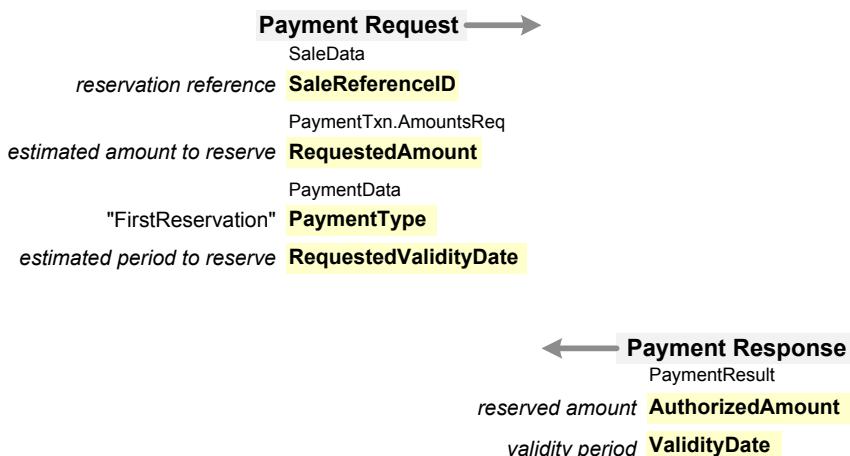


Figure 136: First Reservation Data Elements

The possible Payment message pairs for further reservation update use the following data elements:

- *SaleReferenceID*, the sale identifier for the whole reservation transaction, used for the initial reservation,
- *RequestedAmount*: the new estimated amount of the purchase,
- *PaymentType*: must be "UpdateReservation",
- *RequestedValidityDate*: the new time period of the purchase to reserve,
- *OriginalPOITransaction*: containing the *POITransactionID* component of the Payment response of the last reservation,
- *ValidityDate*: the new validity period of the reservation,
- *AuthorizedAmount*: the new reserved amount.

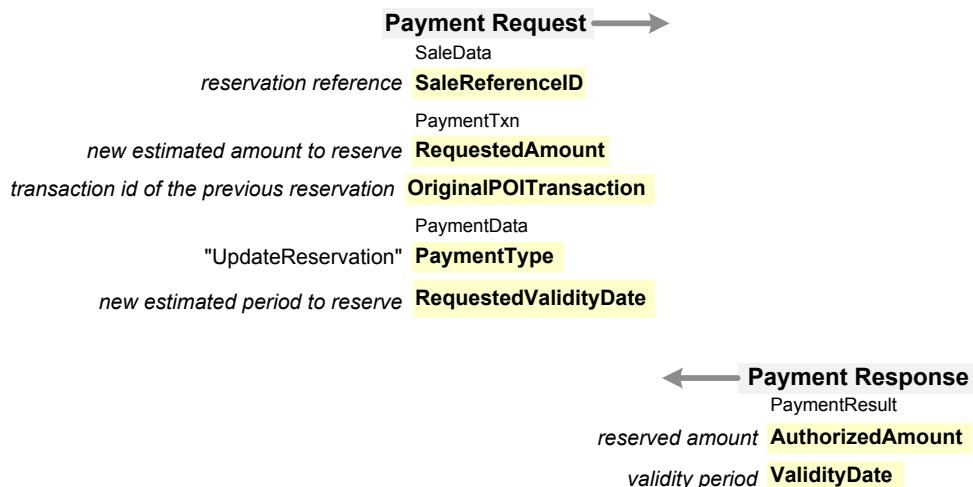


Figure 137: Update Reservation Data Elements

The last Payment message pair to complete the payment after reservation uses the following data elements:

- *SaleReferenceID*, the sale identifier for the whole reservation transaction, used for the initial reservation,
- *RequestedAmount*: the actual amount of the purchase,
- *PaymentType*: must be “Completion”,
- *OriginalPOITransaction*: containing the *POITransactionID* component of the Payment response of the last reservation,
- *AuthorizedAmount*: the actual amount of the purchase, as in *RequestedAmount*.

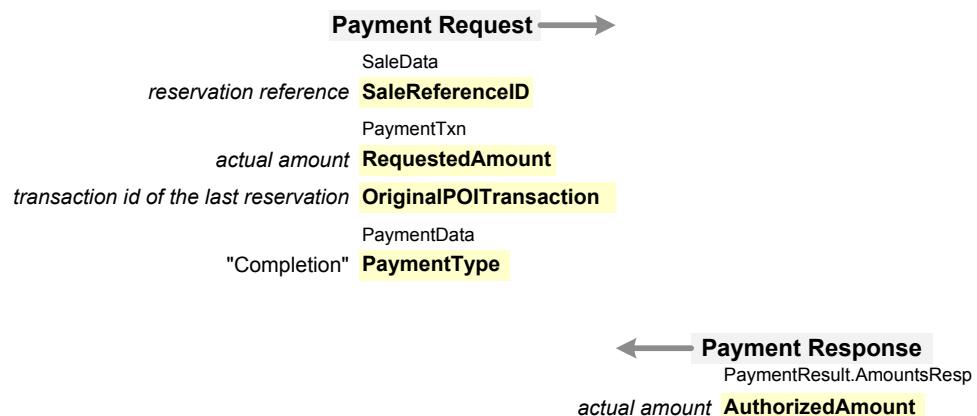


Figure 138: Payment after Reservation Data Elements

The state diagram below presents the possible sequence flows of the Reservation message during the processing of the transaction.

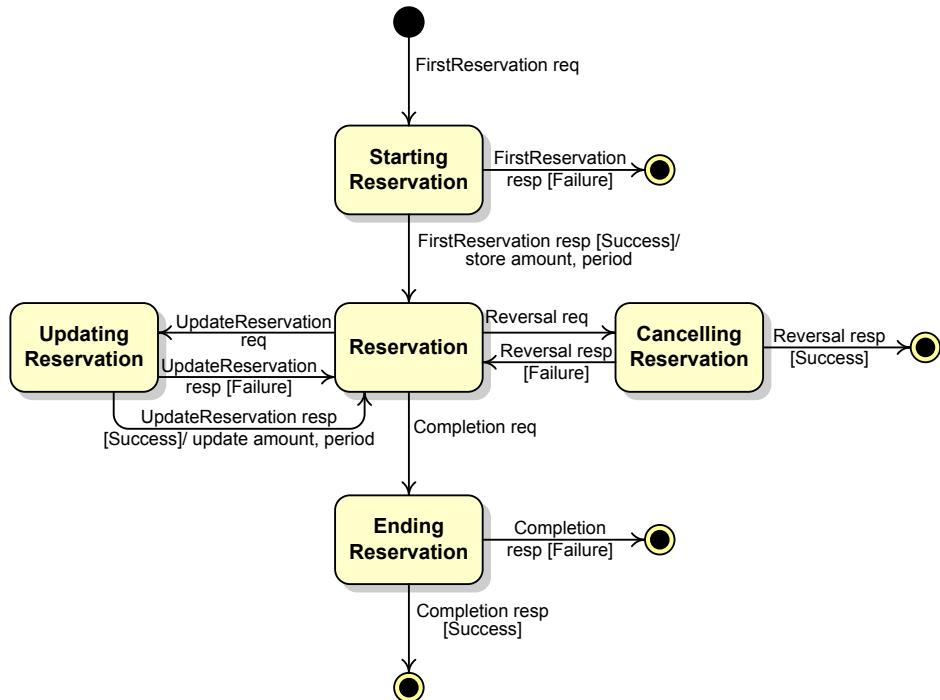


Figure 139: Reservation Transaction State Diagram

The Reservation transaction starts with the sending of the “FirstReservation” Payment request by a Sale Terminal to a POI Terminal, going to the state *Starting Reservation*.

In this state, either

- The “FirstReservation” Payment response fails, and the Reservation transaction terminates, either
- The “FirstReservation” Payment response succeeds, and the Reservation transaction is in progress going to the state *Reservation*, storing the reserved amount and the validity period.

In the *Reservation* state:

- The Reservation transaction could be updated, by the sending of an “UpdateReservation” Payment request, going the state *Updating Reservation*.
- The Reservation transaction could be cancelled, by the sending of a Reversal request, going the state *Cancelling Reservation*.
- The Reservation transaction could be completed, by the sending of a “Completion” Payment request, going the state *Ending Reservation*.

In the *Updating Reservation* state, if the UpdateReservation” Payment response succeeds, the reserved amount and the validity period are updated.

In the *Cancelling Reservation* state, at the end of the exchange, the Reservation transaction ends, cancelling the reserved amount without any transfer of money.

In the *Ending Reservation* state, either

- The “Completion” Payment response fails (for instance if the end of the validity period *ValidityDate* is exceeded, and the Reservation transaction terminates without any transfer of money), either
- The “Completion” Payment response succeeds and the Reservation transaction terminates with the payment of the requested amount.

- Rule 1: The reservation transaction shall be conform to the Figure 139: Reservation Transaction State Diagram, starting a first Payment request with the *PaymentType* set to "FirstReservation", and ending either after an unsuccessful Payment response, or a Payment request the *PaymentType* component set to "Completion" or a successful Reversal exchange.

The reservation transaction is closed if the period is expired. It is recommended that the Sale System sends a Reversal as soon as the reservation is no longer required.

- Rule 2: All the Payment messages in the message sequence of a reservation transaction shall contain the *SaleData* component *SaleReferenceID*, with the same value identifying the reservation transaction.

The *OriginalPOITransaction* could be present in the UpdateReservation and Completion Payment request messages, in particular if the update or completion is performed on the same POI system.

The *POITransactionID* of the sequence of Payment response messages must have a different value.

- Rule 3: If the "FirstReservation" Payment response is unsuccessful (i.e. the *Response.Result* value "Failure"), the reservation transaction is completed, and an "UpdateReservation" or a "Completion" Payment requests are not allowed for the same *SaleReferenceID* value. The *Response* component of such "Completion" Payment response has the value "Failure"- "NotAllowed" (Invalid *OriginalPOITransaction* Value, see section 4.6.4.1.4 *NotAllowed* Value).

- Rule 4: If "FirstReservation" or an "UpdateReservation" Payment is successful, and the Sale transaction fails, the "Completion" Payment must be replaced by a Reversal request to cancel the reservation transaction.

The *POITransactionID* of the Reversal response must have a different value than the *POITransactionID* of the Payment responses.

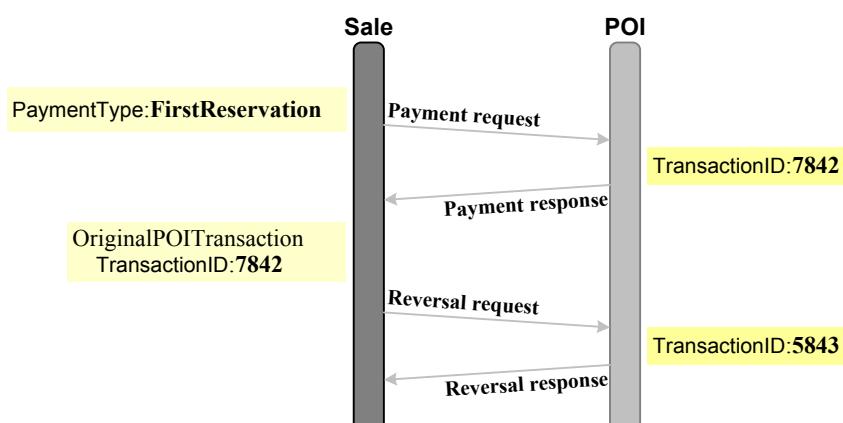


Figure 140: Cancelled Reservation Sequence Flow

- Rule 5: If the *AuthorizedAmount* component of the "FirstReservation" or "UpdateReservation" Payment response has a value lower than the value of the *RequestedAmount* component of the related Payment request, the *Response* component of the Payment response must have the value "Partial".

- Rule 6: If the reservation transaction identified by the *SaleReferenceID* or *OriginalPOITransaction* component of an "UpdateReservation" or a "Completion" Payment request is not found, the *Response* component of the Payment response must have the value "Failure"- "NotFound" (see section 4.6.4.4.1 Transaction Not Found).
- Rule 7: The *RequestedValidityDate* can be present in the OneTimeReservation, FirstReservation or UpdateReservation Payment request.
The *ValidityDate* is present in the Payment response only if *RequestedValidityDate* is decreased.
- Rule 8: If the response message of a "FirstReservation", an "UpdateReservation" or a "Completion" is not received, the error general resolution of the payment must be applied (See section 4.7.5 Error Resolution and Error Situations).
- Rule 9: After an unsuccessful "UpdateReservation", the amount and the period accepted in the previous successful reservation still applies.
- Rule 10: After a successful "FirstReservation", the reservation transaction can only be cancelled by a Reversal message.

4.3.2.5 Recurring

Recurring payment is a regular payment to pay for goods or services that the customer purchases at regular intervals.

The first transaction of a recurring payment is a standard payment with:

- The *PaymentType* set to "Recurring",
- The card must be present (i.e. the *EntryMode* has not the value "File").

The following transactions are standard payments with:

- The *PaymentType* set to "Recurring",
- The card is not necessary present (i.e. the *EntryMode* may have the value "File").

4.3.2.6 Tip

A tip is an extra amount that the Customer gives to someone, for example a waitress or taxi driver, as gratuity, in addition to the payment of a given service.

There are two data elements in the protocol devoted to the tip:

- *TipAmount*: the requested or proposed amount for the tip in *PaymentRequest.PaymentTransaction.AmountsReq*,
- *TipAmount*: the paid amount for tip in *PaymentResponse.PaymentResult.AmountsResp*.

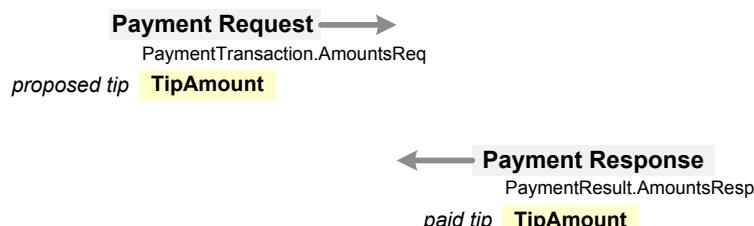


Figure 141: Tip Data Elements

If the amount *TipAmount* is present in the Payment request message, the POI terminal asks the Customer to validate this amount for the tip or update the tip amount.

If the amount *TipAmount* is absent from the Payment request message, the POI terminal asks the Customer to enter an amount for the tip.

The POI performs the payment transaction for the *RequestedAmount* increased by the new value of *TipAmount*, and present in the response if the whole payment is authorised:

$$\text{AmountsResp.AuthorizedAmount} = \text{AmountsReq.RequestedAmount} + \text{AmountsResp.TipAmount}$$

4.3.2.7 Aggregation

An aggregation transaction accumulates a collection of transactions where details of the individual payments are not provided as a normal payment transaction.

The aggregation of the individual payments is performed by the POI System or the payment Acquirer with the same interface between the Sale System and the POI system.

We are considering aggregations of individual payments, where every individual payments could require an authorisation.

There are two types of aggregations:

- *Real-time*: all the individual payments are aggregated in real-time, on a centralised server. It provides a global identification of the aggregation transaction.
- *Deferred*: the aggregation of the individual payments is performed after the individual payment processing. But some of the individual payments could be centralised. This type could be used for real-time aggregation when individual payment is in downgraded mode.

Rule 1: Every individual payments of an aggregation transaction are performed by:

- The same payment card,
- A Payment request message

4.3.2.8 Merchant Instalment

A merchant instalment payment transaction is one of a sequence of partial payments of goods or services that the customer has been purchased.

Rule 1: The first transaction of a merchant instalment is a standard payment with:

- The *PaymentType* set to "Instalment",
- The card must be present (i.e. the *EntryMode* has not the value "File"),
- The *Instalment* structure must be present with:
 - *SequenceNumber* equal to 1,
 - Depending on the Sale processing, other data elements of the *Instalment* structure could be present or absent and requested by the POI to the cashier, depending on the context.

Rule 2: The following transactions are standard payments with:

- The *PaymentType* set to "Instalment",
- The card is not necessary present (i.e. the *EntryMode* may have the value "File").
- The *Instalment* structure must be present with:
 - *SequenceNumber* greater than 1

4.3.2.9 Issuer Instalment

An issuer instalment is the authorisation of an instalment plan negotiated with the customer, for the payments of goods or services that the customer is purchasing.

After the instalment has been approved by the card issuer, the issuer performs directly the payment with the acquirer and the merchant. In most of the cases, the issuer and the acquirer are the same financial institution.

Rule 1: There is only one payment transaction for an issuer instalment with:

- *PaymentType* set to "IssuerInstalment", and
- The data structure *Instalment* present with:
 - *InstalmentType*, *PlanID*, and *CumulativeAmount* presents,
 - *Period*, *PeriodUnit*, and *TotalNbOfPayments* presents if they cannot be deduced from *PlanID* and are not requested by an Input command,
 - *SequenceNumber* absent.

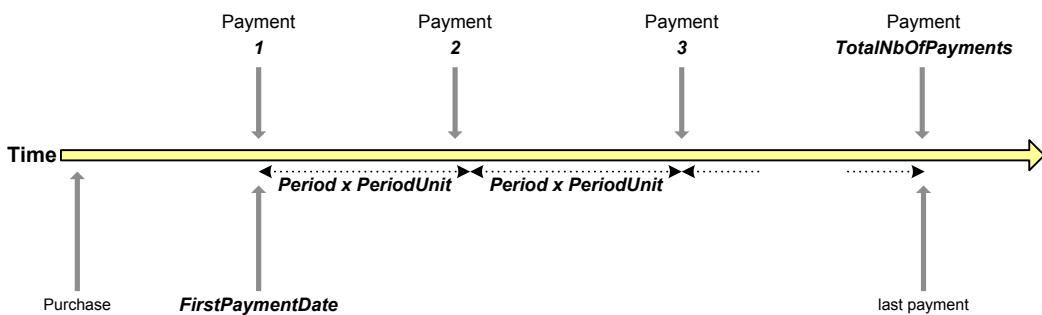


Figure 142: Issuer Instalment

Rule 2: If *InstalmentType* contains the value "DeferredInstalments", the data element *FirstPaymentDate* must be present with a value later than the date of the transaction.

Rule 3: If *InstalmentType* contains the value "InequalInstalments", the data element *FirstAmount* must be present with a value greater than "0".

Rule 4: A CardAcquisition message exchange could be performed before the issuer instalment, to be sure that an eligible card for issuer instalment is provided for the payment.
The issuer instalment transaction may be cancelled by a Reversal message.
The Reconciliation and GetTotals does not take into account issuer instalment transactions.

4.3.2.10 Voice Authorization

The voice authorization process is the following:

- The response from the Payment Acquirer requires a referral action.
- The response to Sale System is "Failure"/"Refusal", with a message on the Cashier interface to call the bank.
- During the call, the bank provides a code if the payment is approved.
- If the payment is approved, the Sale Terminal sends another Payment request message with the *OriginalPOITransaction* identifying the original payment transaction, and containing in the *ApprovalCode* data element the code provided during the call.

4.3.2.11 Refund

Refund credits a Cardholder account from the Merchant account.

The refund is initiated by the Sale System in case of reimbursement of a payment transaction, totally or partially. The POI System has to realise the process of the refund in conformance to the rule of the Card Scheme.

The Site Manager and the Sale System decide the kind of validation to proceed locally for the refund presented by the Cardholder.

Refund could be credit transaction which is not linked to a previous transaction.

A refund transaction is realised with a Payment message exchange where the particular data elements below are used:

- *RequestedAmount*: this is the actual amount of the refund.
- *PaymentType*: must be "Refund",
- *OriginalPOITransaction*: this is an optional component of the Payment request containing the *POITransactionID* component of the payment transaction to pay back with "Normal" or "Completion" *PaymentType*,
- *AuthorizedAmount*: the actual amount of the refund, as in *RequestedAmount*.

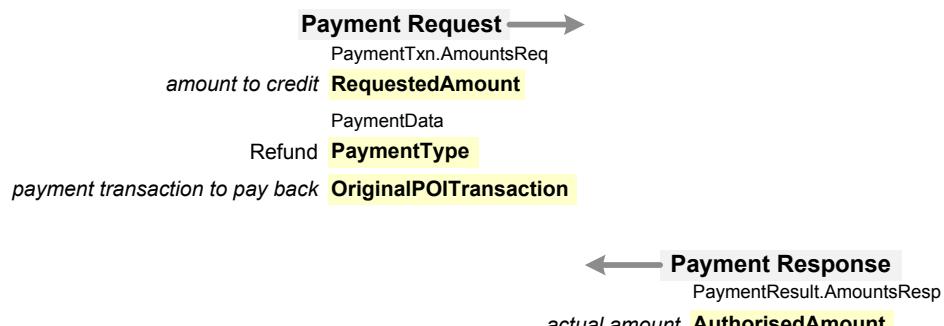


Figure 143: Payment Refund Data Elements

- Rule 1: Refund transaction is performed by a Payment request message containing the *PaymentType* set to "Refund".
If the Payment request message contains the *OriginalPOITransaction*, the *POITransactionID* component is mandatory, and *POIID* is mandatory if the original transaction was performed by another POI Terminal.
- Rule 2: If the *OriginalPOITransaction* in the Payment request message is:
- present: if allowed and achievable, the POI uses the card data recorded in the original transaction and the *EntryMode* is set to "File".
 - absent, the POI must read the card.
- Rule 3: If the *OriginalPOITransaction* is present in the Payment request message, the *RequestedAmount* of the refund Payment request must be less than or equal to *AuthorizedAmount* of the original transaction.
- Rule 4: If the *OriginalPOITransaction* is present in the Payment request message, and the transaction identified by the original transaction is not found by the POI, the *Response* component of the Payment response has the value "Failure"- "NotFound" (see section 4.6.4.4.2 *Message Not Found*).
- Rule 5: Refund transaction Payment request message follows the general error resolution (see section 4.7.5.1 Error Resolutions Specifications).

4.3.2.12 Cash Advance

Cash Advance is a card payment service which allows the Customer to withdraw cash at the Sale Terminal after the successful processing of this service by the POI Terminal.

A Cash Advance transaction is realised with a Payment message exchange where the particular data elements below are used:

- *RequestedAmount*: this is the amount of the cash the Customer wants to withdraw.
- *PaymentType*: must be "CashAdvance",
- *AuthorizedAmount*: the actual amount of the cash, less than or equal *RequestedAmount*.

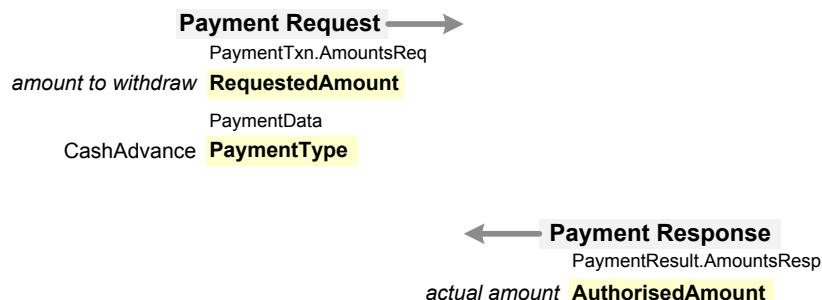


Figure 144: Cash Advance Data Elements

Rule 1: Cash Advance transaction is performed by a Payment request message containing the *PaymentType* set to "CashAdvance".

Rule 2: The *AuthorizedAmount* of the Cash Advance Payment response message must be less than or equal to the *RequestedAmount* of the related Payment request message.
If the *AuthorizedAmount* is greater than the *RequestedAmount*, a reversal has to be sent by the Sale System.

Rule 3: Cash Advance transaction Payment request message follows the general error resolution (see section 4.7.5.1 Error Resolutions Specifications).

4.3.2.13 Currency Conversion

Conversion of currency for the amounts can occur at various times with different involvement on the interface between the Sale and the POI systems:

- 1) By the Sale system before the Payment request. In this case the conversion is invisible on the interface.
- 2) By the POI, at the request of the Sale system. This case is not proposed in this version of the protocol.
- 3) By the POI, at the choice of the customer, in relation with a third party (with a currency conversion service provider directly or through an acquirer).
- 4) By other parties, for instance the card issuer of the customer. In this case the conversion is invisible on the interface.

The Payment message pair uses the following data elements for the reporting of currency conversion on the POI system:

- *Currency* and *RequestedAmount*, the currency and amount requested by the Sale system for the payment in the request message,
- *Currency* and *AuthorizedAmount*, the currency and amount to be payed by the acquirer to the merchant for the payment in the response message. The *AuthorizedAmount* is in the source currency,
- *CurrencyConversion.ConvertedAmount*: the converted amount, containing the target currency and the amount converted in this target currency. This is additional information that the Sale can print on the sale receipt.
- *CurrencyConversion.CustomerApprovedFlag*: the response of the customer (approval or decline) to the proposition of currency conversion.
- *CurrencyConversion.Rate*: the conversion rate from the source currency (*PaymentResult.AmountsResp.Currency*) into the target currency (*PaymentResult.CurrencyConversion.ConvertedAmount.Currency*).
- *CurrencyConversion.Markup*: the conversion markup (a rate).
- *CurrencyConversion.Commission*: the conversion commission (an amount).
- *CurrencyConversion.Declaration*: a declaration to the customer to print on the sale receipt.

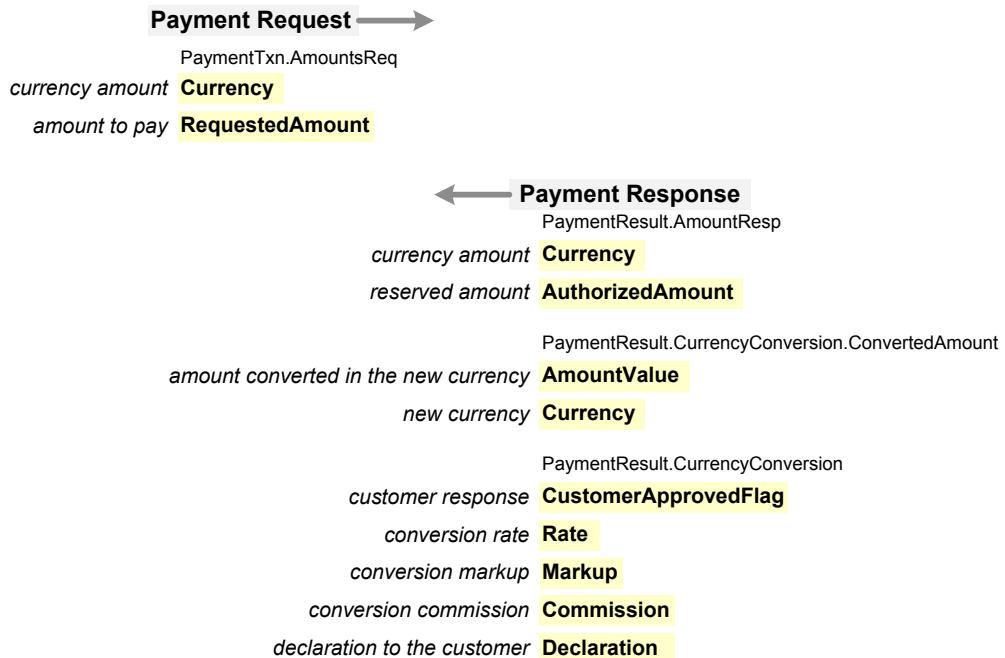


Figure 145: Currency Conversion Data Elements

4.3.2.14 Tokens

For the Sale system, a token provided by the POI system, is a substitute of the PAN which is used by the Sale system to identify a customer by its payment mean. The need of identification may be only for the time of a transaction (e.g. in a multi-channel situation), or for a longer period (e.g. to recognise a customer after the first visit at a store or at a merchant site).

For the Payment system, a payment token replaces the PAN in a payment transaction to protect the PAN by some way. A payment token is always generated by the payment system, POI system, Acquirer, or Issuer.

The payment token used by the payment system to replace the PAN during a transaction may be used or not to provide an identification of the customer to the Sale system.

We can distinguish two classes of payment token:

- 1) *Issuer Token*: the token is an alternate PAN, usable in dedicated payment domain defined during its issuance by the token service provider of the card Issuer.
The token has the same format as a PAN, and belongs to a BIN range reserved to payment tokens.
- 2) *Acquirer Token*: the token is an encrypted PAN or value attached to the PAN, generated by the POI system, an Acquirer or any Agent between, to replace the PAN during a payment transaction on a segment between the Merchant (or Card Acceptor) and the Acquirer.

Issuer Token

The Merchant (or Card Acceptor) is not aware that the PAN of the payment card is replaced by a payment token for the transaction.

The Issuer, using a Token Service Provider, translates the token into the PAN of the card.

In some environments, when an Agent (i.e. a card scheme network) routes the transactions between Acquirer and Issuer, the Agent may route the transaction to the Token Service Provider to provide the PAN associated to the payment token.

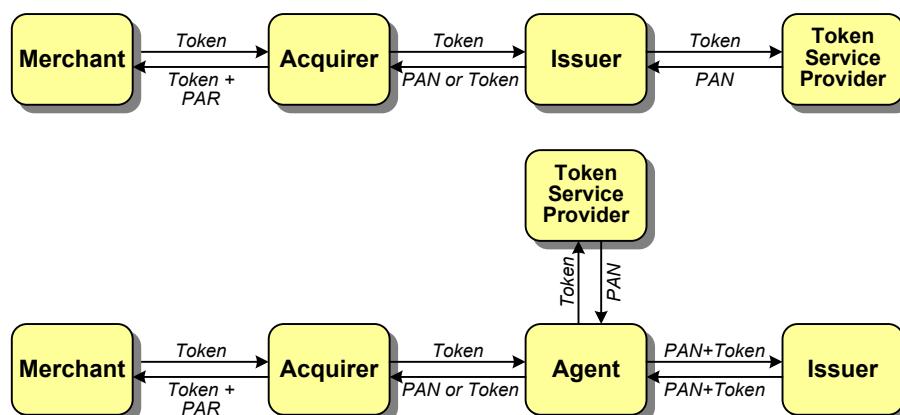


Figure 146: Issuer Token

As the Merchant, and in some case the Acquirer, are not aware that the PAN is replaced by a payment token, in the response to the payment authorisation, the payment token is completed by the *PaymentAccountRef* (PAR), an identifier of the PAN, which cannot be used in a transaction.

The *PaymentAccountRef* is the ideal candidate to identify a customer, but:

- Issuer payment tokens are not generalised enough,
- In case of offline transaction, the *PaymentAccountRef* must be stored in the payment device (card or mobile).

Acquirer Token

The payment token could be:

- A protected PAN to respect the requirements of the Point-to-Point encryption. In this case the payment token is the encrypted value of the PAN, with potentially other sensitive card data.
The scope of the payment token is the Acquirer or the Agent sharing the encryption key with the POI.
The lifetime of the payment token is the lifetime of the key.
Only the POI and the Acquirer or the Agent sharing the encryption key may find the PAN directly with the payment token.
- A value attached to the PAN as the result of a one way secure cryptographic algorithm (it could be also any random value).
Depending on the entity generating the payment token, the scope of the payment token could be the Store, the Merchant (multi-store managed by an Agent), or the Acquirer.
The lifetime is determined by the generator of the payment token.
The PAN cannot be deduced from the payment token, but the payment token generator can verify that the payment token is associated to the PAN.

We can conclude than a payment token is characterised by a type, a value and an expiry date.

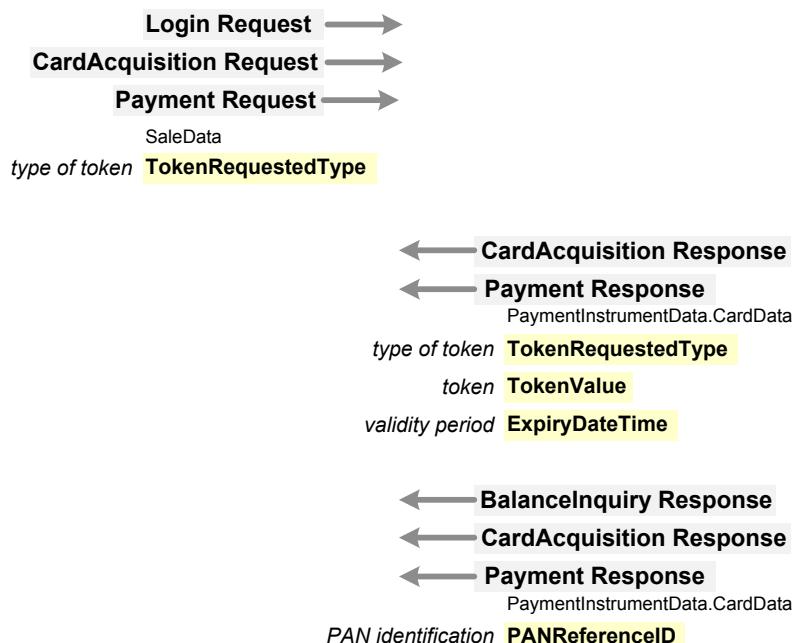


Figure 147: Payment Token Data Elements

A typical use case of token is a real time parking solution:

- At the beginning of the sale transaction, when the card is inserted inside the pay and display machine, the Sale asks to the POI to generate a payment token. Then the Sale starts a reservation payment transaction, and the pay&display machine prints the parking ticket and stores the payment token with the parking deadline.
- When the cardholder get back to the pay&display machine before the deadline, the pay&display machines will be able to update the amount of the sale and complete the payment transaction.

As a consequence, the sale system has been able to manage the cardholder without storing any sensitive data.

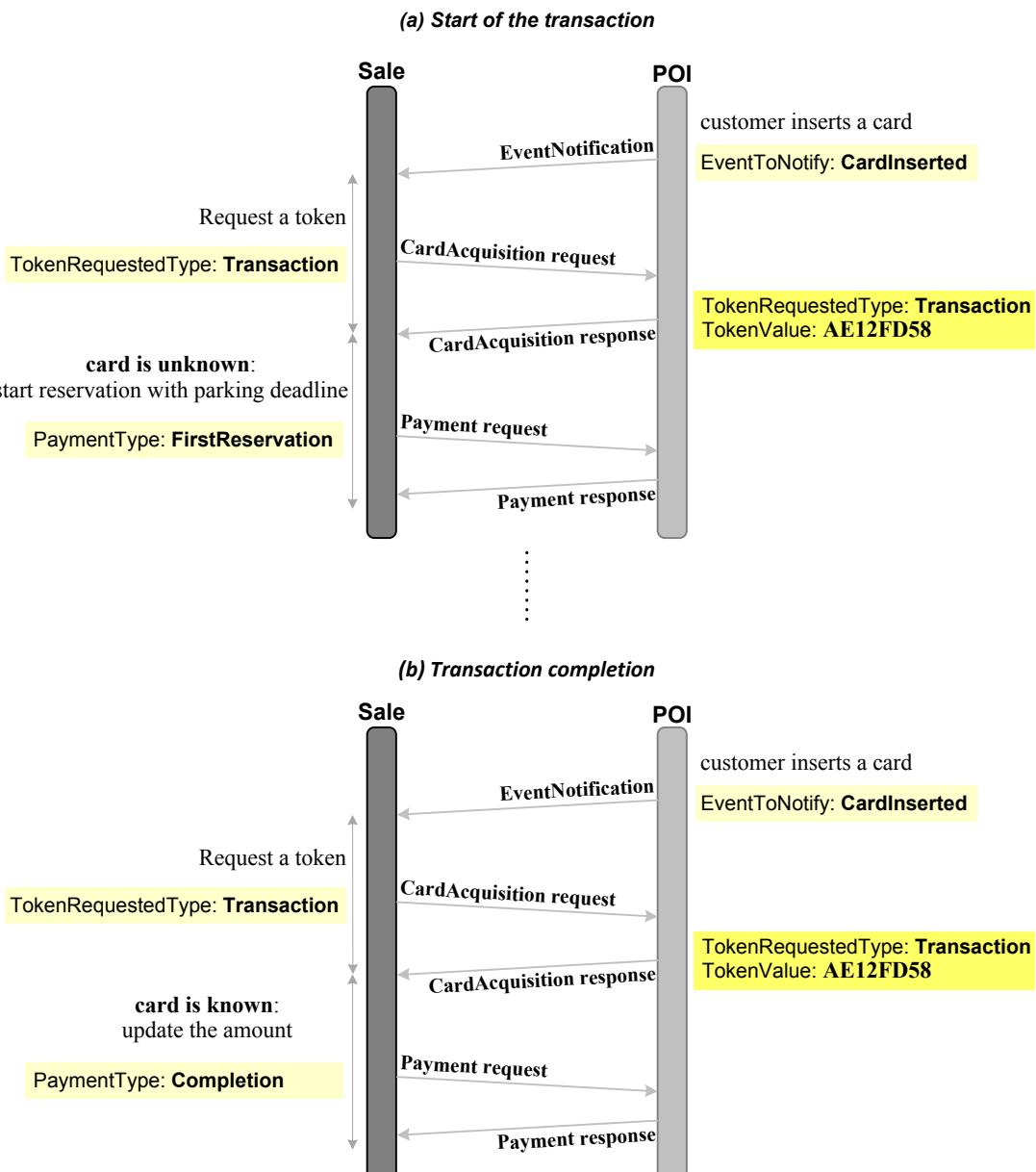


Figure 148: Real-Time Parking Using Token

Rule 1: When the *PaymentAccountRef* is available, the *PaymentAccountRef* must be provided in the Card Acquisition, Payment, and BalanceInquiry response messages.

Rule 2: If the Sale system needs to identify the payment mean of the customer performing a payment transaction:

- The type of the requested payment token *TokenRequestedType* must be present in the Card Acquisition or Payment request message,
- The *PaymentToken* must be present in the response.

If the Sale system has got a payment token in a Card Acquisition response message, *TokenRequestedType* is ignored in the Payment request following the Card Acquisition message.

Rule 3: If a payment token must be requested for each payment, the Sale system must insert the type of payment token *TokenRequestedType* either :

- Insert the type of payment token *TokenRequestedType* in the Login request message or,
- The *PaymentToken* must be present in the response.

Rule 4: If the Sale system has requested a payment token, and the POI system cannot provide it, the response does not contain the *PaymentToken* data structure, without causing a failure in the response.

4.3.2.15 Customer Orders

A sale transaction, also called a customer order (see figure Figure 164: Sale System Payment Cancellation), may require multiple steps: select the items to purchase, perform the payment, deliver the goods or provide the service.

These multiple steps, which could be mixed, are managed by the Sale System.

A typical example is the Click and Collect use case:

- The first step (click step) of the sale consists to order your goods and perform a down payment as the first payment.
- The collect step, which could occur few days, weeks or months later collect the goods and pay the rest of the amount. This step may be repeated.
- The steps may be performed on different channels.
- The payments may be performed with card present or card not present.
- As the time between the click step and the collect step may be long, the same customer could have multiple orders for the same merchant.

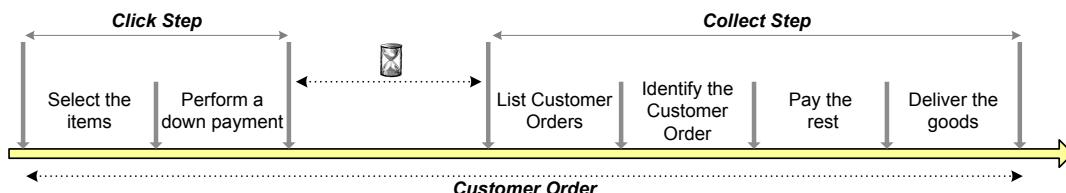
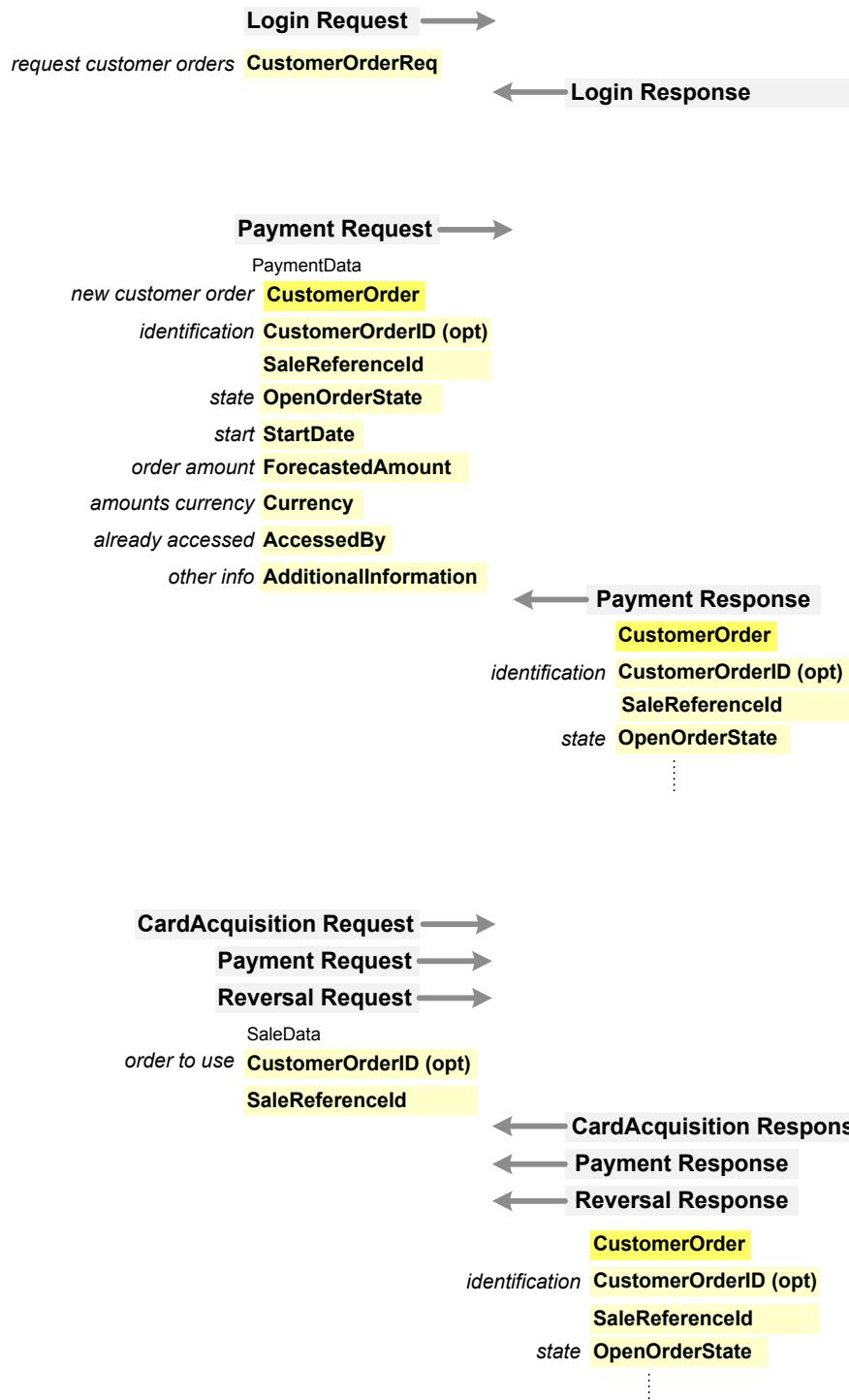


Figure 149: Click and Collect Steps

As there is a request to the POI system for each step of the sale transaction, the POI system can manage the customer orders attached to a card identifying a customer, while processing the requests of the Sale system.

The specific date elements used to manage these customer orders are presented in the figure below:



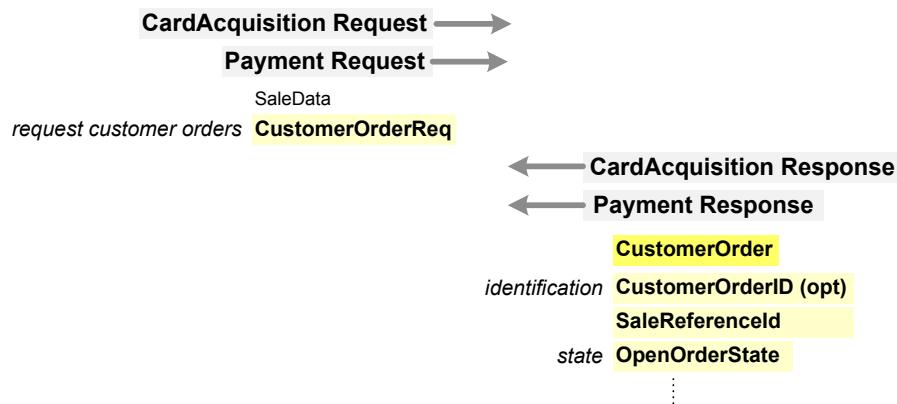


Figure 150: Customer Order Data Elements

Rule 1: Customer order creation.

A customer order is created in the POI system when the data structure *CustomerOrder* is present in the Payment request or Card Acquisition request messages with a new *SaleReferenceId*.

The *CustomerOrder* data is then created at the POI system for the card presented by the customer with the information sent in the request.

If a customer order identified by *SaleReferenceId* is already present in the POI for the same card, the corresponding action (*Payment, Refund, Cancellation, etc*) is part of the *CustomerOrder*.

If customer orders are not managed by the POI, the *Response* component of the response message must have the value “Failure”-“UnAvailable” (see section 4.6.4.3.2 *Unavailable Service for the Card*)

Rule 2: If *CustomerOrderReq* is present in a Card Acquisition, a Payment or a Reversal request message, the data structure *CustomerOrder* must be present in the response message.

After a successful completed payment, *CurrentAmount* is changed accordingly to the action. The *ForecastedAmount* is freely managed by the Sale system

If customer orders are not managed by the POI, the *Response* component of the response message must have the value “Failure”-“UnAvailable” (see section 4.6.4.3.2 *Unavailable Service for the Card*)

Rule 3: Customer order closure.

A customer order is closed when *OpenOrderState* is set to *False* by the Sale System:

Then the *EndDate* is set with the current date time.

Except this control, there is no specific management on *CustomerOrder*. Then a Sale System is still able to add an additional payment, for instance for a dispute management, on a *CustomerOrder* with *OpenOrderState* set to *False*, or to set *OpenOrderState* to *True* if more than one additional payment is needed.

Rule 4: The list of the customer orders present in the POI system for the card that will be used by the customer must be provided in the Card Acquisition or Payment response messages if *CustomerOrderReq* is present either:

- At the session level in the Login request message, or
- In the related Card Acquisition or Payment request message.

The *CustomerOrderReq* provides the list of the customer orders state to return in the response message: "Open", "Closed" or both.

A typical example of click and collect customer order with a click step including a down payment, and a collect step completing the customer order is presented in the figure below.

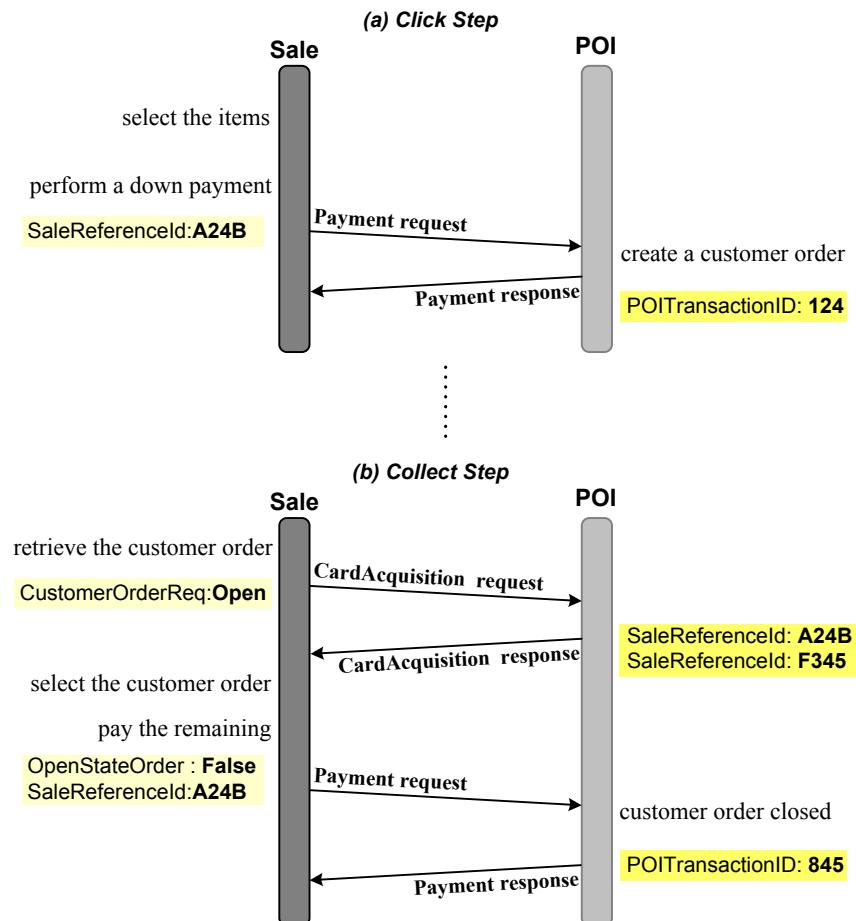


Figure 151: Click and Collect Process Flow Example

4.3.3 Card Acquisition

4.3.3.1 Presentation of the Card Acquisition Messages

The Card Acquisition request message body *CardAcquisitionRequest*, contains the following information:

1. The data structure *SaleData*, containing the same information as the Payment request message.
2. The data structure *CardAcquisitionTransaction*, containing the information related to the card(s) to read and analyse:
 - a. *TotalAmount*: the amount of the transaction required for contactless cards.
 - b. Some possible conditions or restrictions on the payment and loyalty cards: card brands, type of allowed loyalty transaction, entry modes...
 - c. The data element *ForceCustomerSelectionFlag*, requesting to leave the Customer to make selection of the smartcard application.

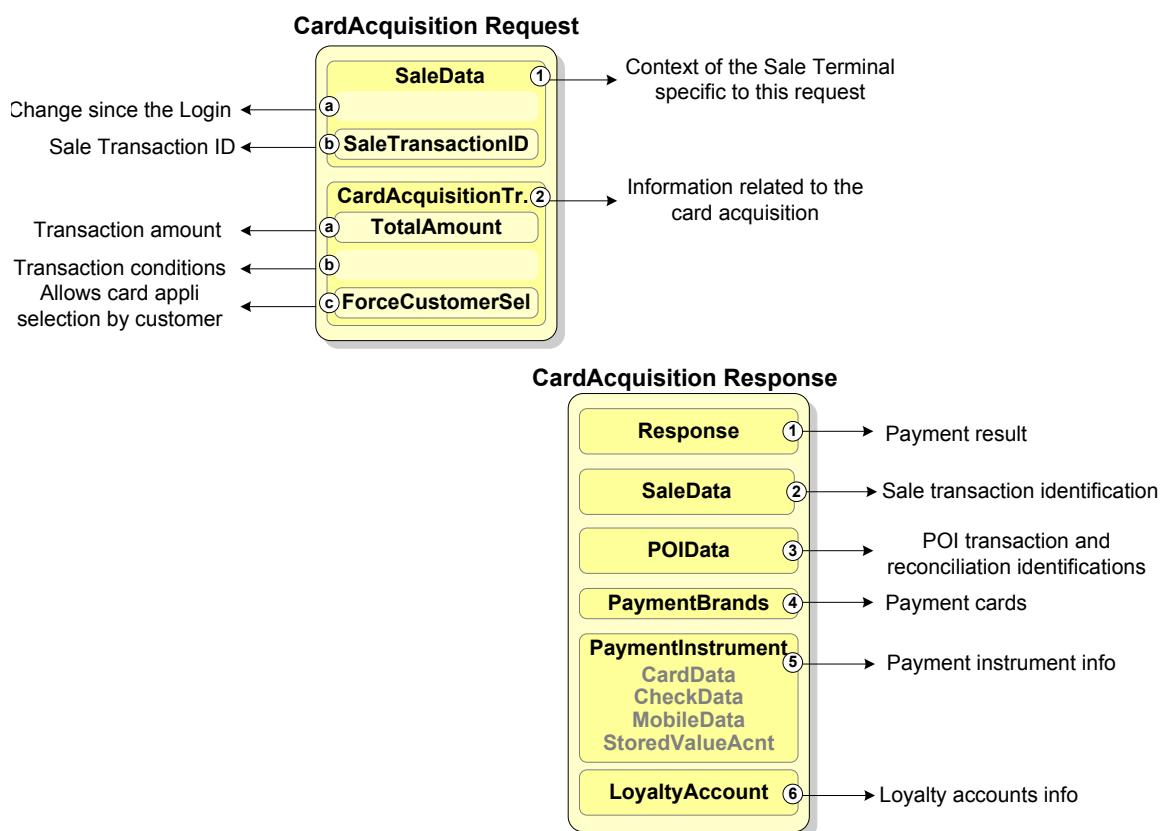


Figure 152: CardAcquisition Information

The Card Acquisition response message body *CardAcquisitionResponse*, contains the following information:

1. The result of the card acquisition and analysis: *Response*.
2. The data structure *SaleData*, containing the Sale transaction identification.
3. The data structure *POIData*, containing the POI transaction identification and the reconciliation identification.
4. The brands of the payment cards: *PaymentBrand*.

5. The *PaymentInstrumentData* containing information on the selected payment instrument: *CardData*, *CheckData*, *MobileData*, or *StoredValueAccountID*.
6. The *LoyaltyAccount* of the selected loyalty accounts if any.

4.3.3.2 Card Acquisition Request Layout

<i>CardAcquisitionRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		CardAcquisition
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
CardAcquisitionRequest	[1..1]	I	
SaleData	[1..1]		
OperatorID	[0..1]		<i>same as PaymentRequest</i>
OperatorLanguage	[0..1]		<i>same as PaymentRequest</i>
ShiftNumber	[0..1]		<i>same as PaymentRequest</i>
CustomerLanguage	[0..1]		<i>same as PaymentRequest</i>
SaleTransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
SaleReferenceID	[0..1]		Mandatory for management of CustomerOrder
SaleTerminalData	[0..1]		<i>same as PaymentRequest</i>
TerminalEnvironment	[0..1]		<i>same as PaymentRequest</i>
SaleCapabilities	[0..1]		<i>same as PaymentRequest</i>
TotalsGroupID	[0..1]		<i>same as PaymentRequest</i>
TokenRequestedType	[0..1]		If a token is requested.
CustomerOrderReq	[0..1]		If customer orders must be listed in the response message.
CardAcquisitionTransaction	[1..1]		
AllowedPaymentBrand	[0..n]		<i>same as PaymentRequest</i>
AllowedLoyaltyBrand	[0..n]	L	<i>same as PaymentRequest</i>
LoyaltyHandling	[0..1]	L	default Allowed
ForceEntryMode	[0..n]		<i>same as PaymentRequest</i>
ForceCustomerSelectionFlag	[0..1]		default False
TotalAmount	[0..1]		Mandatory for contactless card, otherwise absent.
PaymentType	[0..1]		Mandatory for contactless card, otherwise absent.
CashBackFlag	[0..1]		For contactless, True if cash back has been requested, default False. Otherwise absent.

4.3.3.3 Card Acquisition Response Layout

<i>CardAcquisitionResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		CardAcquisition
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
CardAcquisitionResponse	[1..1]	I	
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		
AdditionalResponse	[0..1]		
SaleData	[1..1]		
SaleTransactionID	[1..1]		Copy
TransactionID	[1..1]		
TimeStamp	[1..1]		
SaleReferenceID	[0..1]		Copy
POIData	[1..1]		
POITransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
CustomerLanguage	[0..1]		If the language selected for the transaction is not the default language.
PaymentBrand	[0..n]		Brands available for payment by the card and not chosen by the Customer
PaymentInstrumentData	[0..1]		If this type of payment card is configured to send information in the CardAcquisition response
PaymentInstrumentType	[1..1]		
CardData	[0..1]		If this type of payment card is configured to send information in the CardAcquisition response
MaskedPAN	[0..1]		see <i>PaymentResponse</i>
PaymentAccountRef	[0..1]		see <i>PaymentResponse</i>
EntryMode	[1..1]		
CardCountryCode	[0..1]		If available in the card
ProtectedCardData	[0..1]		SensitiveCardData protected by CMS EnvelopedData
SensitiveCardData	[0..1]		If structure non empty (unprotected)
PAN	[0..1]		if EntryMode is File, Keyed or Manual
CardSeqNumb	[0..1]		if EntryMode is File, Keyed or Manual and data available on the card
ExpiryDate	[0..1]		if EntryMode is File, Keyed or Manual and data available on the card
TrackData	[0..4]		if EntryMode is MagStripe or RFID
TrackNumb	[0..1]		default 2
TrackFormat	[0..1]		default ISO

<i>CardAcquisitionResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
TrackValue	[1..1]		
PaymentToken	[0..1]		If a token has been requested in the request message, or in the Login request for the session.
TokenRequestedType	[1..1]		Copy
TokenValue	[1..1]		
ExpiryDateTime	[0..1]		
AllowedProduct	[0..n]		If the card has restrictions on product that can be purchased.
ProductCode	[1..1]		
EanUpc	[0..1]		
ProductLabel	[0..1]		
AdditionalProductInfo	[0..1]		
CheckData	[0..1]		If PaymentInstrumentType is "Check"
BankID	[0..1]		Mandatory if TrackData absent
AccountNumber	[0..1]		Mandatory if TrackData absent
CheckNumber	[0..1]		Mandatory if TrackData absent
TrackData	[0..1]		Mandatory if CheckNumber absent
TrackNumb	[0..1]		default 2
TrackFormat	[1..1]		"E-13B" or "CMC-7"
TrackValue	[1..1]		
CheckCardNumber	[0..1]		If provided by the customer
TypeCode	[0..1]		default Personal
Country	[0..1]		Absent if country of the Sale system
MobileData	[0..1]		If PaymentInstrumentType is "Mobile"
MobileCountryCode	[0..1]		<i>see PaymentResponse</i>
MobileNetworkCode	[0..1]		<i>see PaymentResponse</i>
MaskedMSISDN	[0..1]		<i>see PaymentResponse</i>
Geolocation	[0..1]		<i>see PaymentResponse</i>
GeographicCoordinates	[0..1]		
Latitude	[1..1]		
Longitude	[1..1]		
UTMCoordinates	[0..1]		
UTMZone	[1..1]		
UTMEastward	[1..1]		
UTMNorthward	[1..1]		
ProtectedMobileData	[0..1]		<i>see PaymentResponse</i>
SensitiveMobileData	[0..1]		<i>see PaymentResponse</i>
MSISDN	[1..1]		<i>see PaymentResponse</i>
IMSI	[0..1]		
IMEI	[0..1]		<i>see PaymentResponse</i>
StoredValueAccountID	[0..1]		<i>see PaymentResponse</i>
StoredValueAccountType	[1..1]		
StoredValueProvider	[0..1]		<i>see PaymentResponse</i>
OwnerName	[0..1]		<i>see PaymentResponse</i>
ExpiryDate	[0..1]		<i>see PaymentResponse</i>
EntryMode	[1..1]		

<i>CardAcquisitionResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
IdentificationType	[1..1]		
StoredValueID	[1..1]		
LoyaltyAccount	[0..n]		If loyalty card selected by the customer
LoyaltyAccountID	[1..1]		
EntryMode	[1..1]		
IdentificationType	[1..1]		
IdentificationSupport	[1..1]		
LoyaltyID	[1..1]		
LoyaltyBrand	[0..1]		<i>see Loyalty response</i>
CustomerOrder	[0..n]		If the list of customer orders has been requested.
CustomerOrderID	[0..1]		
SaleReferenceId	[1..1]		
OpenOrderState	[0..1]		default "True"
StartDate	[1..1]		
EndDate	[0..1]		If OpenOrderState = "False".
ForecastedAmount	[1..1]		
CurrentAmount	[1..1]		
Currency	[0..1]		If multiple currencies are allowed.
AccessedBy	[0..1]		If order process in progress.
AdditionalInformation	[0..1]		

4.3.3.4 Card Acquisition and Transaction Processing

The *CardAcquisition* message pair is used to separate the selection of the payment and loyalty cards from the transactions realised with these cards for the following reasons:

- The Sale System could change the context of the payment transaction according to the type of card,
- Entering the card information only once if several transactions or successive operations use the same card,
- Card entering could be a way to identify the beginning of a transaction by the Customer for the Sale System for unattended situations,

Rule 1: CardAcquisition message pair could be requested by the Sale System to read and analyse card before the following requests: *BalanceInquiryRequest*, *LoyaltyRequest*, *PaymentRequest*, and *ReversalRequest*.

The following data elements are present to link one of these message requests to the previous CardAcquisition message pair:

- *CardAcquisitionReference*: the identification of the transaction provided in the CardAcquisition response message, including *TransactionID* and *TimeStamp*,

(*Payment*, *Loyalty* or *BalanceInquiry*) **Message Request** →

PaymentData
tx*n* identification of CardAcquisition **CardAcquisitionReference**

LoyaltyData
tx*n* identification of CardAcquisition **CardAcquisitionReference**

Figure 153: CardAcquisition Link Data Elements

These components are located in the data structure *PaymentData* for the payment card, and in the data structure *LoyaltyData* for the loyalty cards.

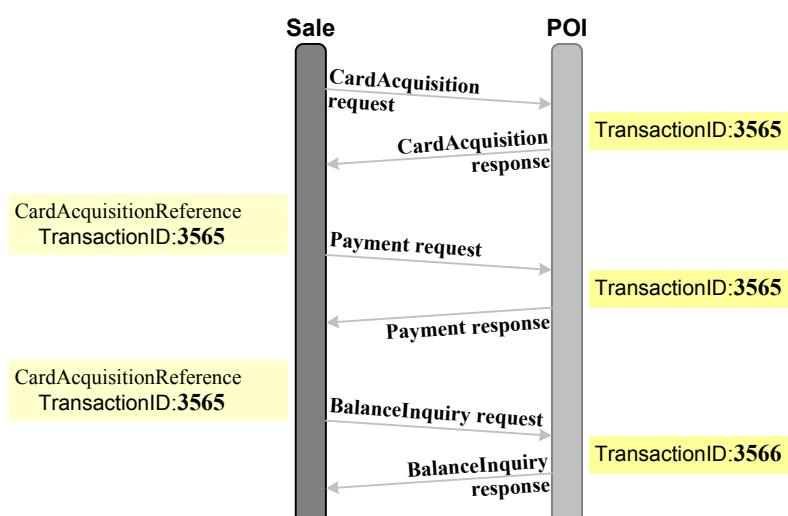


Figure 154: Card Acquisition Sequence Flow Example

- Rule 2: If the CardAcquisition response is unsuccessful (i.e. the *Response.Result* value “Failure”), there is no transaction context open related to the CardAcquisition, and any Message request referencing it is forbidden.
 In particular if CardAcquisition is not allowed by the POI, the *Response* component has the value “Failure”-“NotAllowed” (Unacceptable CardAcquisitionReference Value, see section 4.6.4.1.4 *NotAllowed Value*).
- Rule 3: If the CardAcquisition is successful, and the card is not used subsequently for a transaction, then the Sale Terminal must send an EnableService request with the *TransactionAction* component set to “AbortTransaction”. The POI terminates the card processing and removes the card from the POI Terminal card reader.

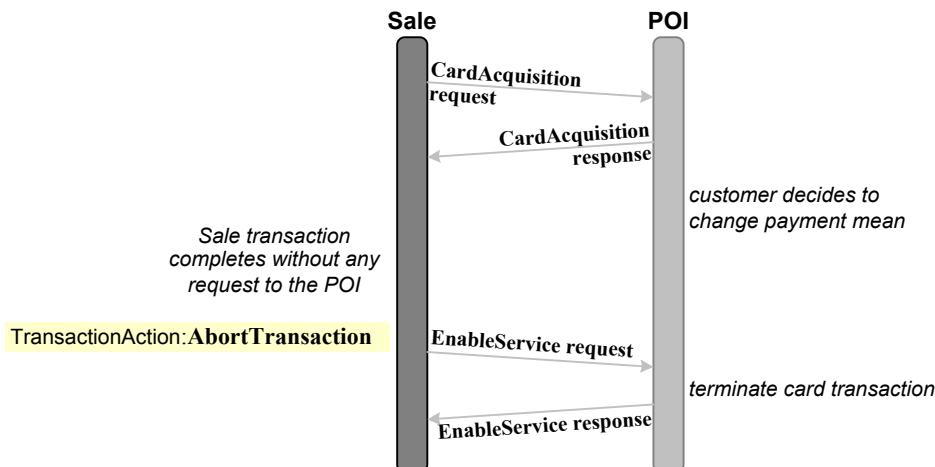


Figure 155: Card Acquisition Sequence Flow Example

This is typically the case if the Customer, after entering a card, changes his mind and pays with another payment mean where the POI is not involved.

- Rule 4: The Payment, Loyalty or BalanceInquiry request can only be linked to the last CardAcquisition transaction. If the request is not linked to the last one, the *Response* component of the message response must have the value “Failure”-“NotAllowed” (Unacceptable CardAcquisitionReference Value, see section 4.6.4.1.4 *NotAllowed Value*).
- Rule 5: If the *ForceCustomerSelectionFlag* component of the CardAcquisition request is set to “False”, the *PaymentBrand* component of the CardAcquisition response must contain all the brands of the payment application if the Customer selects a multi-application smartcard.
- Rule 6: If the *ForceCustomerSelectionFlag* component of the CardAcquisition request is set to “True”, the *PaymentBrand* component of the CardAcquisition response must contain only one brand, which is the payment application selected by the Customer on a multi-application smartcard.
- Rule 7: If the POI does not find the CardAcquisition transaction identified by the *CardAcquisitionReference* component of the Message request, the *Response* component of the Message response must have the value “Failure”-“NotFound” (see section 4.6.4.4.1 *Transaction Not Found*).

Rule 8: The *CustomerLanguage* data element is present if a language is selected by the cardholder during the processing of the card Acquisition request, and is different from the language defined in the Login or Card Acquisition requests.

It allows the Sale System to display further texts and inputs on the POI using the language selected by the customer.

Rule 9: According to the POI server capability, *AllowedProduct* can be present when a customer inserts a card having restrictions on products that can be purchased. This feature can avoid looping on a product selection until a right one is selected.

In case of an unattended fuel transaction, it allows the Sale system to display to the customer only the authorised products during the Input request which propose the choice of fuel, according to the fuel dispenser configuration and products states (nozzle available or not, tank empty for this product ...) to avoid a product selection by the customer and a payment response message with the value "Failure" – "PaymentRestriction" (see section 4.6.8.3 *PaymentRestriction Error*).

In case of attended fleet card transaction, it allows the cashier to verify that the items are payable by the fleet card, avoiding a payment failure at the first payment attempt.

Rule 10: For contactless card, additional information may be requested for Card Acquisition request to start a chip transaction:

TotalAmount: the amount of the transaction,

PaymentType: the type of payment service,

CashBackFlag: if cash back has been requested.

4.3.3.5 Error Cases

When the CardAcquisition request is successfully processed, the CardAcquisition message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved or before to try another CardAcquisition attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The CardAcquisition request is received during a Device dialogue or another Service dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

Cancel

The user has aborted the transaction on the Customer interface (e.g. the POI Terminal keyboard), because he do not want to continue the payment (e.g., problem of PIN remembering, chooses another payment mean, stop the purchase on a vending machine), see section 4.6.6.2.1 *User Cancellation*.

Abort

The Sale System sent an Abort request message before the end of the CardAcquisition request processing. The POI has aborted the CardAcquisition transaction, and sends the CardAcquisition response message with this *ErrorCondition*, to report the result of the aborted CardAcquisition (see section 4.6.6.1 *Aborted Error*).

DeviceOut

The POI element cannot start the CardAcquisition transaction, because of a temporary error on a device (e.g. printer without paper), see section 4.6.3.1.1 *POI Temporary Unavailable*.

The POI element cannot start CardAcquisition transaction, because of a permanent error on a device (e.g. card reader out of order, pin-pad disconnected). This error is returned just after the problem occurs on the POI Terminal, during the payment processing or because the Sale has not taken into account the reception of the Event Notification reporting the occurrence of the problem (see section 4.6.3.1.2 *POI Permanently Unavailable*).

InvalidCard

The POI terminates the transaction because no card is entered by the Customer, or the inserted card is not configured in the system and cannot be processed (see section 4.6.7.1.1 *No Card Entered* and section 4.6.7.1 *InvalidCard Error*).

4.3.3.6 Example

The following example provides first a CardAcquisition:

MessageHeader

MessageClass	Service
MessageCategory	CardAcquisition
MessageType	Request
ServiceID	653
SaleID	SaleTermA
POIID	POITerm1

*(message example 17)***CardAcquisitionRequest****SaleData**

SaleTransactionID	
TransactionID	589
TimeStamp	2009-08-05T22:17:52.33+01:00

CardAcquisitionTransaction

LoyaltyHandling	Forbidden
ForceCustomerSelectionFlag	True

MessageHeader*(message example 18)*

MessageClass	Service
MessageCategory	CardAcquisition
MessageType	Response
ServiceID	653
SaleID	SaleTermA
POIID	POITerm1

CardAcquisitionResponse**Response**

Result	Success
--------	---------

SaleData

SaleTransactionID	
TransactionID	589
TimeStamp	2009-08-05T22:17:52.33+01:00

POIData

POITransactionID	
TransactionID	491
TimeStamp	2009-08-05T22:18:34.06+01:00
POIReconciliationID	200903101
PaymentBrand	CardFlash

Followed by a Payment message pair:

MessageHeader	<i>(message example 19)</i>	
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	654	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	589	
TimeStamp	2009-08-05T22:17:52.33+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	174	
TransactionConditions		
LoyaltyHandling	Forbidden	
PaymentData		
PaymentType	Normal	
CardAcquisitionReference		
TransactionID	491	
TimeStamp	2009-08-05T22:18:34.06+01:00	

MessageHeader	<i>(message example 20)</i>	
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	654	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentResponse		
Response		
Result	Success	
SaleData		
SaleTransactionID		
TransactionID	589	
TimeStamp	2009-08-05T22:17:52.33+01:00	
POIData		
POITransactionID		
TransactionID	491	
TimeStamp	2009-08-05T22:18:34.06+01:00	
POIReconciliationID	200903101	
PaymentResult		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	CardFlash	
EntryMode	ICC	
ProtectedCardData		
ContentType	id-digestedData	
DigestedData		
DigestAlgorithm		
Algorithm	id-sha256	

EncapsulatedContent	
ContentType	id-data
Digest	FE52FFF263BEDD328746B6C5414D259A71883CFA94526187D81C8FF AF758722A
AmountsResp	
AuthorizedAmount	174
PaymentAcquirerData	
AcquirerID	497867
MerchantID	mer77-130209
AcquirerPOIID	456
ApprovalCode	9473

4.3.4 Loyalty Services

4.3.4.1 Loyalty Services Overview

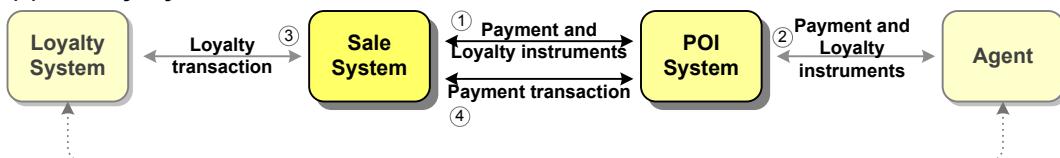
Depending on the loyalty program, there are three possible repartitions of loyalty services between the Sale system and the POI system.

- (1) *Sale System Loyalty*: The Sale system is in charge of managing all loyalty services.
- (2) *POI Loyalty Account Identification*: The loyalty account is identified by the POI system, before to perform any loyalty or payment transaction.
- (3) *POI System Loyalty*: Loyalty transactions are performed by the POI system.

(1) Sale System Loyalty



(2) POI Loyalty Account Identification



(3) POI System Loyalty

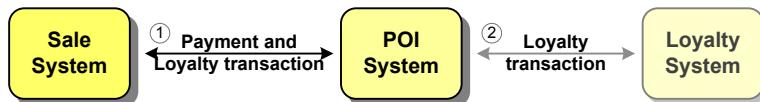


Figure 156: Sale and POI Systems Loyalty Services Management

1) Sale System Loyalty

The Sale system is in charge of the managing of all Loyalty services, including the customer management. The Sale system:

- Sends Payment request messages to the POI system without loyalty (*LoyaltyHandling* has the value "Forbidden" or "Processed"),
- Never sends Loyalty request messages to the POI system.

If a payment transaction requires a loyalty transaction, the Sale system performs the loyalty transaction before or after the payment transaction. The Sale System is also responsible to cancel a loyalty transaction if the payment transaction is not successful and performed after the loyalty transaction.

2) POI Loyalty Account Identification

The Sale system is in charge of the managing of all Loyalty services, including the customer management.

The POI system is in charge of the identification of the loyalty account, because the loyalty account is linked to the payment instrument used by the Customer, or the payment instrument is hybrid.

- The Sale system sends first a Card Acquisition request message to the POI system.
- The POI system performs the selection of the payment instrument by the Customer. The POI system analyses the payment card and detects if a loyalty account is linked to the payment instrument, by configuration or asking to a dedicated service provider.
- The Sale system sends Payment request messages to the POI system without loyalty (*LoyaltyHandling* has the value "Forbidden" or "Processed"),

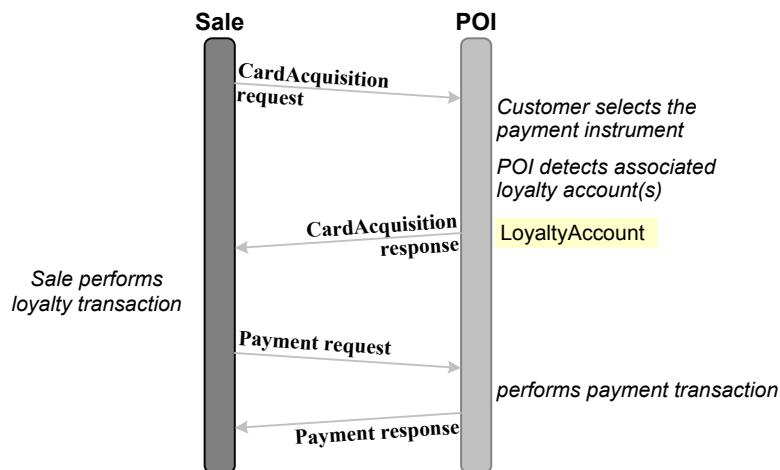


Figure 157: POI Identification of a Loyalty Account

If a payment transaction requires a loyalty transaction, the Sale system performs the loyalty transaction before or after the payment transaction, and the Sale system never sends Loyalty request messages to the POI system.

3) POI System Loyalty

The POI system manages loyalty services based on product or payment instruments only. This case typically occurs when the POI system manages payment instruments that are restricted to some products as fleet card, or an Intermediary Agent to process these added value cards.

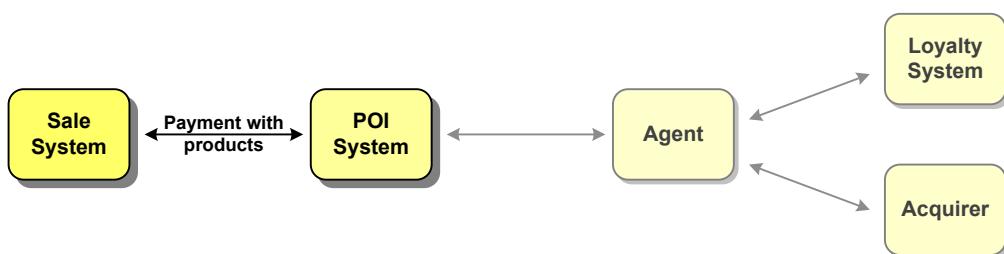


Figure 158: Example of POI Loyalty Services Management

The Sale system sends:

- Payment request messages to the POI system with loyalty (*LoyaltyHandling* has a value "Allowed", "Proposed" or "Required"),

- Loyalty request messages to the POI system, when the loyalty transaction is not associated to a payment transaction managed by the POI system.

4.3.4.2 Presentation of the Loyalty Messages

The Loyalty request message body *LoyaltyRequest*, contains the following information:

1. The data structure *SaleData*, containing the context of the Sale Terminal specific to this payment request:
 - a. The information which has changed since or was absent from the last Login request (*OperatorID*, *OperatorLanguage*...).
 - b. The identification of the transaction for the Sale System *SaleTransactionID*, containing the identification completed by a time stamp to ensure the uniqueness of the identification.
 - c. Information not interpreted by the POI, for the POI log, the Acquirer or the Issuer.
2. The data structure *LoyaltyTransaction*, containing the information related to the transaction, common to all the loyalty brands:
 - a. The loyalty transaction type *LoyaltyTransactionType*, which identifies the loyalty service: "Award", "Rebate", "Redemption", "AwardRefund", "RebateRefund", or "RedemptionRefund".
 - b. The amount *TotalAmount* and the currency *Currency*.
 - c. Link to a previous transaction *OriginalPOITransaction*.
 - d. The data structure *TransactionConditions*, containing possible conditions or restrictions on the loyalty: type of allowed loyalty brands *AllowedLoyaltyBrand*, chosen customer language *CustomerLanguage*, force an online interrogation *ForceOnlineFlag*, and restrict the card entry modes *ForceEntryMode*.
 - e. The data structure *SaleItem*, containing sold items if they could be necessary for the loyalty.
3. The repeated data structure *LoyaltyData*, containing specific information to the loyalty:
 - a. Link to the previous card reading (*CardAcquisitionReference*).
 - b. The data structure *LoyaltyAccountID*, if the card is read by the Sale Terminal.
 - c. The data structure *LoyaltyAmount*, to store an amount on the loyalty account independently of the loyalty transaction.

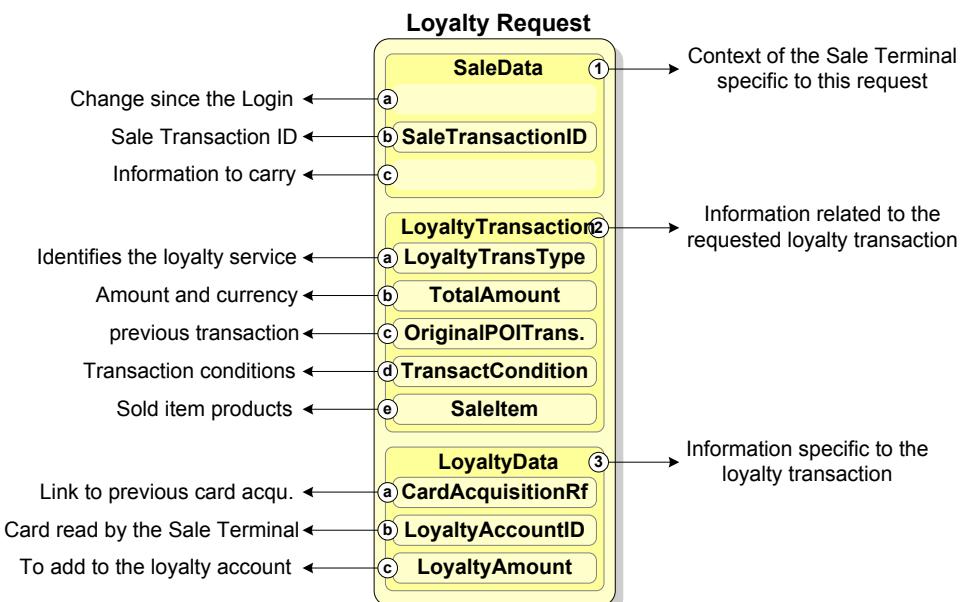


Figure 159: Loyalty Request Information

The Loyalty response message body *LoyaltyResponse*, contains the following information:

1. The result of the Loyalty transaction: *Response*.
2. The data structure *SaleData*, containing the Sale transaction identification.
3. The data structure *POIData*, containing the POI transaction identification and the reconciliation identification.
4. The repeated data structure *LoyaltyResult*, containing the result of the loyalty transactions:
 - a. Information on the loyalty account: *LoyaltyAccount*. It contains the identification of the loyalty account *LoyaltyAccountId* and the loyalty program name *LoyaltyBrand*.
 - b. The amounts of the response: *LoyaltyAmount*.
 - c. Identification of the transaction for the loyalty host: *LoyaltyAcquirerData*.
 - d. Information on the rebates, related to the total or per sold items: *Rebates*.

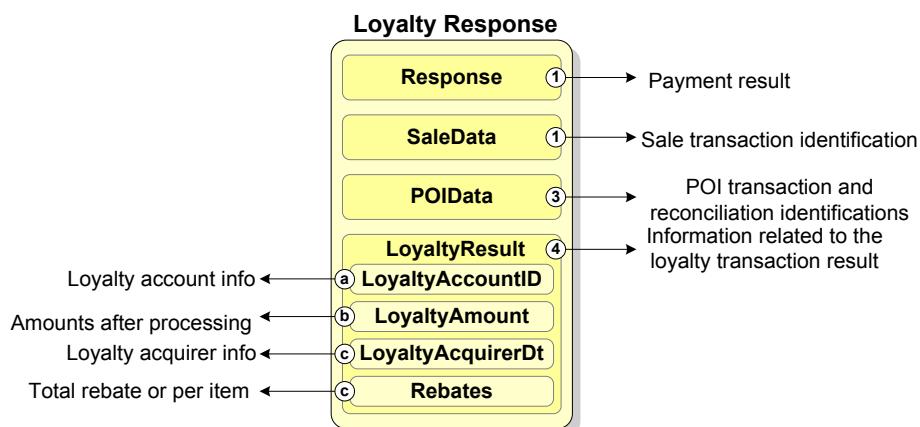


Figure 160: Loyalty Response Information

When the Loyalty transaction is realised through the Payment message pair, the content of the *LoyaltyResponse*, i.e. the data structures *Response*, *POIData*, and *LoyaltyResult* are with the same format in the *PaymentResponse* message.

4.3.4.3 Presentation of the Loyalty Part of the Payment Messages

When the Loyalty transaction is realised through the Payment message pair, the components *SaleData* and *LoyaltyData* are with the same format in the *PaymentRequest* message.

The components *LoyaltyTransaction* and *PaymentTransaction* are different in the 2 messages:

- *TotalAmount* and *Currency* are included or deduced from *AmountsReq* components.
- *TransactionConditions*:
 - *LoyaltyTransactionType* absent in the Payment is deduced from the *PaymentType* data element.
 - *LoyaltyHandling* is included only in the Payment message, to indicate if the loyalty is possible or required with the payment transaction.
 - *AllowedLoyaltyBrand*, *CustomerLanguage*, *ForceOnlineFlag*, and *ForceEntryMode* are common in the two messages.
- *SaleItem* is the same in both messages.

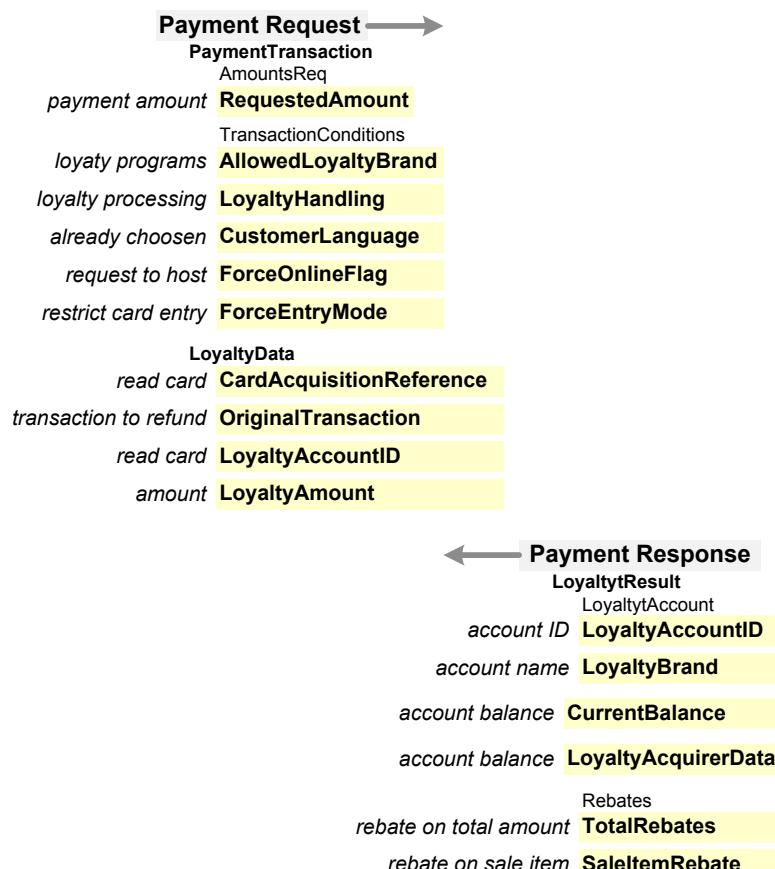


Figure 161: Loyalty Components of the Payment Messages

The component *LoyaltyResult* is the same for the 2 messages.

4.3.4.4 Loyalty Request Layout

LoyaltyRequest Component	Mult.	Profile	Rule
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Loyalty
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
LoyaltyRequest	[1..1]		
SaleData	[1..1]		same as <i>PaymentRequest.SaleData</i>
OperatorID	[0..1]		
OperatorLanguage	[0..1]		
ShiftNumber	[0..1]		
SaleTransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
SaleTerminalData	[0..1]		
TerminalEnvironment	[0..1]		
SaleCapabilities	[0..1]		
TotalsGroupID	[0..1]		
SaleToPOIData	[0..1]		
SaleToAcquirerData	[0..1]		
SaleToIssuerData	[0..1]		
StatementReference	[0..1]		
LoyaltyTransaction	[1..1]		
LoyaltyTransactionType	[1..1]		
Currency	[0..1]		If TotalAmount is present, and different from default currency
TotalAmount	[0..1]		If the loyalty transaction is linked to a payment transaction
OriginalPOITransaction	[0..1]	B L	if LoyaltyTransactionType is "AwardRefund", "RebateRefund" or "RedemptionRefund"
POITransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
ReuseCardDataFlag	[0..1]		default True
TransactionConditions	[0..1]		If one data element is present
AllowedLoyaltyBrand	[0..n]	L	Restrict brand if data sent and no LoyaltyData.CardAcquisitionReference
CustomerLanguage	[0..1]		see <i>Payment request</i>
ForceOnlineFlag	[0..1]		default False. Go online if data sent
ForceEntryMode	[0..n]		Restrict entry mode if sent and no LoyaltyData.CardAcquisitionReference
SaleItem	[0..n]		If relevant for the loyalty transaction (see <i>PaymentRequest.transactionData</i>)
ItemID	[1..1]		

<i>LoyaltyRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
ProductCode	[1..1]		
EanUpc	[0..1]		
UnitOfMeasure	[0..1]		if Quantity present
Quantity	[0..1]		
UnitPrice	[0..1]		if Quantity present
ItemAmount	[1..1]		
TaxCode	[0..1]		
SaleChannel	[0..1]		
ProductLabel	[0..1]		
AdditionalProductInfo	[0..1]		
LoyaltyData	[0..n]		If content is not empty
CardAcquisitionReference	[0..1]		if the loyalty account ID comes from the previous CardAcquisition
TransactionID	[1..1]		
TimeStamp	[1..1]		
LoyaltyAccountID	[0..1]		If loyalty identification of the loyalty account is realised by the Sale System
EntryMode	[1..1]		
IdentificationType	[1..1]		
LoyaltyID	[1..1]		
LoyaltyAmount	[0..1]		If loyalty amount is not computed on TotalAmount
LoyaltyUnit	[0..1]		default Point
Currency	[0..1]		if LoyaltyUnit is Monetary
AmountValue	[1..1]		

4.3.4.5 Loyalty Response Layout

<i>LoyaltyResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Loyalty
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
LoyaltyResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>
SaleData	[1..1]		
SaleTransactionID	[1..1]		Copy
TransactionID	[1..1]		
TimeStamp	[1..1]		
POIData	[1..1]		
POITransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
POIReconciliationID	[1..1]		
LoyaltyResult	[0..n]		<i>see PaymentResponse</i>
LoyaltyAccount	[1..1]		
LoyaltyAccountID	[1..1]		
EntryMode	[1..1]		
IdentificationType	[1..1]		
IdentificationSupport	[1..1]		
LoyaltyID	[1..1]		
LoyaltyBrand	[0..1]		If a card is analysed
CurrentBalance	[0..1]		If known (provided by the card or an external host)
LoyaltyAmount	[0..1]		Mandatory if transaction success and no rebate.
LoyaltyUnit	[0..1]		default Point
Currency	[0..1]		If LoyaltyUnit is Monetary
AmountValue	[1..1]		
LoyaltyAcquirerData	[0..1]		If acquirer contacted
LoyaltyAcquirerID	[0..1]		If available
ApprovalCode	[0..1]		If provided by the Acquirer
LoyaltyTransactionID	[0..1]		If provided by the Loyalty Acquirer
TransactionID	[1..1]		
TimeStamp	[1..1]		
HostReconciliationID	[0..1]		If provided by the Acquirer
Rebates	[0..1]		If rebates awarded
TotalRebate	[0..1]		If rebate on the total amount for this loyalty

<i>LoyaltyResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
			program
RebateLabel	[0..1]		If provided by the Acquirer
SaleItemRebate	[0..n]		only item with rebate (identified by ItemID)
ItemID	[1..1]		
ProductCode	[1..1]		
EanUpc	[0..1]		if present in the related SaleItem
UnitOfMeasure	[0..1]		if Quantity present
Quantity	[0..1]		If rebate is additional units
ItemAmount	[0..1]		If rebate on the line item amount
RebateLabel	[0..1]		If provided by the Acquirer
PaymentReceipt	[0..*]		If Basic profile implementation with no printer on the POI.
DocumentQualifier	[1..1]		SaleReceipt or CashierReceipt
IntegratedPrintFlag	[0..1]		<i>same as PrintRequest</i>
RequiredSignatureFlag	[0..1]		default False.
OutputContent	[1..1]		
OutputFormat	[1..1]		Text, XHTML
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		<i>same as Display</i>
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
EndOfLineFlag	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>

4.3.4.6 Loyalty Processing

This section specifies the loyalty service processing initiated by a Payment or a Loyalty request message.

Loyalty associates a loyalty account to a loyalty identifier which is generally a card. This account is used for earning points in order to buy special gift, or for earning rebates on the total or some particular item of the purchase.

Loyalty transactions are realised on the POI in two ways:

- (a) In the same time and synchronised with a payment transaction. The Sale Terminal uses the Payment request message to ask for both payment and loyalty transactions, and get back the result of these two transactions in the Payment response message.
- (b) When the payment is not realised by the POI, or when the loyalty transaction is not synchronised to a payment transaction. The Sale Terminal uses the dedicated Loyalty message pair.

Loyalty services the POI could provide are:

- *Loyalty Award*: credit a loyalty account.
- *Loyalty Rebate*: rebate or get free items on a purchase.
- *Loyalty Redemption*: debit on a loyalty account.
- *Loyalty Award Refund*: refund a loyalty award transaction.
- *Loyalty Rebate Refund*: refund a rebate.
- *Loyalty Redemption Refund*: refund a loyalty redemption transaction.

The Payment message pair, for payment and loyalty synchronisation, offers the following services:

- *Loyalty Award*, *Loyalty Rebate*, *Loyalty Award Refund*, and *Loyalty Rebate Refund*.

The Payment message pair does not provide the loyalty redemption services. Particularly it does not allow to buy a gift associated to a loyalty account, involving the synchronisation with a unique Payment message exchange of both the gift payment and the redemption of the loyalty account to get a special price. The sequence of a Payment message exchange and a Loyalty message exchange could achieve the same result.

The Loyalty message pair, for loyalty alone, offers the following services:

- *Loyalty Award*, *Loyalty Rebate*, *Loyalty Redemption*, *Loyalty Award Refund*, *Loyalty Rebate Refund*, and *Loyalty Redemption Refund*.

A loyalty account contains point or monetary value (amount and currency).

Rule 1: The loyalty service is requested by the *LoyaltyTransactionType* component of the Loyalty message.

Rule 2: In a Payment message, *LoyaltyHandling* indicates the required loyalty processing (no loyalty for the values "Forbidden" or "Processed", possible loyalty transaction for "Allowed" or "Proposed", a loyalty transaction for "Required")
The loyalty service is award or rebate, depending on the loyalty program, for all the values of *PaymentType* except "Refund".
A Payment request with *PaymentType* set to "Refund" shall make a refund of the loyalty transaction realised in the original transaction if any.

Rule 3: With a Payment message, the points or the amount to earn or refund for a loyalty account are based on the *AmountsReq* of the request message. In addition, *LoyaltyAmount* could be included to make an additional award on the identified loyalty account.

Rule 4: With a Loyalty message, the points or the amount to take into consideration for the loyalty transaction is either:

- The *LoyaltyAmount* component of the request message (this amount could be different for each loyalty program), for a redemption or a refund,
- The *TotalAmount* which indicates the amount of a related payment transaction, to compute the earning or the rebate.

Rule 5: Identification of the loyalty account is realised either by:

- The Sale Terminal, placing the *LoyaltyAccountID* component in the request (inside *LoyaltyData*), or
- The previous CardAcquisition message exchange, including the *CardAcquisitionReference* component in the request (inside *LoyaltyData*), or
- A previous processed transaction, including the *OriginalPOITransaction* component with *ReuseCardDataFlag* set to "True" in the request (inside *PaymentTransaction* or *LoyaltyTransaction*), or
- The POI Terminal, when all these components are absents in the request.

Rule 6: If the original transaction identified by *OriginalPOITransaction* is unsuccessful (i.e. the *Response.Result* has the value "Failure"). The *Response* component of the related response message has the value "Failure"- "NotAllowed" (Invalid *OriginalPOITransaction* Value, see section 4.6.4.1.4 NotAllowed Value)

Rule 7: If the POI does not find the transaction identified by the *OriginalPOITransaction* component of the request message, the *Response* component of the response message must have the value "Failure"- "NotFound" (see section 4.6.4.4.1 Transaction Not Found).

Rule 8: The *Result* component of the response message is set to "Success" if and only if all the loyalty transactions are successful. Otherwise the loyalty transactions must be reversed by the POI.

Rule 9: The *Rebates* component is present in the response message if this service is requested in the Loyalty message, or provided in addition to the payment transaction:

- If a rebate is provided on the total amount of the purchase, *TotalRebate* contains its value, and *RebateLabel* has to be printed on the receipt if present.
- If a rebate is provided on the amount of the item identified by *ItemID*, in the related *SaleItemRebate*, *ItemAmount* contains its value, and *RebateLabel* has to be printed on the receipt if present.
- If a rebate is provided as additional free elements of the item identified by *ItemID*, in the related *SaleItemRebate*, *Quantity* contains its value, and *RebateLabel* has to be printed on the receipt if present.

Rule 10: If the Loyalty response is not received by the Sale System for some reason, the Sale System must follow the general error resolution (see section 4.7.5.1 *Error Resolutions Specifications*):

4.3.4.7 Error Cases

When the request is successfully processed, the response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

UnavailableService

The Loyalty financial service is not available in the POI System (see section 4.6.4.3.2 *Unavailable Service for the Card*).

DeviceOut

The Loyalty financial service couldn't be provided by the POI System, e.g. the communication to the Host is out of order (see section 4.6.3.1 *DeviceOut Error*).

Cancel

The user has aborted the transaction on the Customer interface (e.g. the POI Terminal keyboard), because he does not want to continue the transaction, see section 4.6.6.2.1 *User Cancellation*.

UnreachableHost

The Host is unreachable or has not answered to an online request, so it is considered as temporary unavailable. The Cashier has not forced the transaction, and the payment cannot be accepted (see section 4.6.8.1.1 *Host Unreachable* and section 4.6.8.1.2 *No Host Answer*).

Refusal

The transaction is refused by the Host. A specific message is normally displayed to the Customer and the Cashier if they are presents, the information below could be logged for further information (see section 4.6.8.2 *Refusal Error*).

4.3.4.8 Examples

a) Loyalty with Payment

The Sale Terminal reads the payment card with a bar-code and sends card data to the POI Terminal.
The card is both a payment and a loyalty card.

MessageHeader		(message example 21)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	651	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	581	
TimeStamp	2009-08-09T17:17:21.9+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	38.01	
TransactionConditions		
LoyaltyHandling	Require	
PaymentData		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
EntryMode	Scanned	
SensitiveCardData		
PAN	7011014570541535	
LoyaltyData		
LoyaltyAccountID		
EntryMode	Scanned	
IdentificationType	PAN	
LoyaltyID	7011014570541535	

The loyalty account earns 5 points, involving a balance of 15 points on the account.

MessageHeader		(message example 22)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	651	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentResponse		
Response		
Result	Success	
SaleData		
SaleTransactionID		
TransactionID	581	
TimeStamp	2009-08-09T17:17:21.9+01:00	
POIData		
POITransactionID		
TransactionID	491	
TimeStamp	2009-08-09T17:20:52.4+01:00	
POIReconciliationID	200903101	
PaymentResult		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	MerRel	
EntryMode	MagStripe	
SensitiveCardData		
PAN	7011014570541535	
AmountsResp		
AuthorizedAmount	38.01	
PaymentAcquirerData		
MerchantID	mer77-130209	
AcquirerPOIID	65-POITerm1	
LoyaltyResult		
LoyaltyAccount		
LoyaltyAccountID		
EntryMode	Scanned	
IdentificationType	PAN	
IdentificationSupport	HybridCard	
LoyaltyID	7011014570541535	
LoyaltyBrand	MerBonus	
CurrentBalance	15	
LoyaltyAmount		
AmountValue	5	

a) Payment with Rebates

The Sale System sends the sold item products to be paid. The content of the basket is a pack of 5 bottles of mineral water and 38.23 litres of Diesel.

MessageHeader		(message example 23)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	643	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	592	
TimeStamp	2015-04-06T18:26:34.1+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	38.52	
TransactionConditions		
LoyaltyHandling	Forbidden	
SaleItem		
ItemID	1	
ProductCode	673	
EanUpc	84116369	
Quantity	5	
UnitPrice	0.46	
ItemAmount	2.30	
ProductLabel	Mineral Water 1,5L	
SaleItem		
ItemID	2	
ProductCode	101	
UnitOfMeasure	Litre	
Quantity	38.23	
UnitPrice	0.869	
ItemAmount	33.22	
PaymentData		
PaymentType	Normal	

The 5 bottles of mineral water get a global rebate of 0.15 EUR, thanks to the fleet card which is a linked card.

MessageHeader		(message example 24)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	643	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentResponse		
Response		
Result	Success	
SaleData		
SaleTransactionID		
TransactionID	592	
TimeStamp	2015-04-06T18:26:34.1+01:00	
POIData		
POITransactionID		
TransactionID	479	
TimeStamp	2015-04-06T18:26:43.9+01:00	
POIReconciliationID	200903101	
PaymentResult		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	FleetUniverse	
EntryMode	MagStripe	
SensitiveCardData		
PAN	6900014570541535	
ExpiryDate	0411	
AmountsResp		
AuthorizedAmount	38.37	
PaymentAcquirerData		
AcquirerID	690001	
MerchantID	mer77-130209	
AcquirerPOIID	963276433	
ApprovalCode	6541	
LoyaltyResult		
LoyaltyAccount		
LoyaltyAccountID		
EntryMode	MagStripe	
IdentificationType	PAN	
IdentificationSupport	LinkedCard	
LoyaltyID	6900014570541535	
LoyaltyBrand	FleetBonus	
Rebates		
SaleItemRebate		
ItemID	1	
ProductCode	673	
ItemAmount	0.15	
RebateLabel	Special promotion	

c) Loyalty Award Refund

The Sale Terminal requests the award earned on a payment of 48.19 Euros.

MessageHeader

MessageClass	Service
MessageCategory	Loyalty
MessageType	Request
ServiceID	666
SaleID	SaleTermA
POIID	POITerm1

(message example 25)

LoyaltyRequest

SaleData

SaleTransactionID

TransactionID	582
TimeStamp	2009-08-09T18:33:21.9+01:00

LoyaltyTransaction

LoyaltyTransactionType	Award
Currency	EUR
TotalAmount	48.19

The loyalty account earns 5 points, involving a balance of 42 points on the account.

MessageHeader

(message example 26)

MessageClass	Service
MessageCategory	Loyalty
MessageType	Response
ServiceID	666
SaleID	SaleTermA
POIID	POITerm1

LoyaltyResponse

Response

Result	Success
--------	---------

SaleData

SaleTransactionID

TransactionID	582
TimeStamp	2009-08-09T18:33:21.9+01:00

POIData

POITransactionID	
TransactionID	491
TimeStamp	2009-08-09T18:33:52.4+01:00
POIReconciliationID	200903101

LoyaltyResult

LoyaltyAccount

LoyaltyAccountID	
EntryMode	Scanned
IdentificationType	AccountNumber
IdentificationSupport	NoCard
LoyaltyID	201401

LoyaltyBrand	MerBonus
--------------	----------

CurrentBalance	42
----------------	----

LoyaltyAmount

AmountValue	5
-------------	---

The Sale Terminal requests the refund of the previous transaction.

MessageHeader		(message example 27)
MessageClass	Service	
MessageCategory	Loyalty	
MessageType	Request	
ServiceID	683	
SaleID	SaleTermA	
POIID	POITerm1	
LoyaltyRequest		
SaleData		
SaleTransactionID		
TransactionID	593	
TimeStamp	2009-08-09T18:35:44.9+01:00	
LoyaltyTransaction		
LoyaltyTransactionType	AwardRefund	
OriginalPOITransaction		
POITransactionID		
TransactionID	491	
TimeStamp	2009-08-09T18:33:52.4+01:00	
ReuseCardDataFlag	True	
LoyaltyData		
LoyaltyAmount		
AmountValue	5	

The loyalty account earns 5 points, involving a balance of 42 points on the account.

MessageHeader		(message example 28)
MessageClass	Service	
MessageCategory	Loyalty	
MessageType	Response	
ServiceID	683	
SaleID	SaleTermA	
POIID	POITerm1	
LoyaltyResponse		
Response		
Result	Success	
SaleData		
SaleTransactionID		
TransactionID	593	
TimeStamp	2009-08-09T18:35:44.9+01:00	
POIData		
POITransactionID		
TransactionID	504	
TimeStamp	2009-08-09T18:36:52.2+01:00	
POIReconciliationID	200903101	
LoyaltyResult		
LoyaltyAccount		
LoyaltyAccountId		
EntryMode	Scanned	
IdentificationType	AccountNumber	
IdentificationSupport	NoCard	
LoyaltyID	201401	
LoyaltyBrand	MerBonus	
CurrentBalance	37	

LoyaltyAmount

AmountValue

5

4.3.5 Stored Value Messages

4.3.5.1 Presentation of the Messages

The Stored Value request message body *StoredValueRequest*, contains the following information:

1. The data structure *SaleData*, containing the context of the Sale Terminal specific to this payment request:
 - a. The information which has changed since or was absent from the last Login request (*OperatorID*, *OperatorLanguage*...).
 - b. The identification of the transaction for the Sale System *SaleTransactionID*, containing the identification completed by a time stamp to ensure the uniqueness of the identification.
 - c. Information not interpreted by the POI, for the POI log, the Acquirer or the Issuer.
2. *CustomerLanguage*, customer language if already chosen for the transaction by the Customer.
3. The repeated data structure *StoredValueData*, containing information on a stored value card to load or reload:
 - a. *StoredValueProvider*, to identify the host who provide the loading management.
 - b. *StoredValueTransType*, the type of stored value transaction or operation.
 - c. The data structure *StoredValueAccountID*, for each stored value card, providing the card entry mode, the stored value card/account identification, and the entry number in the sale item list (*EntryMode*, *IdentificationType*, *StoredValueID* and *ItemID*).
 - d. *OriginalPOITransact*, to link a sequence of stored value transactions or operations.
 - e. *ProductCode* and *EanUpc*, containing the code of stored value card product.
 - f. *ItemAmount*, the amount to load or reload in the stored value cards.
 - g. The currency *Currency*.

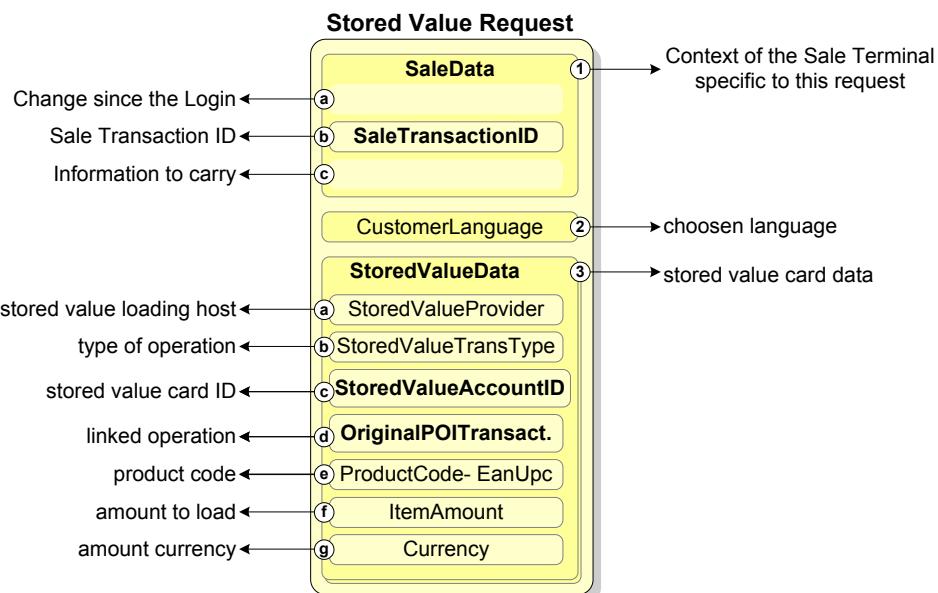


Figure 162: Stored Value Request Information

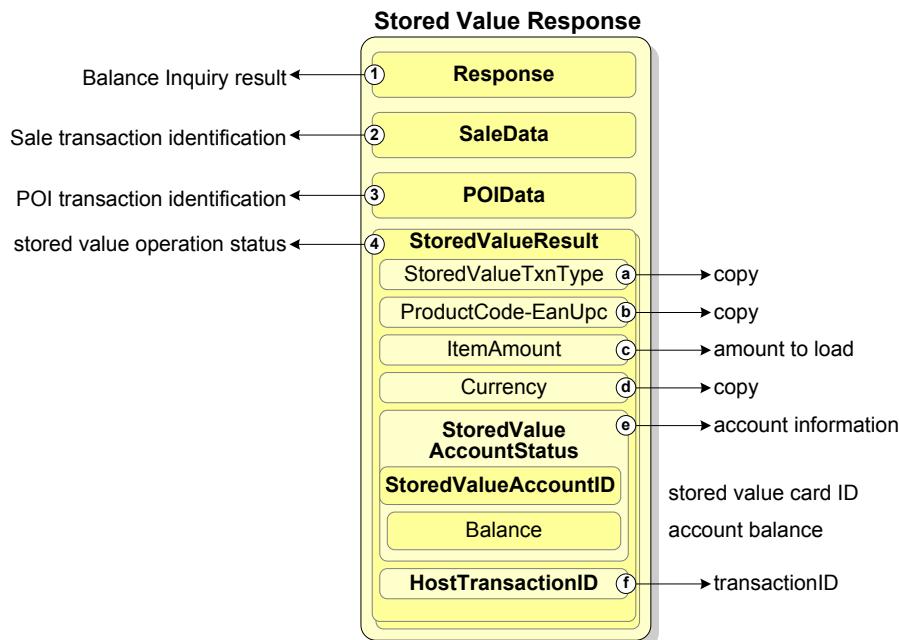


Figure 163: Stored Value Response Information

The Stored Value response message body *StoredValueResponse*, contains the following information:

1. The result of the Stored Value transaction: *Response*.
2. The data structure *SaleData*, containing the Sale transaction identification.
3. The data structure *POIData*, containing the POI transaction identification.
4. The repeated data structure *StoredValueResult*, containing the result of each stored value card which has been loaded or reloaded:
 - a. The copy of the *StoredValueTransactionType*,
 - b. The copy of the *ProductCode*,
 - c. The amount to pay *ItemAmount*, and
 - d. The *Currency*.
 - e. The data structure *StoredValueAccountStatus*, containing the result of each stored value card which has been loaded or reloaded:
 - The stored value card/account information: *StoredValueAccountID*, including the entry mode *EntryMode*, the type of card identifier/*IdentificationType*, the identifier *StoredValueID*.
 - The balance of the stored value account: *CurrentBalance*.
 - f. The identification of the Host transaction if any: *HostTransactionID*.

4.3.5.2 Stored Value Request Layout

<i>StoredValueRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		StoredValue
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
StoredValueRequest	[1..1]		
SaleData	[1..1]		<i>see PaymentRequest.SaleData</i>
OperatorID	[0..1]		
OperatorLanguage	[0..1]		
ShiftNumber	[0..1]		
SaleTransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
SaleTerminalData	[0..1]		
TerminalEnvironment	[0..1]		
SaleCapabilities	[0..1]		
SaleProfile	[0..1]		
TotalsGroupID	[0..1]		
CustomerLanguage	[0..1]		If the language is selected by the Sale System before the request to the POI.
StoredValueData	[1..n]		
StoredValueProvider	[0..1]		If more than one provider to manage on the POI, and StoredValueAccountID absent.
StoredValueTransactionType	[1..1]		
StoredValueAccountID	[0..1]		If the identification of the Stored Value account or card has been made by the Sale System before the request
StoredValueAccountType	[1..1]		
StoredValueProvider	[0..1]		If available for the card or account.
OwnerName	[0..1]		If available for the card or account.
ExpiryDate	[0..1]		If required for the card or account.
EntryMode	[1..1]		
IdentificationType	[1..1]		
StoredValueID	[1..1]		
OriginalPOITransaction	[0..1]		
POITransactionID	[0..1]		If HostTransactionID not present
TransactionID	[1..1]		
TimeStamp	[1..1]		
HostTransactionID	[0..1]		If POITransactionID not present
TransactionID	[1..1]		
TimeStamp	[0..1]		
POIID	[0..1]		If original transaction is coming from another POI

<i>StoredValueRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
ProductCode	[0..1]		
EanUpc	[0..1]		
ItemAmount	[0..1]		
Currency	[0..1]		

4.3.5.3 Stored Value Response Layout

<i>StoredValueResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		StoredValue
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
StoredValueResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>
SaleData	[1..1]		
SaleTransactionID	[1..1]		Copy
TransactionID	[1..1]		
TimeStamp	[1..1]		
POIData	[1..1]		
POITransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
POIReconciliationID	[1..1]		
StoredValueResult	[0..n]		If StoredValueResponse.Result is "Success" or "Partial", one entry per StoredValueRequest.StoredValueData loaded or activated
StoredValueTransactionType	[1..1]		Copy
ProductCode	[0..1]		Copy
EanUpc	[0..1]		Copy
ItemAmount	[0..1]		
Currency	[0..1]		Copy
StoredValueAccountStatus	[0..1]		if Result = "Success"
StoredValueAccountID	[1..1]		
StoredValueAccountType	[1..1]		
StoredValueProvider	[0..1]		<i>see PaymentResponse</i>
OwnerName	[0..1]		<i>see PaymentResponse</i>
ExpiryDate	[0..1]		<i>see PaymentResponse</i>
EntryMode	[1..1]		<i>see PaymentResponse</i>
IdentificationType	[1..1]		
StoredValueID	[1..1]		
CurrentBalance	[0..1]		if relevant and known
HostTransactionID	[0..1]		If provided by the Host
TransactionID	[1..1]		
TimeStamp	[0..1]		
PaymentReceipt	[0..*]		If Basic profile implementation with no printer on

<i>StoredValueResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
			the POI.

4.3.5.4 Stored Value Processing

Stored Value message pair allows loading and reloading of one or several stored value cards.

The request is processed locally on the card, or with the help of a dedicated host. The payment must be realised by a standard Payment message exchange, or outside of the POI System.

Rule 1: The stored value card could be either anonymous, or attached to an account identified by *StoredValueAccountID*, which is provided by the Sale Terminal in the request, or by the POI in the response.

Rule 2: The Stored Value request message must contain a *StoredValueData* per card:

- The operation to perform on the card is identified by *StoredValueTransactionType*,
- The amount to load or unload is notified by *ItemAmount* and *Currency*,
- The type of card may be identified by *StoredValueProvider* and the *ProductCode*.

Rule 3: The data element *Result*, in the Stored Value response message contains the value

- "Success" if all the loading or unloading requested in the *StoredValueData* are successful,
- "Failure", if none of the loading or unloading requested in the *StoredValueData* are successful.
- "Partial" in the other cases, at least one successful, at least one fails.

Rule 4: If the Stored Value response is not received by the Sale System for some reason, the Sale System must follow the error general error resolution (see section 4.7.5.1 *Error Resolutions Specifications*):

4.3.5.5 Error Cases

When the Stored Value request is successfully processed, the Stored Value response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

UnavailableService

The Stored Value financial service is not available in the POI System (see section 4.6.4.3.3 *Unavailable Administrative Service*).

DeviceOut

The Stored Value financial service couldn't be provided by the POI System, e.g. the communication to the Host is out of order (see section 4.6.3.1 *DeviceOut Error*).

Cancel

The user has aborted the transaction on the Customer interface (e.g. the POI Terminal keyboard), because he does not want to continue the transaction, see section 4.6.6.2.1 *User Cancellation*.

UnreachableHost

The Host is unreachable or has not answered to an online request, so it is considered as temporary unavailable. The Cashier has not forced the transaction, and the payment cannot be accepted (see section 4.6.8.1.1 *Host Unreachable* and section 4.6.8.1.2 *No Host Answer*).

Refusal

The transaction is refused by the Host. A specific message is normally displayed to the Customer and the Cashier if they are presents, the information below could be logged for further information (see section 4.6.8.2 *Refusal Error*).

4.3.5.6 Example

MessageHeader	<i>(message example 29)</i>	
MessageClass	Service	
MessageCategory	StoredValue	
MessageType	Request	
ServiceID	630	
SaleID	SaleTermA	
POIID	POITerm1	
StoredValueRequest		
SaleData		
SaleTransactionID		
TransactionID	580	
TimeStamp	2009-07-22T23:08:42.4+01:00	
StoredValueData		
StoredValueTransactionType	Load	
ProductCode	512	
ItemAmount	14.99	
Currency	EUR	
MessageHeader	<i>(message example 30)</i>	
MessageClass	Service	
MessageCategory	StoredValue	
MessageType	Response	
ServiceID	630	
SaleID	SaleTermA	
POIID	POITerm1	
StoredValueResponse		
Response		
Result	Success	
SaleData		
SaleTransactionID		
TransactionID	580	
TimeStamp	2009-07-22T23:08:42.4+01:00	
POIData		
POITransactionID		
TransactionID	481	
TimeStamp	2009-07-22T23:09:05.1+01:00	
POIReconciliationID	200903101	
StoredValueResult		
StoredValueTransactionType	Load	
ProductCode	512	
ItemAmount	14.99	
Currency	EUR	
StoredValueAccountStatus		
StoredValueAccountID		
StoredValueAccountType	GiftCard	
EntryMode	MagStripe	
IdentificationType	AccountNumber	
StoredValueID	8678252755678	
CurrentBalance	14.99	

4.3.6 Reversal Messages

4.3.6.1 Overview of Abort, Reversal and Refund

This section presents the main steps that influence the methods to cancel a payment transaction for the Sale system, the POI system, and the mechanisms provided by the protocol.

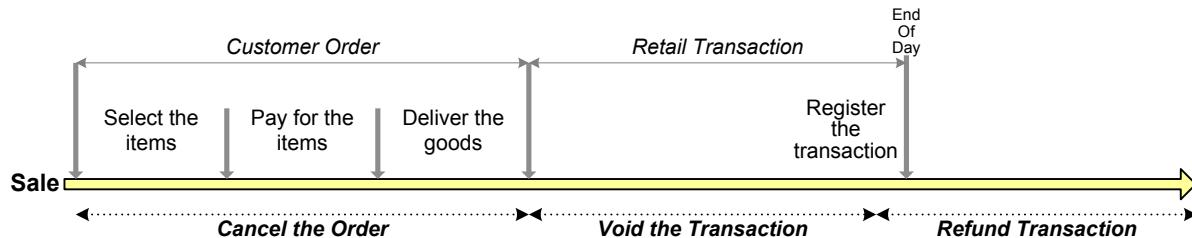


Figure 164: Sale System Payment Cancellation

After selecting the items of the purchase, the Sale system:

- Perform the payment of the purchase that can include several payments or loyalty transactions,
- Deliver the goods, last step of the customer order where the cancellation of the order involves the annulation of the payment,
- Register the transaction, typically at the end of the day, but it can be realized in real-time. Until the retail transaction is stored in the book, the transaction can be voided and annuls the payment,
- After registration of the retail transaction (typically at the end of day), the payment can only be voided by a refund transaction, which requires a new retail transaction.

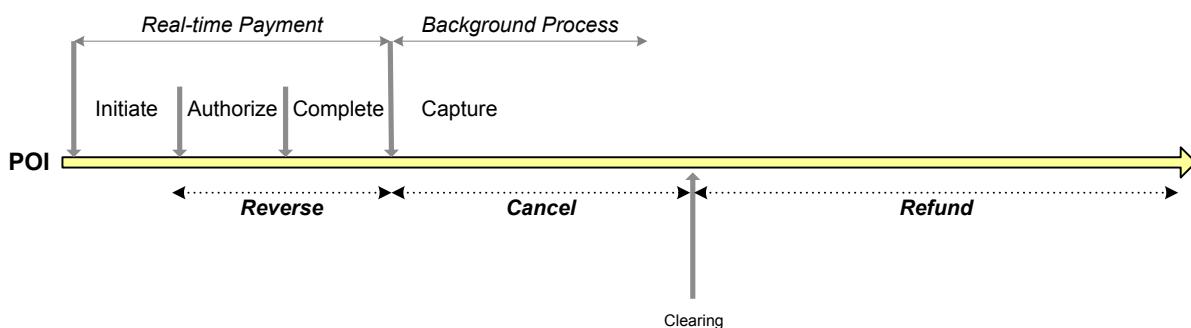


Figure 165: POI System Payment Cancellation

After receiving the payment request from the Sale, the POI:

- Initiate the payment including the selection of the payment instrument,
- Realize the authorization process online or offline,
- Complete the payment transaction. Until the real-time payment completion, the payment transaction can be voided by a reversal either automatically (e.g. timeout on the authorization response), or by the Customer or the Sale System.
- The financial data are captured by the Acquirer, and the POI can cancel the payment until the clearing of the payment transaction by the Acquirer.
- After the clearing of the transaction by the Acquirer, the payment transaction can only be voided by a refund transaction, which requires a new payment transaction.

The Sale to POI protocol provides to the Sale system three messages for payment annulation:

1. The Abort message (see section 4.7.2.3 page 479) to terminate the payment when the Sale system knows that the payment transaction is still in progress. This part of the General Error Resolution of the Sale system described in the section 4.7.5.2 page 499.
2. The Reversal messages exchange (see section 4.3.6.5 page 305) to cancel a previous completed payment at the POI.
3. The Refund message exchange (see section 4.3.2.11 page 240) to credit a Cardholder account from the Merchant account, when an Abort or a Reversal is no more possible.

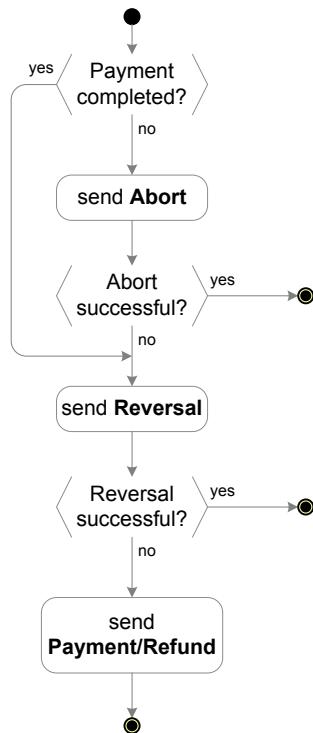


Figure 166: Sale Payment Annulation

The process at the Sale system for a payment transaction is described in the process flow above:

- If the payment transaction is still in progress, because the Payment response message is not received and a TransactionStatus message response contains *Result*=“Failure” and *ErrorCondition*=“InProgress”, an Abort request has to be sent.
- If the payment transaction is completed or the Abort does not succeed (Payment response does not contain *ErrorCondition*= “Aborted”), a Reversal request message has to be sent.
- If the Reversal response message indicates a failure, a refund transaction has to be initiated (Payment request message with *PaymentType*= “Refund”).

4.3.6.2 Presentation of the Messages

The Reversal request message body *ReversalRequest*, contains the following information:

1. *SaleReferenceID*, when the transaction is a reservation to identify globally the transaction.
2. The data structure *OriginalPOITransaction*, containing identification of transaction to reverse:
 - a. *SaleID*, if the reversal is sent from another Sale Terminal.
 - b. *POIID*, if the reversal is sent to another POI Terminal.
 - c. *POITransactionID*, the identification of the transaction to reverse, or *AmountValue*, the amount of the transaction to reverse.
3. *ReversalReason*, providing the cause of the reversal.
4. *ReversedAmount*, for partial reversal

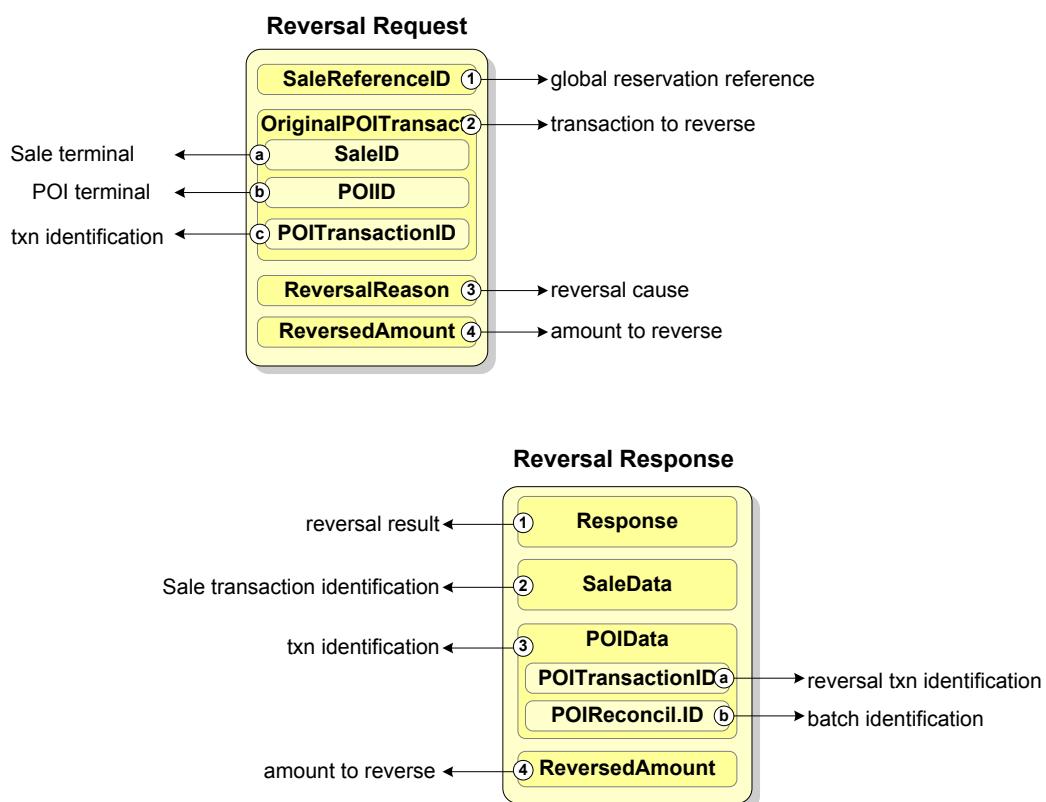


Figure 167: Reversal Request/Response Information

The Reversal response message body *ReversalResponse*, contains the following information:

1. *Response*, the result of the reversal.
2. The data structure *SaleData*, containing the Sale transaction identification.
3. The data structure *POIData*, containing identification of reversed transaction:
 - a. *POITransactionID*, identification of the reversal transaction.
 - b. *POIRconciliationID*, the related reconciliation period identification.
4. The reversed amount.

4.3.6.3 Reversal Request Layout

<i>ReversalRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Reversal
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
ReversalRequest	[1..1]		
SaleData	[0..1]		
OperatorID	[0..1]		if different from the Login and see <i>Login.SaleData</i>
OperatorLanguage	[0..1]		if different from the Login
ShiftNumber	[0..1]		if different from the Login and see <i>Login.SaleData</i>
SaleTransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
SaleReferenceID	[0..1]	R	Mandatory if payment reservation or if CustomerOrder is present
SaleTerminalData	[0..1]		If content is not empty
TerminalEnvironment	[0..1]		If modified since the login
SaleCapabilities	[0..1]		If modified since the login (devices failures)
TotalsGroupID	[0..1]		if modified since, or not in the login and used by the Sale System
TokenRequestedType	[0..1]		If a token is requested.
CustomerOrderReq	[0..1]		If customer orders must be listed in the response message.
SaleToPOIData	[0..1]		Stored with the transaction
SaleToAcquirerData	[0..1]		Send to the Acquirer if present
SaleToIssuerData	[0..1]		Send to the Acquirer if present
StatementReference	[0..1]		Information to print on the bank statement
OriginalPOITransaction	[1..1]		
SaleID	[0..1]		default MessageHeader.SaleID
POIID	[0..1]		default MessageHeader.POID
POITransactionID	[0..1]		Identification of the Loyalty/Payment to reverse
TransactionID	[1..1]		
TimeStamp	[1..1]		
CustomerLanguage	[0..1]		Customer language of the original transaction, if not the POI default language.
AcquirerID	[0..1]		If several Acquirers, acquirer used for the original transaction.
AmountValue	[0..1]		Present only if POITransactionID absent.
ReversedAmount	[0..1]		ReversedAmount is implicitly the AuthorizedAmount if absent.

<i>ReversalRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
ReversalReason	[1..1]		
CustomerOrder	[0..1]		If related to a customer order.
CustomerOrderID	[0..1]		
StartDate	[1..1]		
ForecastedAmount	[1..1]		
OpenOrderState	[0..1]		True by default
Currency	[0..1]		If multiple currencies are allowed.
AdditionalInformation	[0..1]		

4.3.6.4 Reversal Response Layout

<i>ReversalResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Reversal
MessageType	[1..1]		Response
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
ReversalResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>same as PaymentResponse</i>
POIData	[0..1]		If Result is Success
POITransactionID	[1..1]		Identification of the reversal transaction for the POI
TransactionID	[1..1]		
TimeStamp	[1..1]		
POIReconciliationID	[0..1]		If Result is Success
OriginalPOITransaction	[0..1]		Present if POITransactionID absent in the request, otherwise absent.
SaleID	[0..1]		default MessageHeader.SaleID
POIID	[0..1]		default MessageHeader.POIID
POITransactionID	[0..1]		Identification of the original Loyalty/Payment transaction.
TransactionID	[1..1]		
TimeStamp	[1..1]		
ReversedAmount	[0..1]		Copy if present in the request
CustomerOrder	[0..n]		If the payment is related to a customer order in progress or if the list of customer orders has been requested.
CustomerOrderID	[0..1]		
SaleReferenceId	[1..1]		
OpenOrderState	[0..1]		default "True"
StartDate	[1..1]		
EndDate	[0..1]		If OpenOrderState = "False".
ForecastedAmount	[1..1]		
CurrentAmount	[1..1]		
Currency	[0..1]		If multiple currencies are allowed.
AccessedBy	[0..1]		If order process in progress.
AdditionalInformation	[0..1]		
PaymentReceipt	[0..*]		If Basic profile implementation with no printer on the POI.
DocumentQualifier	[1..1]		SaleReceipt or CashierReceipt
IntegratedPrintFlag	[0..1]		<i>same as PrintRequest</i>
RequiredSignatureFlag	[0..1]		default False.

<i>ReversalResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
OutputContent	[1..1]		
OutputFormat	[1..1]		Text, XHTML
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		<i>same as Display</i>
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
EndOfLineFlag	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>

4.3.6.5 Reversal Processing

Reversal is the function initiated by the Sale System, to cancel a previous completed payment at the POI e.g. cardholder changes his mind and uses another payment mean, or the end of the transaction fails at the Sale System. Typically a problem of delivery involves a partial or complete reversal of the original transaction.

- Rule 1: Reversal is not part of the protocol error resolution. This is a way to cancel a previous payment or loyalty transaction, completed on the POI System.
The transaction which is reversed could be any payment or loyalty service, including a refund, but not a reversal.
- Rule 2: When the reversal uses a different Sale terminal (resp. POI terminal) than the transaction to reverse, the Reversal request message shall contain the data element *OriginalPOITransaction.SaleID* (resp. *OriginalPOITransaction.POIID*) with the identification of the Sale terminal (resp. POI terminal) which has performed the original transaction.
- Rule 3: If the reversal is not allowed for the card used in the original transaction, the *Response* component of the Reversal response must have the value “Failure”-“UnAvailable” (see section 4.6.4.3.2 *Unavailable Service for the Card*).
- Rule 4: If the original transaction is not found, the *Response* component of the Reversal response must have the value “Failure”-“NotFound” (see section 4.6.4.4.1 *Transaction Not Found*).
- Rule 5: A reservation reversal globally cancels the transaction reservation which is then completed. The reversal may be requested at any status of the transaction, and *SaleReferenceID* is mandatory.
- Rule 6: If the transaction cannot be reversed (e.g. it is too late), the *Response* component of the Reversal response must have the value “Failure”-“Refusal” (see section 4.6.8.2 *Refusal Error*).
- Rule 7: Partial reversal:
If the Reversal request does not contain the data element *ReversedAmount*, the amount to reverse is implicitly the *AuthorizedAmount* of the original transaction identified in *OriginalPOITransaction*.
If the Reversal request contains the data element *ReversedAmount* with a value greater than the data element *AuthorizedAmount* of the original transaction, the *Response* component of the Reversal response must have the value “Failure”-“NotFound” (see section 4.6.4.4.1 *Transaction Not Found*).
If the reversal is partial, Reversal request must contain *ReversedAmount*, with a value lower than the *AuthorizedAmount* of the original transaction. If partial reversal is not allowed by the Acquirer, the *Response* component of the Reversal response must have the value “Failure”-“Unavailable Service” (see section 4.6.4.3.2 *Unavailable Service for the Card*).

In some country, a typical case of partial reversal is when the delivery of goods is partially completed, for instance, in a vending machine.

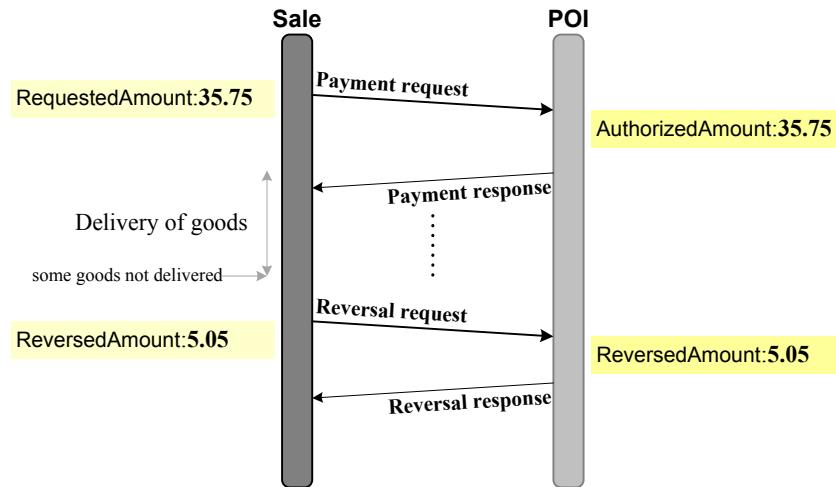


Figure 168: Partial Reversal Example

Rule 8: Reversal without original transaction identification:

When the manual reversal is performed some time after the original transaction, and the Sale system is not able to search the original POI transaction, to avoid typo entering the identification of the transaction, the POI must search the transaction with the following steps:

- 1) The Sale terminal send a Reversal request to the POI with the identification of the original transaction *POITransactionID* absent and the amount of the original transaction *AmountValue* present.
- 2) The POI reads the card used by the original transaction, and searches the last payment performed transaction with the amount and the entered card. If no transaction is found, the POI returns a Reversal response with *Response* "Failure"-“NotFound” (see section 4.6.4.4.1 *Transaction Not Found*).
- 3) The POI terminal sends an Input request to ask the cashier if the sale transaction identification found is correct (additional information of the transaction could be also verified). If the identification is incorrect, the POI returns a Reversal response with *Response* "Failure" “NotFound” (see section 4.6.4.4.1 *Transaction Not Found*).
- 4) The POI performs the reversal.
- 5) The POI sends in the response the *POITransactionID* of the original transaction to allow the Sale system to retrieve the transaction that has been reversed.
SaleID (respectively *POIID*) is present if the reversal is performed on a different Sale terminal (respectivement POI terminal).

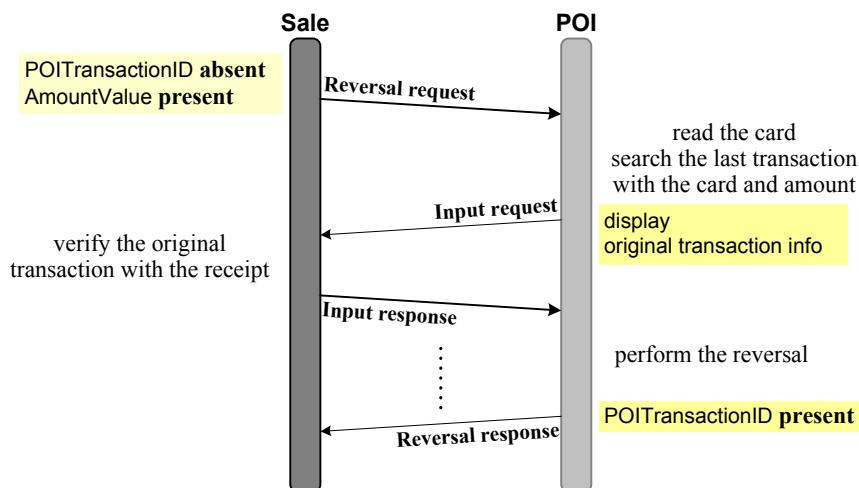


Figure 169: Reversal without Identification

4.3.6.6 Error Cases

When the Reversal request is successfully processed, the Reversal response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved before to try another Payment attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The Payment request is received during a Device dialogue or another Service dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

UnavailableService

The reversal financial service is not available for this card (see section 4.6.4.3.2 *Unavailable Service for the Card*).

DeviceOut

The POI element cannot start the payment transaction, because of an error on a device (e.g. printer without paper), see section 4.6.3.1.1 *POI Temporary Unavailable* or section 4.6.3.1.2 *POI Permanently Unavailable*.

NotFound

The original transaction is not found by the receiver of the message. (see section 4.6.4.4.2 *Message Not Found*).

UnreachableHost

The Acquirer is unreachable or has not answered to an online request, so it is considered as temporary unavailable. (see section 4.6.8.1.1 *Host Unreachable* and section 4.6.8.1.2 *No Host Answer*).

Refusal

The reversal is refused by the payment Acquirer or the rules associated to the card. A specific message could be displayed to the Customer and the Cashier if they are presents, the information below could be logged for further information (see section 4.6.8.2 *Refusal Error*).

4.3.6.7 Example

The Sale Terminal requests a payment.

MessageHeader		<i>(message example 31)</i>
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	633	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	582	
TimeStamp	2009-08-09T20:06:33.1+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	38.01	
TransactionConditions		
LoyaltyHandling	Require	
PaymentData		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
EntryMode	Scanned	
SensitiveCardData		
PAN	7011014570541535	

MessageHeader		<i>(message example 32)</i>
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	633	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentResponse		
Response		
Result	Success	
SaleData		
SaleTransactionID		
TransactionID	582	
TimeStamp	2009-08-09T20:06:33.1+01:00	
POIData		
POITransactionID		
TransactionID	492	
TimeStamp	2009-08-09T20:33:52.0+01:00	
POIReconciliationID	200903101	
PaymentResult		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	MerRel	

EntryMode	MagStripe
SensitiveCardData	
PAN	7011014570541535
AmountsResp	
AuthorizedAmount	68.95
PaymentAcquirerData	
MerchantID	mer77-130209
AcquirerPOIID	65-POITerm1

The payment is reversed from another Sale Terminal, for instance at a central cash register, to the POI Server.

MessageHeader *(message example 33)*

MessageClass	Service
MessageCategory	Reversal
MessageType	Request
ServiceID	15
SaleID	SaleTermB
POIID	POIServer

ReversalRequest**OriginalPOITransaction**

SaleID	SaleTermA
POIID	POITerm1
POITransactionID	
TransactionID	492
TimeStamp	2009-08-09T20:33:52.0+01:00
ReversalReason	CustCancel

MessageHeader*(message example 34)*

MessageClass	Service
MessageCategory	Reversal
MessageType	Response
ServiceID	15
SaleID	SaleTermB
POIID	POIServer

ReversalResponse**Response**

Result	Success
--------	---------

POIData

POITransactionID	
TransactionID	178
TimeStamp	2009-08-09T21:19:15.3+01:00

4.3.7 Batch Messages

4.3.7.1 Presentation of the Batch Messages

The Batch request message body *BatchRequest*, contains either:

1. A flag *RemoveAllFlag* requesting to remove the transactions stored and not yet performed.
2. The data structure *PaymentRequest*, containing the same information as the Payment request message, or
3. The data structure *LoyaltyRequest*, containing the same information as the Loyalty request message, or
4. The data structure *ReversalRequest*, containing the same information as the Reversal request message.

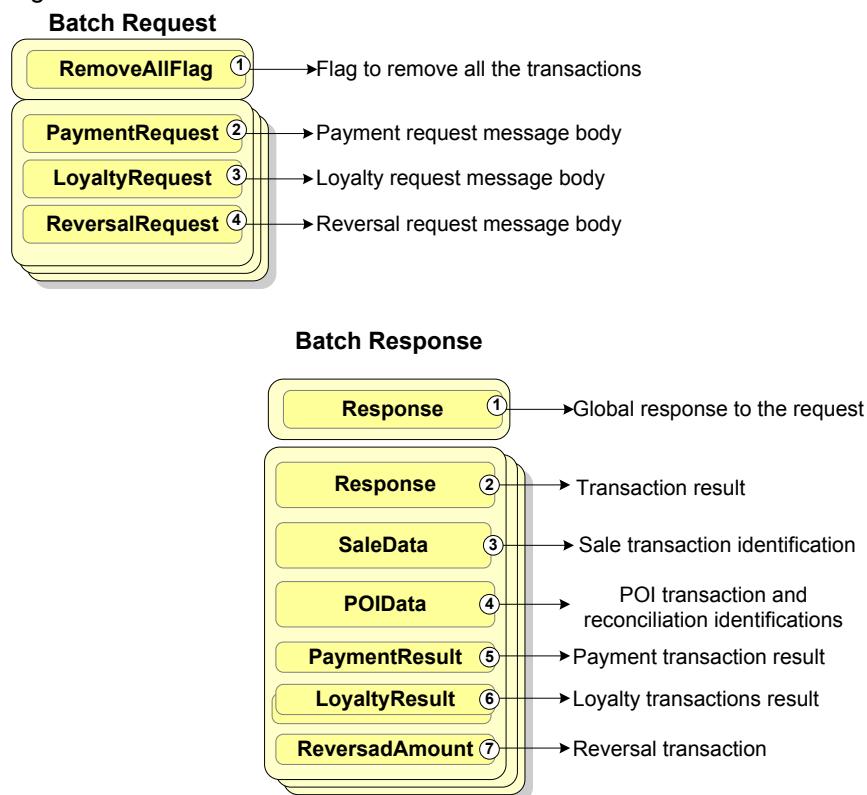


Figure 170: Batch Information

The Batch response message body *BatchResponse* contains:

1. The global result of the request, and optionally a collection of performed transactions including:
2. The result of the transaction: *Response*.
3. The data structure *SaleData*, containing the Sale transaction identification, if a request has been made by the Sale system.
4. The data structure *POIData*, containing the POI transaction identification and the reconciliation identification.
5. The result of a payment transaction: *PaymentResult*.
6. The result of one or several payment or loyalty transactions, alone or synchronised with the payment transaction: *LoyaltyResult*.
7. The *ReversedAmount* if a transaction Reversal has been performed.

4.3.7.2 Batch Request Layout

Batch Request Component	Mult.	Profile	Rule
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Batch
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
BatchRequest	[1..1]		
RemoveAllFlag	[0..1]		default False
TransactionToPerform	[0..n]		For each occurrence, exactly one of the <i>PaymentRequest</i> , <i>LoyaltyRequest</i> , or <i>ReversalRequest</i> may be present.
PaymentRequest	[0..1]		
LoyaltyRequest	[0..1]		
ReversalRequest	[0..1]		

4.3.7.3 Batch Response Layout

<i>Batch Response Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Batch
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
BatchResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>
PerformedTransaction	[0..n]		One occurrence per Payment, Loyalty without Payment, or Reversal transaction.
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>
SaleData	[0..1]		If the transaction request has been generated by the Sale system.
SaleTransactionID	[1..1]		Copy
TransactionID	[1..1]		
TimeStamp	[1..1]		
POIData	[1..1]		
POITransactionID	[1..1]		
TransactionID	[1..1]		
TimeStamp	[1..1]		
POIReconciliationID	[1..1]		
PaymentResult	[0..1]		If a Payment transaction has been performed and one (or several) data element is present
LoyaltyResult	[0..n]	L	If a Loyalty transaction has been performed alone or with the Payment transaction.
ReversedAmount	[0..1]		If a transaction Reversal has been performed.

4.3.7.4 Batch Processing

The *Batch* message pair is used to request or get the result of transactions (payment, loyalty and reversal) performed without connection to the Sale system. The *Batch* message pair may be used in two different ways:

- 1) To retrieve the transactions performed by the POI when the Sale system is not connected (e.g. payment of delivery).

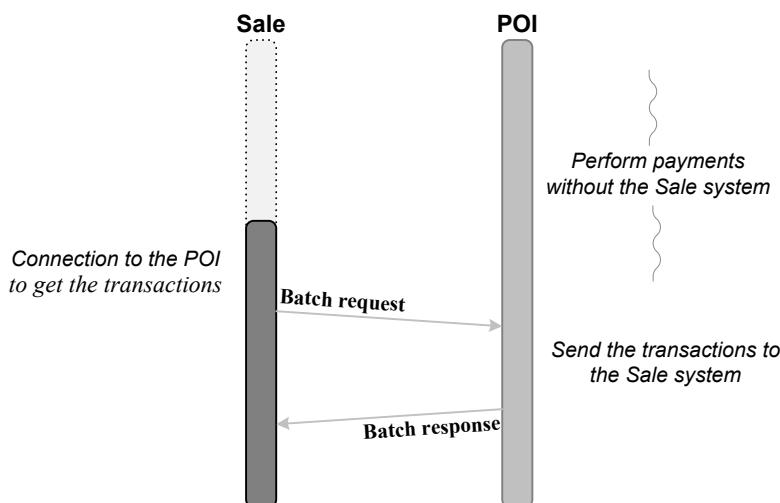


Figure 171: Retrieval of Transactions Performed without Sale System

- 2) To send transactions to perform later without the Sale system, and retrieve these transactions after their completion.

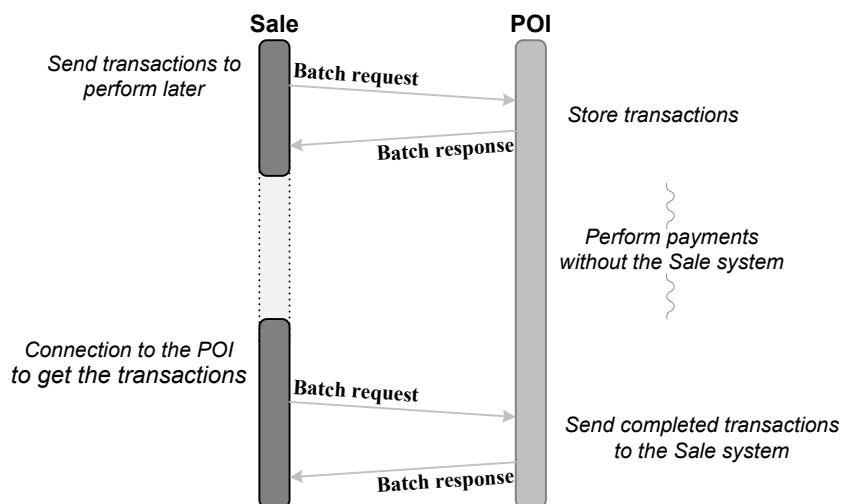


Figure 172: Sending of Transactions to be Performed Later

Rule 1: To request the processing of a collection of payment, loyalty and reversal transactions:

- The Sale sends a Batch request message containing a sequence of *TransactionToPerform*.
- The POI then returns a Batch response message containing no occurrence of *PerformedTransaction*, but the result of the parsing and storing of requests in the data element *Response*.

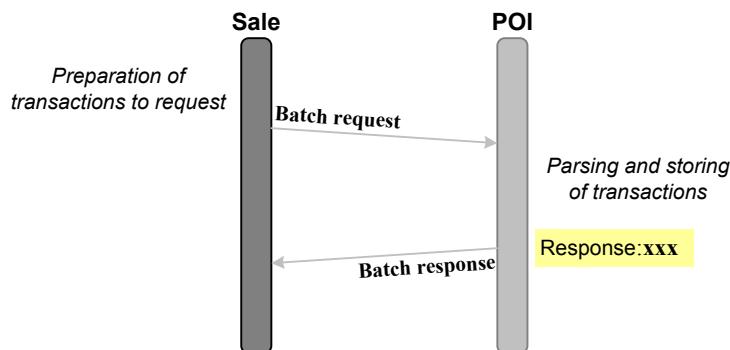


Figure 173: Batch Request of Transaction Processing

Rule 2: To acquire the result of the processing of performed payment transactions, loyalty transactions and reversal transactions:

- The Sale sends a Batch request message containing only one occurrence of *BatchRequest* which contains no occurrence of *TransactionToPerform*.
 - If *RemoveAllFlag* has the value "True", the transactions not yet performed are not returned in the response and are removed from the POI.
 - If *RemoveAllFlag* has the value "False", the transactions not yet performed are not returned in the response and stay in the POI to be performed later.
- The POI returns a Batch response message containing one occurrence of *PerformedTransaction* per completed transaction.

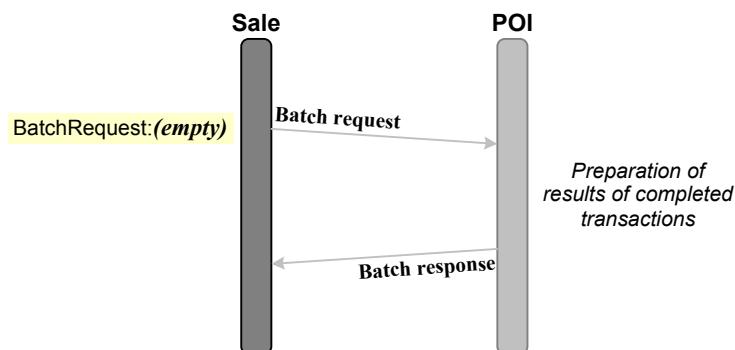


Figure 174: Batch Request of Transaction Results

Rule 3: The Batch request message contains either:

- No occurrence of *TransactionToPerform*, or
- One or several occurrences of *TransactionToPerform* containing exactly one and only one data element among:
 - *PaymentRequest*,
 - *LoyaltyRequest*, or
 - *ReversalRequest*.

Rule 4: *TransactionToPerform* contains *LoyaltyRequest* if a loyalty transaction has to be performed alone, i.e. without any payment synchronisation.

Rule 5: The response message *BatchResponse* contains either:

- No occurrence of *PerformedTransaction*, but only *Response* (see Rule 1), or
- One or several occurrences of *PerformedTransaction*.

Rule 6: For an occurrence of *PerformedTransaction* in the Batch response message:

- *SaleData* is present only when the transaction is the result of a request that has been generated by the Sale system.
- *POIData* is mandatory.
- If this occurrence is the result of a successful payment transaction:
 - *PaymentResult* is mandatory,
 - *LoyaltyResult* is present if loyalty card has been used for the payment transaction, and
 - *ReversedAmount* must be absent.
- If this occurrence is the result of a successful loyalty transaction:
 - *PaymentResult* and *ReversedAmount* must be absents, and
 - One occurrence of *LoyaltyResult* is mandatory.
- If this occurrence is the result of a successful reversal transaction:
 - *PaymentResult* and *LoyaltyResult* must be absents, and
 - *ReversedAmount* is mandatory.

Rule 7: If the Sale requests the processing of a collection of transactions, the result of completed transaction requested later may contains only a subset of these requested transactions. When the Sale system request the result of completed transaction, the Batch response might contains in the same message transactions:

- Coming from several Batch request messages,
- Not requested by the Sale system, but performed by the POI without connection to the Sale system.

However, as requested in the following rule, all the transaction stored in the POI must be requested by the Sale system.

Rule 8: Several Batch pair might be exchanged during a reconciliation period.

Before to send a ReconciliationRequest message, the Sale system needs to ask the completed transactions and removing transaction not yet performed, sending a Batch request message with *RemoveAllFlag* set to "True".

Transactions not yet performed are not counted by the Sale or the POI in the totals of the Reconciliation or Get Totals message.

Rule 9: Batch error handling:

- When sending performed transaction in a Batch request, the error management is the same as for the Payment request, using the Transaction Status message and eventually the Abort.
- When requesting the transactions to perform to the POI in a Batch request, the request is idempotent and could be sent again.

4.3.7.5 Example

The following example provides a Batch messages exchange to retrieve the transactions made by the POI terminal during a delivery.

These payments transaction have not been stored before in the POI terminal by a Batch messages exchange.

MessageHeader		(message example 35)
MessageClass	Service	
MessageCategory	Batch	
MessageType	Request	
ServiceID	670	
SaleID	SaleTermA	
POIID	POITerm1	

BatchRequest

MessageHeader		(message example 36)
MessageClass	Service	
MessageCategory	Batch	
MessageType	Response	
ServiceID	670	
SaleID	SaleTermA	
POIID	POITerm1	

BatchResponse	
Response	
Result	Success

PerformedTransaction	
Response	
Result	Success
POIData	
POITransactionID	
TransactionID	481
TimeStamp	2014-06-22T23:08:42.4+01:00
POIReconciliationID	201406221
PaymentResult	
PaymentType	Normal
PaymentInstrumentData	
PaymentInstrumentType	Card
CardData	
PaymentBrand	CardPlus
EntryMode	MagStripe
SensitiveCardData	
PAN	0011014570541535
ExpiryDate	0417
AmountsResp	
AuthorizedAmount	104.11
PaymentAcquirerData	
AcquirerID	400012
MerchantID	mer77-130209
AcquirerPOIID	963276433
ApprovalCode	8347
PerformedTransaction	
Response	
Result	Partial

POIData**POITransactionID**

TransactionID	482
TimeStamp	2014-06-22T23:30:02.4+01:00
POIReconciliationID	201406221

PaymentResult

PaymentType	Normal
-------------	--------

PaymentInstrumentData

PaymentInstrumentType	Card
-----------------------	------

CardData

PaymentBrand	MerRel
--------------	--------

EntryMode	MagStripe
-----------	-----------

SensitiveCardData

PAN	7011014570541535
-----	------------------

AmountsResp

AuthorizedAmount	31.12
------------------	-------

TotalFeesAmount	0.19
-----------------	------

PaymentAcquirerData

MerchantID	mer77-130209
------------	--------------

AcquirerPOIID	65-POITerm1
---------------	-------------

4.3.8 Payment Use Cases

4.3.8.1 Pay at the Table

"Pay at the table" is a typical use case for the payment at a restaurant, with the following characteristics:

- There is a global sale transaction per table of the restaurant,
- The payment of the transaction is performed by any POI Terminal²²,
- The sale transaction is often paid by several payment means, including cash,
- The sale transaction is often paid by several people at the table, the payment is then split between several people,
- Tip is frequently offered to the waiter.
- The payment of a table may be split between people of the same table depending on their consumption.

The use of the Sale to POI protocol to realise such transaction is limited by the following constraints:

- Only the payment means managed by the protocol, values of *PaymentInstrumentType* which in particular includes "Cash", could be used for the transaction.
- Specific functions have to be implemented in the POI system to manage the "pay at the table" payment transaction.

The process of the "pay at the table" includes three successive steps:

1. The *Meal* at a table: this is a sale transaction performed for a "table".
2. The *Selection* of the POI and the Transaction: The waiter takes a POI terminal and type a reference to allow the Sale terminal to retrieve the sale transaction.
3. The *Payment* at the table: the payment is processed by the waiter at the table.

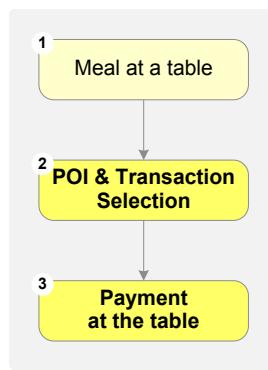


Figure 175: Pay at the Table Steps

²² Several POI terminals may be used for performing the payment of a sale transaction at the same table. The Sale system manages then the synchronisation of the split payments.

Rule 1: After taking a POI to perform the payment,

- The waiter types a reference (as the sale transaction reference or the table number) to ask the Sale system to retrieve the transaction.
- The POI then sends an Event message with the data element *EventToNotify* set to the value "SaleWakeUp", and the data element *EventDetails* set to the value of the reference typed by the waiter,
- The Sale terminal retrieves the transaction identified by *EventDetails*, and starts the payment of the transaction.

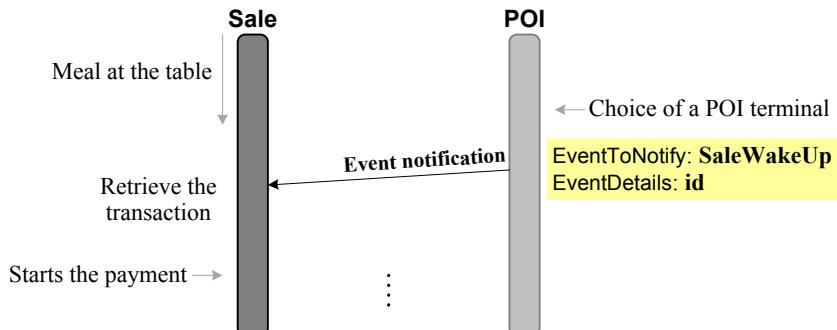


Figure 176: Transaction Selection to Pay at the Table

Rule 2: After receiving the Event notification, if the Sale terminal is unable to find the transaction, it sends a Display message to show the failure of the transaction selection. The Display message must have *ResponseRequiredFlag* absent or set to "False".
The waiter then has to type a new reference, and the application sends an Event notification with containing this new reference.

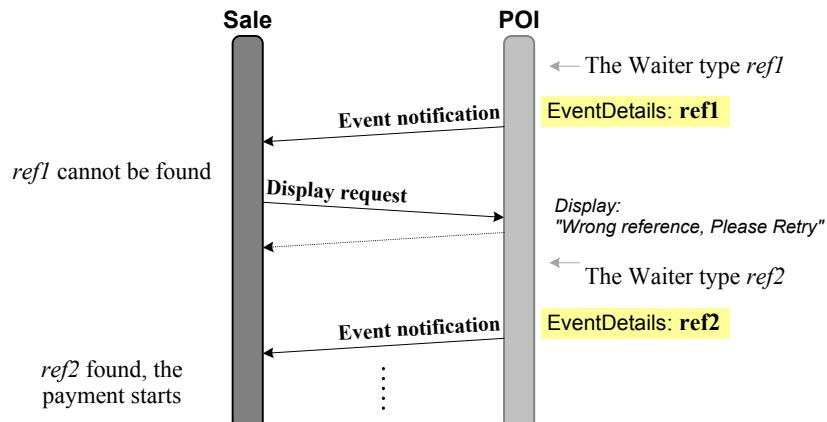


Figure 177: Invalid Transaction Selection to Pay at the Table

Rule 3: If the POI terminal is not logged to the Sale terminal, after the reception of the Event, the Sale terminal may send a Login request message.

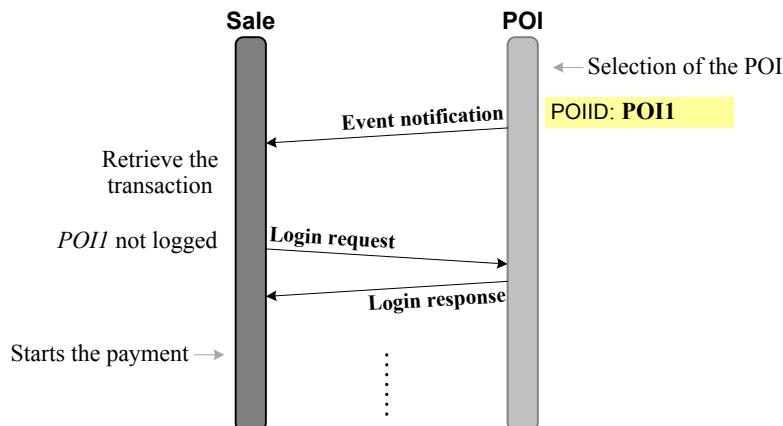


Figure 178: Login After Sale Wake-up

Rule 4: The payment at the table is initiated by the Sale terminal through a Payment request message.

The payment may be split between several customers at the table, or between several means of payment for the same customers as specified in the section 4.3.2.2 *Split Payment*.

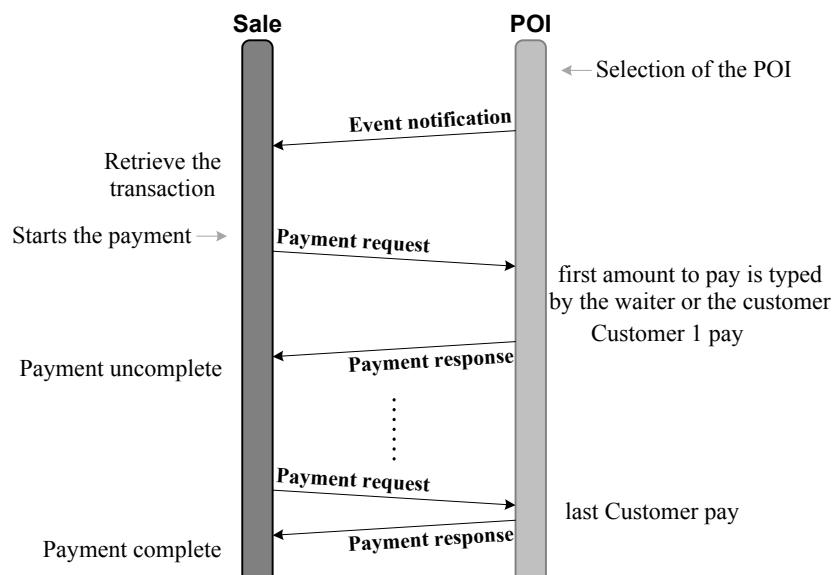


Figure 179: Use Case Pay-at-the-table

Rule 5: The payment at the table transaction is completed when the full requested amount (*RequestedAmount*) is paid.

If some payment means are not managed by the POI terminal (see *PaymentType* values²³), they are processed outside the POI system, and the transaction is completed when the *Result* has the value Success and *AuthorizedAmount* has a value 0.

²³ In particular, the cash could be managed by the POI terminal.

Rule 6: The amount paid by each customers may be computed on their consumption during the meal.

If the Sale Terminal is able to perform a sale transaction per customer, after the choice of the POI terminal, the waiter asks the transaction identifier for each customer, possibly with several terminals.

The different error cases could be occurs for each payment transaction.

4.4 Administrative Services

4.4.1 Reconciliation Messages

4.4.1.1 Presentation of the Messages

The Reconciliation request message body *ReconciliationRequest*, contains the following information:

1. The type of reconciliation which is requested: *ReconciliationType*, to allow synchronisation or not with an Acquirer, reconciliation with an Acquirer only, or the result of a previous reconciliation.
2. The identification of Acquirers *AcquirerID*, if some Acquirer reconciliations are requested.
3. The identification of the reconciliation period for the result of a previous reconciliation.

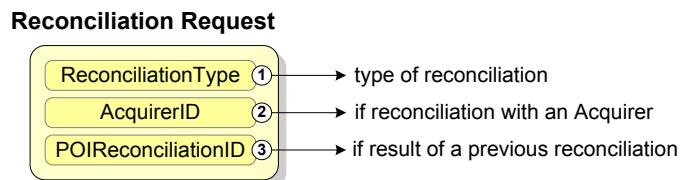


Figure 180: Reconciliation Request Information

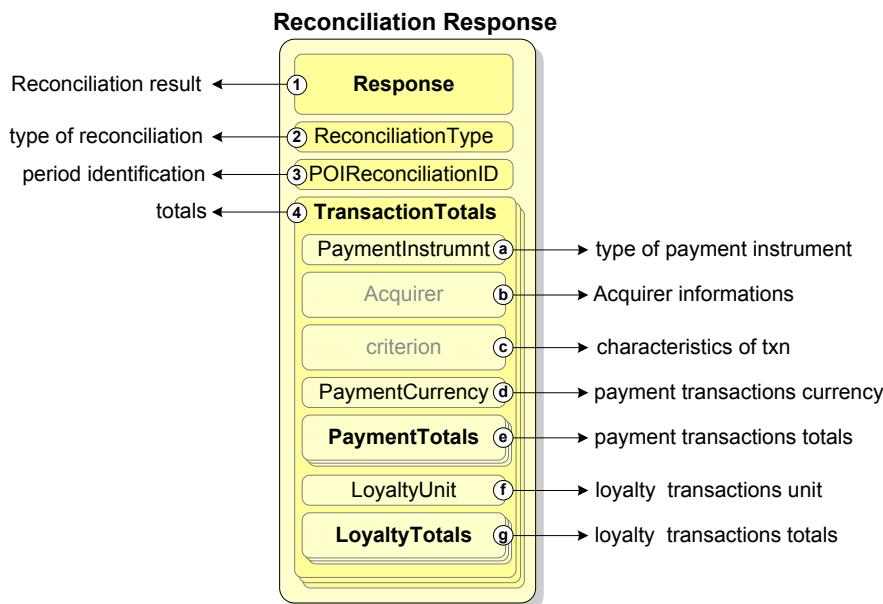


Figure 181: Reconciliation Response Information

The Reconciliation response message body *ReconciliationResponse*, contains the following information:

1. The result of the Reconciliation: *Response*.
2. The copy of the requested type of reconciliation: *ReconciliationType*.
3. The identification of the reconciliation period related to the provided totals: *POIReconciliationID*.
4. A sequence data structure containing totals of transactions computed with some criteria *TransactionTotals*:
 - a. The type of payment instrument used for these transactions: *PaymentInstrumentType*.
 - b. Information related to the Acquirer managing the brand: Acquirer identification *AcquirerID*, potential problem for Acquirer reconciliation *ErrorCondition*, and the identification of the Acquirer reconciliation *HostReconciliationID*.
 - c. Value of criterions used to organize the totals: POI Terminal *POIID*, Sale Terminal *SaleID*, Cashier *OperatorID*, shift *ShiftNumber*, and *TotalsGroupID*.
 - d. The payment currency: *PaymentCurrency*.
 - e. The totals of the payment transactions per type of transaction: *PaymentTotals*, containing the numbers of transaction *TransactionCount*, and the sum of the amounts *TransactionAmount*.
 - f. The unit of the loyalty transactions: *LoyaltyUnit* and *LoyaltyCurrency*.
 - g. The totals of the loyalty transaction per type of transactions: *LoyaltyTotals*, containing also the numbers of transaction *TransactionCount*, and the sum of the amounts *TransactionAmount*.

4.4.1.2 Reconciliation Request Layout

<i>ReconciliationRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Reconciliation
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
ReconciliationRequest	[1..1]		
ReconciliationType	[1..1]		
AcquirerID	[0..n]		Could be present only if ReconciliationType is "AcquirerReconciliation" or "AcquirerSynchronisation"
POIReconciliationID	[0..1]		Present if ReconciliationType is "PreviousReconciliation"

4.4.1.3 Reconciliation Response Layout

ReconciliationResponse Component	Mult.	Profile	Rule
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Reconciliation
MessageType	[1..1]		Response
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
ReconciliationResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>same as PaymentResponse</i>
ReconciliationType	[1..1]		Copy
POIReconciliationID	[0..1]		Absent if <i>ReconciliationType</i> is "AcquirerReconciliation"
TransactionTotals	[0..n]		if Response.Result is Success One set of totals per value of CardBrand and AcquirerID, ..., TotalsGroupID if presents
PaymentInstrumentType	[1..1]		
AcquirerID	[0..1]		If available
ErrorCondition	[0..1]		if Response.Result is Partial, and the reconciliation with this Acquirer failed.
HostReconciliationID	[0..1]		If available
CardBrand	[0..1]		If configured to present totals per card brand, and Response.Result is Success
POIID	[0..1]		If configured to present totals per POI terminals, or for the POI Terminal receiving the request.
SaleID	[0..1]		If configured to present totals per Sale terminals, or for the Sale Terminal receiving the request.
OperatorID	[0..1]		If configured to present totals per Cashier, or for the Cahier logged in the current session and identified in the Login message.
ShiftNumber	[0..1]		If configured to present totals per shift, or for the current shift identified in the Login message.
TotalsGroupID	[0..1]		If configured to present totals per TotalsGroupID
PaymentCurrency	[0..1]		
PaymentTotals	[0..10]		If If both TransactionCount and TransactionAmount are not equal to zero
TransactionType	[1..1]		Debit, Credit, ReverseDebit, ReverseCredit, OneTimeReservation, CompletedDeffered, FirstReservation, UpdateReservation, CompletedReservation, CashAdvance, IssuerInstalment, Failed, Declined
TransactionCount	[1..1]		
TransactionAmount	[1..1]		
LoyaltyUnit	[0..1]		default Point
LoyaltyCurrency	[0..1]		If LoyaltyUnit is Monetary
LoyaltyTotals	[0..6]	L	If both TransactionCount and TransactionAmount are not equal to zero

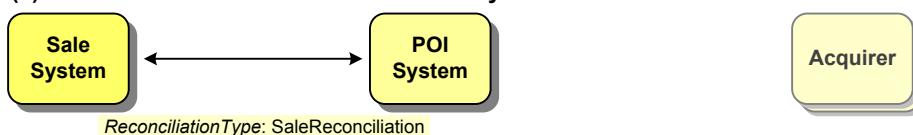
<i>ReconciliationResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
TransactionType	[1..1]		Award, ReverseAward, Redemption, ReverseRedemption, Rebate, ReverseRebate, Failed
TransactionCount	[1..1]		
TransactionAmount	[1..1]		

4.4.1.4 Reconciliation Processing

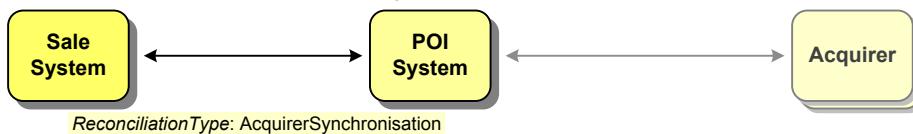
The Sale System could request different types of reconciliations to the POI System through the element *ReconciliationType*:

- (1) “*SaleReconciliation*”: Reconciliation and closure of the current reconciliation period, without any synchronisation with Acquirer reconciliations.
- (2) “*AcquirerSynchronisation*”: Reconciliation and closure of the current period, with synchronisation of reconciliations between the POI and Acquirers.
- (3) “*AcquirerReconciliation*”: Reconciliation between the POI and one or several Acquirers. In this case, there is no reconciliation between the Sale System and the POI System.
- (4) “*PreviousReconciliation*”: Result of an already closed period, i.e. the result of a previous reconciliation.

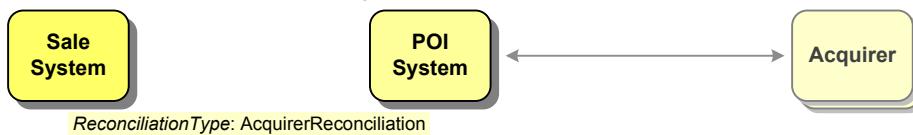
(1) Reconciliation between Sale and POI Systems



(2) With Acquirer Reconciliation synchronisation



(3) Acquirer Reconciliation Only



(4) Previous Reconciliation Result

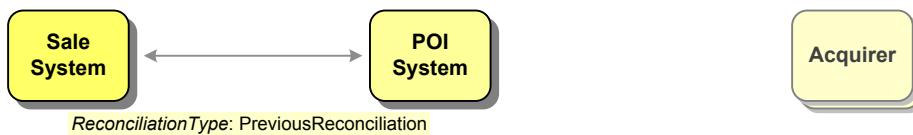


Figure 182: Type of Reconciliation Requests

When the type of reconciliation (*ReconciliationType*) is “*SaleReconciliation*” (case (1)), there is no synchronisation between the reconciliation of Sale/POI and POI/Acquirer(s).

Considering the example below with the reconciliations between:

- The Sale and the POI, period $n-1$, n and $n+1$,
- The POI and the Acquirer 1, period $i-1$, i and $i+1$,
- The POI and the Acquirer 2, period $j-2$, $j-1$, j , $j+1$ and $j+2$.

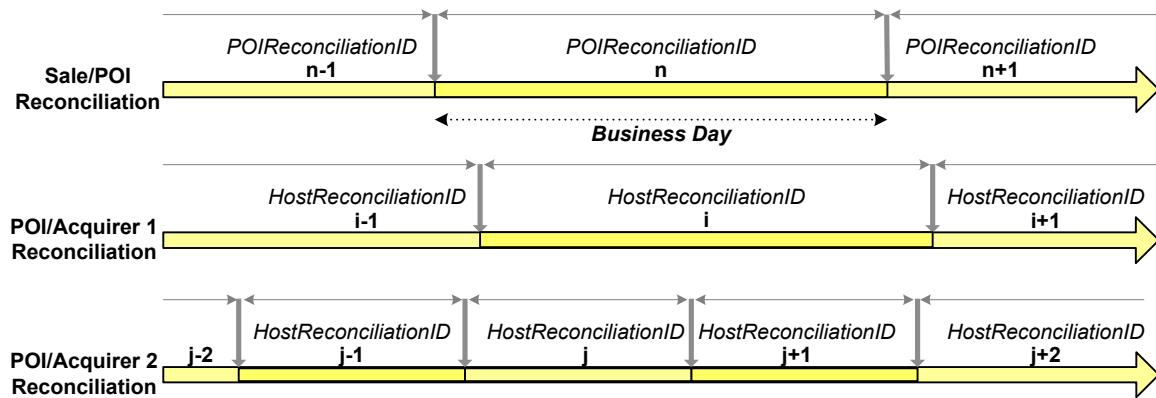


Figure 183: Reconciliation Periods Overlapping

The Sale/POI period n includes transactions acquired by:

- The Acquirer 1, period $i-1$,
- The Acquirer 1, period i ,
- The Acquirer 2, period $j-1$,
- The Acquirer 2, period j ,
- The Acquirer 2, period $j+1$.

All these transactions have the identification of the related period in the Payment response message, and can be split by acquirer reconciliation period.

Rule 1: The Reconciliation request message could be sent by the Sale Server or a Sale Terminal to the POI Server or a POI Terminal.

Rule 2: The reconciliation between the Sale System and the POI System provide the payment and loyalty totals related to the current reconciliation period. The reconciliation closes the current reconciliation period before counting the totals and opens a new reconciliation period.

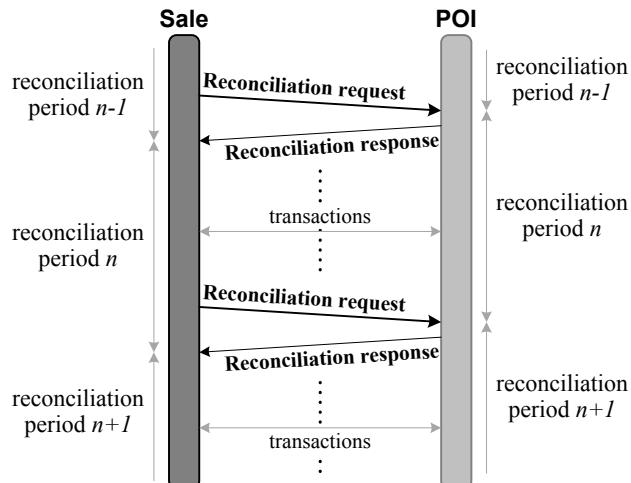


Figure 184: Reconciliation between Sale System and POI System

Rule 3: Reconciliation periods are identified by the digit string *POIReconciliationID*, which is assigned by the POI System. Every payment and loyalty transactions are assigned to a reconciliation period by the POI identified by the component *POIReconciliationID* of the response message.

Rule 4: Reconciliation periods between POI System and Acquirers are identified by the digit string *HostReconciliationID*, and must be present if there a synchronisation with Acquirer reconciliations or the Sale System has configured the POI to provide them.

Rule 5: The criterions to compute and to present the totals in the Reconciliation response are configured in the POI System, and never sent in the Reconciliation request:

- per card brand (*CardBrand*),
- per POI/Acquirer reconciliation period identification (*HostReconciliationID*),
- per POI Terminal (*POIID*),
- per Sale Terminal (*SaleID*),
- per Cashier (*OperatorID*),
- per shift (*ShiftNumber*),
- per *TotalsGroupID*.

Rule 6: A POI and Sale configuration parameter specifies if the Sale System and the POI System can manage transactions during the reconciliation processing. In this case:

- The POI System must indicate unambiguously in the message response of the transaction the reconciliation period.
- The POI System cannot send transaction responses of the closed period after the reconciliation response message.

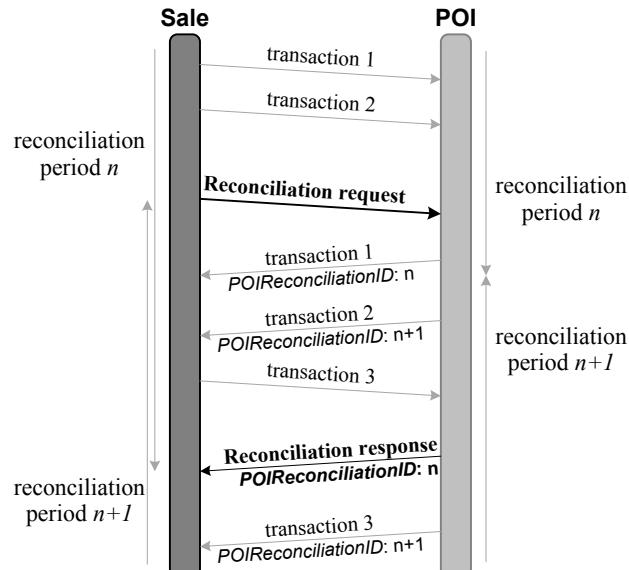


Figure 185: Reconciliation between Sale System and POI System

In the example above, transactions 1 and 2 are in progress when the Sale System requests reconciliation. The POI System starts the reconciliation and closes the period n just after the end of the transaction 1. The next transactions, 2 and 3, belong to the reconciliation period $n+1$.

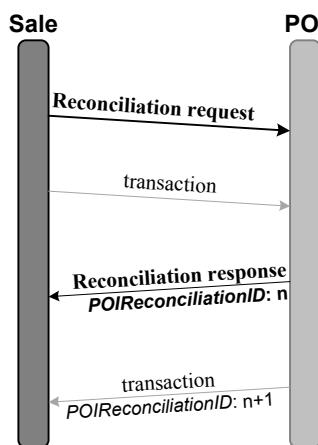


Figure 186: Overlapping of Reconciliation Periods

Rule 7: If POI and Sale System are not configured for accepting transaction during Reconciliations, no transactions can be processed during reconciliation.
 If transactions are in progress (i.e. the message response is not sent yet, or the processing is not completed) during the reception of the Reconciliation request message, the Reconciliation response is sent with *Result*="Failure" and *ErrorCondition*=" Busy" (see section 4.6.5.1.2 *POI Busy*).
 If a transaction request is received during the processing of a reconciliation, the transaction response message gets the same result.

Rule 8: The counting of transaction of a reconciliation period must be conforming to the rules of the table below.

<i>Transactions</i>	OneTime Reserv	Compl. Deferred	First Reserv	Update Reserv	Compl. Reserv	Reverse Debit
uncompleted one time reservation	X					
uncompleted one time reservation reversed	X					
completed one time reservation	X	X				
completed one time reservation reversed	X	X				X
uncompleted reservation not updated			X			
uncompleted reservation not updated but reversed			X			
uncompleted and updated reservation			X	X		
uncompleted, updated and reversed reservation			X	X		
completed reservation not updated			X		X	
completed reservation not updated but reversed			X		X	X
completed and updated reservation			X	X	X	
completed, updated and reversed reservation			X	X	X	X

Table 16: Reconciliation Transaction Counting

Example:

The POI System makes the following transactions:

- (1) 2 Payments with PaymentType="Normal"
- (2) 1 Payment with PaymentType="UpdateReservation"
- (3) 1 Payment with PaymentType="OneTimeReservation", followed by the Payment with PaymentType="Completion"
- (4) 1 Payment with PaymentType="Completion" with an OriginalPOITransaction of PaymentType="FirstReservation"
- (5) A Reversal of the Payment performed in the step (4)
- (6) 1 Payment with PaymentType="Refund"

The count of transactions per type of transaction would be:

Debit:	2 (1)
Credit:	1 (6)
ReverseDebit:	1 (5)
ReverseCredit:	0
OneTimeReservation:	1 (3)
CompletedDeferred:	1 (3)
FirstReservation:	0
UpdateReservation:	1 (2)
CompletedReservation:	1 (4)

CashAdvance: 0)

Rule 9: If the *POIReconciliationID* reconciliation cannot be provided by the POI System for a “PreviousReconciliation” *ReconciliationType*, the Reconciliation response is sent with *Result*=“Failure” and *ErrorCondition*=“NotFound” (see section 4.6.4.4.3 Reconciliation Not Found).

Rule 10: If the Reconciliation administrative service, or the *ReconciliationType* reconciliation is not available in the POI System, the Reconciliation response is sent with *Result*=“Failure” and *ErrorCondition*=“UnavailableService” (see section 4.6.4.3.3 *Unavailable Administrative Service*).

Rule 11: If the Reconciliation administrative service couldn't be provided by the POI System, the Reconciliation response is sent with *Result*=“Failure” and *ErrorCondition*=“DeviceOut” (see section 4.6.3.1 *DeviceOut Error*).

Rule 12: If the Reconciliation with the Acquirers fails for a “AcquirerSynchronisation” or “AcquirerReconciliation” *ReconciliationType*, the Reconciliation response is sent with *Result*=“Partial” or “Failure” depending of the number of failures. The *ErrorCondition*=“UnreachableHost” (see section 4.6.8.1 *UnreachableHost Error*) has to be put in the record of the related Acquirer.

Rule 13: If the totals mismatch between Sale System and POI for a reconciliation, there is no automatic resolution.

Rule 14: If the Reconciliation response is not received by the Sale System for some reason, the Sale System must follow the General Error Resolution (see section 4.7.5.1 *Error Resolutions Specifications*):

- 1) The Sale System sends a TransactionStatus request message related to the Reconciliation.
- 2) If the Reconciliation is still in progress, the ReconciliationTransactionStatus response for the last Reconciliation, is “InProgress”.
- 3) If the Reconciliation is finished, the ReconciliationTransactionStatus response for the last Reconciliation, contains the whole Reconciliation response of the last reconciliation.

The Sale System and the POI System have the following configuration parameters:

Name	Reconciliation Details
<i>Definition</i>	Details of the transaction totals sent in the reconciliation response: per card brand per POI/Acquirer reconciliation period identification per POI Terminal per Sale Terminal per Cashier per shift per TotalsGroupID
<i>Usage</i>	Allows computation of transaction totals according to the various criteria.
<i>Specification</i>	4.4.1.4 Reconciliation Processing

Configuration 11: Reconciliation Details

4.4.1.5 Error Cases

When the Reconciliation request is successfully processed, the Reconciliation response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotFound

The POI System receives a Reconciliation request from the Sale System, with the identification *POIReconciliationID* of a previous reconciliation which cannot be found by the POI System (see section 4.6.4.4.3 *Reconciliation Not Found*).

Busy

The POI System receives a Reconciliation request and transactions are still in progress, or the POI System receives a message request to process a transaction during a reconciliation processing, and the configuration does not allow mix of transaction and reconciliation (see section 4.6.5.1.2 *POI Busy*).

UnavailableService

The Reconciliation administrative service, or the *ReconciliationType* reconciliation is not available in the POI System (see section 4.6.4.3.3 *Unavailable Administrative Service*).

DeviceOut

The Reconciliation administrative service couldn't be provided by the POI System (see section 4.6.3.1 *DeviceOut Error*).

UnreachableHost

The Reconciliation with the Acquirers fails for “AcquirerSynchronisation” or “AcquirerReconciliation” *ReconciliationType* (see section 4.6.8.1 *UnreachableHost Error*) has to be put in the record of the related Acquirer.

Refusal

The Reconciliation is refused by the payment Acquirer (see section 4.6.8.2 *Refusal Error*).

4.4.1.6 Example

The Sale Server requests to the POI Server a Reconciliation with closure of the current period, without any Acquirers synchronisation.

MessageHeader		(message example 37)
MessageClass	Service	
MessageCategory	Reconciliation	
MessageType	Request	
ServiceID	617	
SaleID	SaleServer	
POIID	POIServer	
ReconciliationRequest		
ReconciliationType	SaleReconciliation	

The transactions realised during the period number 8926 are presented on the following page:

- 182 debits Visa for 47184.17
- 1 credit Visa for 27.01
- 157 debits Mastercard for 3753.61
- 1 debit Mastercard reversed for 37.99
- 143 awards SuperBonus for 1889 points

MessageHeader		(message example 38)
MessageClass	Service	
MessageCategory	Reconciliation	
MessageType	Response	
ServiceID	617	
SaleID	SaleServer	
POIID	POIServer	
ReconciliationResponse		
Response		
Result	Success	
POIReconciliationID	8926	
ReconciliationType	SaleReconciliation	
TransactionTotals		
PaymentInstrumentType	Card	
AcquirerID	876355543	
HostReconciliationID	98535	
CardBrand	Visa	
PaymentCurrency	EUR	
PaymentTotals		
TransactionType	Debit	
TransactionCount	182	
TransactionAmount	47184.17	
PaymentTotals		
TransactionType	Credit	
TransactionCount	1	
TransactionAmount	27.01	
TransactionTotals		
PaymentInstrumentType	Card	
AcquirerID	876355543	
HostReconciliationID	98535	
CardBrand	ExpressCard	
PaymentCurrency	EUR	
PaymentTotals		
TransactionType	Debit	
TransactionCount	157	
TransactionAmount	3753.61	
PaymentTotals		
TransactionType	ReverseDebit	
TransactionCount	1	
TransactionAmount	37.99	
TransactionTotals		
PaymentInstrumentType	Card	
AcquirerID	93582	
CardBrand	SuperBonus	
LoyaltyTotals		
TransactionType	Award	
TransactionCount	143	
TransactionAmount	1889	

4.4.2 Get Totals Messages

4.4.2.1 Presentation of the Messages

The Get Totals request message body *GetTotalsRequest*, contains the following information:

1. The way to compute and present the totals: *TotalDetails*, per POI Terminal, per Sale Terminal, per Cashier, per shift, and/or per Sale group.
2. The filter of the transaction if any: *TotalFilter*,
 - a. For a particular POI Terminal,
 - b. For a particular Sale Terminal,
 - c. For a particular Cashier,
 - d. For a particular shift,
 - e. For a particular Sale group.

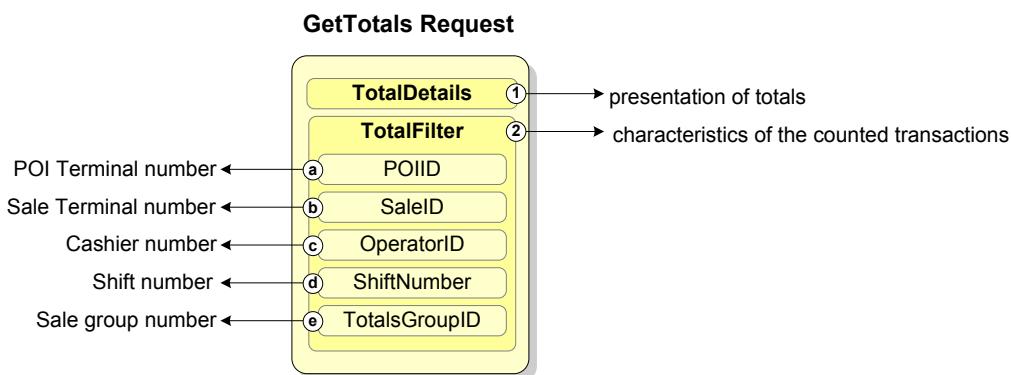


Figure 187: Get Totals Request Information

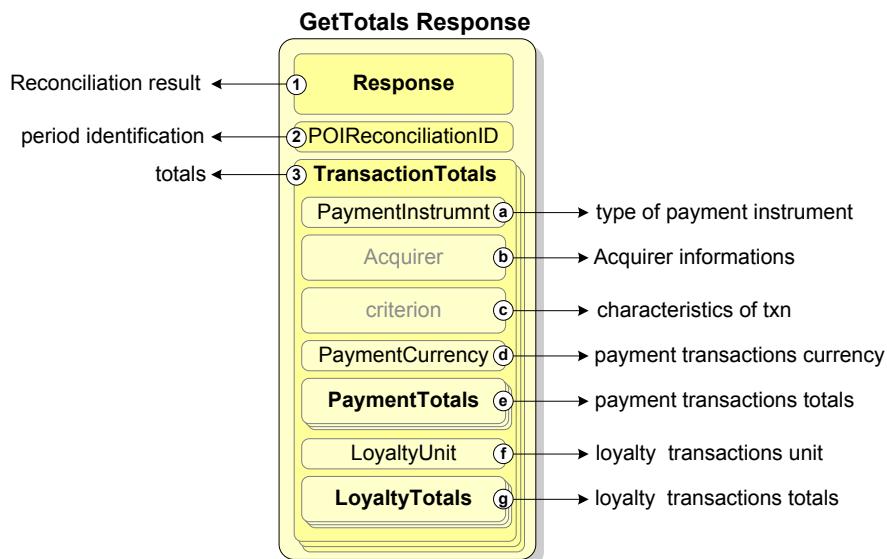


Figure 188: Get Totals Response Information

The Get Totals response message body *GetTotalsResponse*, contains practically the same information that the Reconciliation message response (there are only one type of request without any synchronisation with Acquirers):

1. The result of the Get Totals: *Response*.
2. The identification of the reconciliation period related to the provided totals: *POIReconciliationID*.
3. A sequence data structure containing totals of transactions for computed with some criteria *TransactionTotals*:
 - a. The type of payment instrument used for these transactions: *PaymentInstrumentType*.
 - b. Information related to the Acquirer managing the brand: Acquirer identification *AcquirerID*, and the identification of the Acquirer reconciliation *HostReconciliationID*.
 - c. Value of criterions used to organize the totals: POI Terminal *POIID*, Sale Terminal *SaleID*, Cashier *OperatorID*, shift *ShiftNumber*, and *TotalsGroupID*.
 - d. The payment currency: *PaymentCurrency*.
 - e. The totals of the payments transaction per type of transactions: *PaymentTotals*, containing the numbers of transaction *TransactionCount*, and the sum of the amounts *TransactionAmount*.
 - f. The unit of the loyalty transactions: *LoyaltyUnit* and *LoyaltyCurrency*.
 - g. The totals of the loyalty transaction per type of transactions: *LoyaltyTotals*, containing also the numbers of transaction *TransactionCount*, and the sum of the amounts *TransactionAmount*.

4.4.2.2 *GetTotals Request Layout*

<i>GetTotalsRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		GetTotals
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
GetTotalsRequest	[1..1]		
TotalDetails	[0..1]		Require to present totals per value of element included in this cluster (POI Terminal, Sale Terminal, Cashier, Shift, TotalsGroupID)
TotalFilter	[0..1]		If structure is not empty
POIID	[0..1]		If totals in the response have to be computed only for this particular value of POIID
SaleID	[0..1]		If totals in the response have to be computed only for this particular value of SaleID
OperatorID	[0..1]		If totals in the response have to be computed only for this particular value of OperatorID
ShiftNumber	[0..1]		If totals in the response have to be computed only for this particular value of ShiftNumber
TotalsGroupID	[0..1]		If totals in the response have to be computed only for this particular value of TotalsGroupID

4.4.2.3 **GetTotals Response Layout**

<i>GetTotalsResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		GetTotals
MessageType	[1..1]		Response
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
GetTotalsResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>same as PaymentResponse</i>
POIReconciliationID	[1..1]		
TransactionTotals	[0..n]		if Response.Result is Success One set of totals per value of CardBrand and AcquirerID, ..., TotalsGroupID if presents
PaymentInstrumentType	[1..1]		
AcquirerID	[0..1]		If available
HostReconciliationID	[0..1]		If available
CardBrand	[0..1]		If configured to present totals per card brand, and Response.Result is Success
POIID	[0..1]		If requested in the message request
SaleID	[0..1]		If requested in the message request
OperatorID	[0..1]		If requested in the message request
ShiftNumber	[0..1]		If requested in the message request
TotalsGroupID	[0..1]		If requested in the message request
PaymentCurrency	[0..1]		If payment record
PaymentTotals	[0..10]		Totals not equal to zero
TransactionType	[1..1]		Debit, Credit, ReverseDebit, ReverseCredit, OneTimeReservation, CompletedDeferred, FirstReservation, UpdateReservation, CompletedReservation, CashAdvance, IssuerInstalment, Failed, Declined
TransactionCount	[1..1]		
TransactionAmount	[1..1]		
LoyaltyUnit	[0..1]		default Point
LoyaltyCurrency	[0..1]		If LoyaltyUnit is Monetary
LoyaltyTotals	[0..6]		Totals not equal to zero
TransactionType	[1..1]		Award, ReverseAward, Redemption, ReverseRedemption, Rebate, ReverseRebate, Failed
TransactionCount	[1..1]		
TransactionAmount	[1..1]		

4.4.2.4 Get Totals Processing

- Rule 1: The Get Totals request message could be sent by the Sale Server or a Sale Terminal to the POI Server or a POI Terminal.
- Rule 2: The transactions totals provided in the Get Totals response are the transactions realised since the beginning of the current reconciliation period until the reception of the Get Totals request message. The current reconciliation period is never closed by the Get Totals exchange.
- Rule 3: The criteria below, used for the computation and the presentation of the totals in the Get Totals response are sent in the cluster *TotalDetails* of the request message, rather than using the parameters configured in the POI System for Reconciliation:
- per POI Terminal (*POIID*),
 - per Sale Terminal (*SaleID*),
 - per Cashier (*OperatorID*),
 - per shift (*ShiftNumber*),
 - per *TotalsGroupID*.
- On the other hand, the following criteria is configured:
- per Acquirer reconciliation with POI (*HostReconciliationID*),
- Rule 4: If the values requested in the components of *TotalFilter* are not found in the transactions of the POI System, the Get Totals response provide no record for these values. The Get Totals response could be empty, i.e. without any *TransactionTotals*.
- Rule 5: The Get Totals request could be requested without stopping the flow of transactions toward the POI (Get Totals is a report and not a synchronisation between the Sale and POI Systems).
- Rule 6: The counting of transaction is realised by message pair exchanges and must be conforming to the rules of the table presented for the Reconciliation.
- Rule 7: If the Get Totals administrative service is not available in the POI System, the Get Totals response is sent with *Result*=”Failure” and *ErrorCondition*=”UnavailableService” (see section 4.6.4.3.3 *Unavailable Administrative Service*).
- Rule 8: If the Get Totals administrative service couldn't be provided by the POI System, the Get Totals response is sent with *Result*=”Failure” and *ErrorCondition*=”DeviceOut” (see section 4.6.3.1 *DeviceOut Error*).
- Rule 9: If the Get Totals response is not received by the Sale System for some reason, the Sale System must follow the error resolution of Idempotent Requests, i.e. the Get Totals must be repeated (see section 4.7.5.1 *Error Resolutions Specifications*):

4.4.2.5 Error Cases

When the Get Totals request is successfully processed, the Get Totals response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

UnavailableService

The Get Totals administrative service is not available in the POI System (see section 4.6.4.3.3 *Unavailable Administrative Service*).

DeviceOut

The Get Totals administrative service couldn't be provided by the POI System (see section 4.6.3.1 *DeviceOut Error*).

4.4.2.6 Example

The Sale Server requests to the POI Server the totals for the Cashier 213, presented per Sale Terminal and shift.

MessageHeader		(message example 39)
MessageClass	Service	
MessageCategory	GetTotals	
MessageType	Request	
ServiceID	619	
SaleID	SaleServer	
POIID	POIServer	
GetTotalsRequest		
TotalDetails	SaleID ShiftNumber	
TotalFilter		
OperatorID	213	

MessageHeader		(message example 40)
MessageClass	Service	
MessageCategory	GetTotals	
MessageType	Response	
ServiceID	619	
SaleID	SaleServer	
POIID	POIServer	
GetTotalsResponse		
Response		
Result	Success	
POIReconciliationID	8927	
TransactionTotals		
PaymentInstrumentType	Card	
AcquirerID	876355543	
HostReconciliationID	98535	
CardBrand	CardPlus	
SaleID	SaleTermA	
OperatorID	213	
ShiftNumber	1	
PaymentCurrency	EUR	
PaymentTotals		
TransactionType	Debit	
TransactionCount	61	
TransactionAmount	4253.19	
PaymentTotals		
TransactionType	Credit	
TransactionCount	1	
TransactionAmount	27.01	
TransactionTotals		
PaymentInstrumentType	Card	
AcquirerID	876355543	
HostReconciliationID	98535	
CardBrand	CardPlus	
SaleID	SaleTermD	
OperatorID	213	
ShiftNumber	2	
PaymentCurrency	EUR	
PaymentTotals		
TransactionType	Debit	

TransactionCount	45
TransactionAmount	744.79
TransactionTotals	
PaymentInstrumentType	Card
AcquirerID	876355543
HostReconciliationID	98535
CardBrand	ExpressCard
SaleID	SaleTermD
OperatorID	29
ShiftNumber	1
PaymentCurrency	EUR
PaymentTotals	
TransactionType	Debit
TransactionCount	17
TransactionAmount	353.61
TransactionTotals	
PaymentInstrumentType	Card
AcquirerID	876355543
HostReconciliationID	98535
CardBrand	ExpressCard
SaleID	SaleTermD
OperatorID	213
ShiftNumber	2
PaymentCurrency	EUR
PaymentTotals	
TransactionType	Debit
TransactionCount	18
TransactionAmount	711.48
PaymentTotals	
TransactionType	ReverseDebit
TransactionCount	1
TransactionAmount	37.99
TransactionTotals	
PaymentInstrumentType	Card
AcquirerID	93582
CardBrand	SuperBonus
SaleID	SaleTermD
OperatorID	213
ShiftNumber	1
LoyaltyTotals	
TransactionType	Award
TransactionCount	25
TransactionAmount	278
TransactionTotals	
PaymentInstrumentType	Card
AcquirerID	93582
CardBrand	SuperBonus
SaleID	SaleTermD
OperatorID	213
ShiftNumber	2
LoyaltyTotals	
TransactionType	Award
TransactionCount	23
TransactionAmount	182

4.4.3 Balance Inquiry Messages

4.4.3.1 Presentation of the Messages

The Balance Inquiry request message body *BalanceInquiryRequest*, contains the following information:

1. Information about the account of a payment card *PaymentAccountReq*:
 - a. The type of account on which the balance is requested: *AccountType*,
 - b. Identification of a previous transaction: *CardAcquisitionReference*, if the card has to be found on the transaction log and not asked to the Customer,
 - c. Information on the payment instrument of the generic data structure : *PaymentInstrumentData* containing *CardData*, *MobileData*, or *StoredValueAccountID*, if the card is read by the Sale Terminal.
2. Information about the account of a loyalty card *LoyaltyAccountReq*:
 - a. Identification of a previous transaction: *CardAcquisitionReference*, if the card has to be found on the transaction log and not asked to the Customer,
 - b. Loyalty account information: *LoyaltyAccountID*, if the card is read by the Sale Terminal.

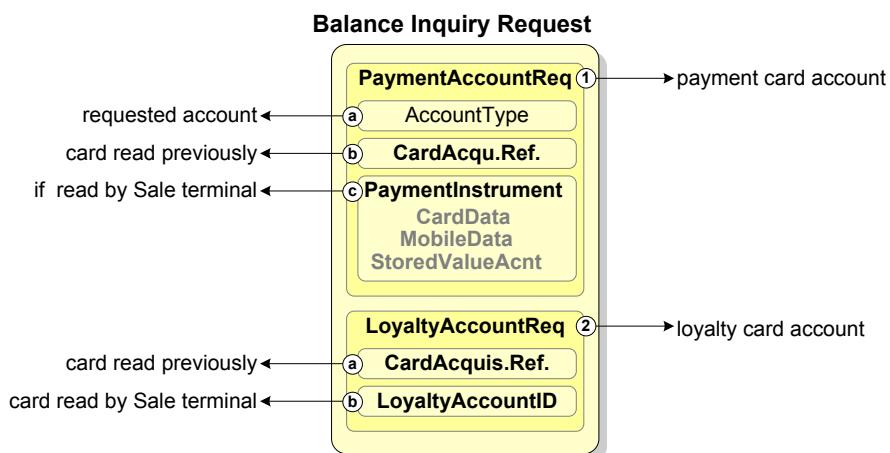


Figure 189: Balance Inquiry Request Information

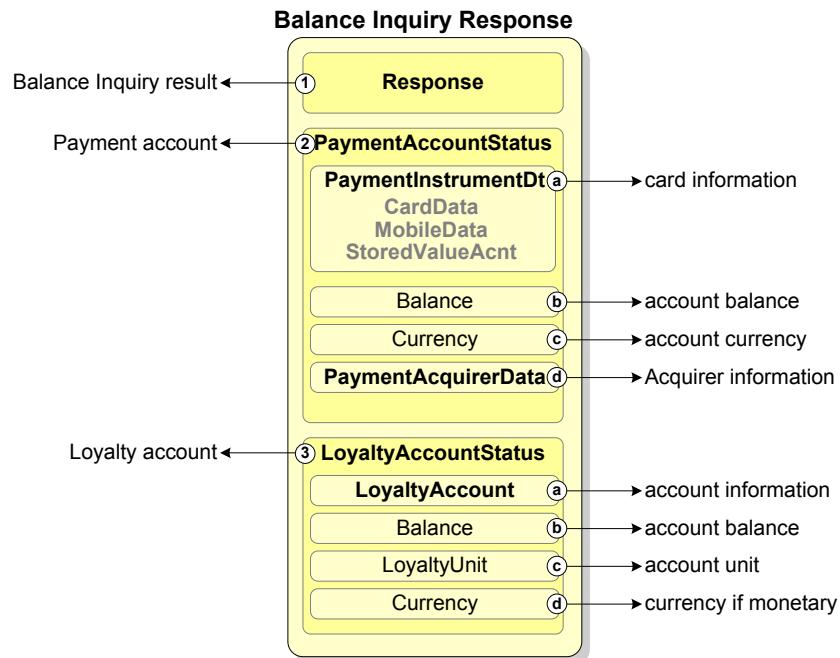


Figure 190: Balance Inquiry Response Information

The Balance Inquiry response message body *BalanceInquiryResponse*, contains the account balances provided by the card:

1. The result of the Balance Inquiry: *Response*.
2. For an account related to a payment card, *PaymentAccountStatus*:
 - a. Information on the payment instrument of the generic data structure : *PaymentInstrumentData* containing *CardData*, *CheckData*, or *StoredValueAccountId*.
 - b. The balance of the account attached to the payment card (default or requested): *CurrentBalance*.
 - c. The currency of the account balance: *Currency*.
 - d. The information related to the payment Acquirer: *PaymentAcquirerData*.
3. For an account related to a loyalty card, *LoyaltyAccountStatus*:
 - a. The loyalty card and loyalty account information: *LoyaltyAccount*.
 - b. The balance of the loyalty account: *CurrentBalance*.
 - c. The unit of the loyalty account (point or monetary): *LoyaltyUnit*.
 - d. The currency of the balance for a monetary account: *Currency*.

4.4.3.2 Balance Inquiry Request Layout

<i>BalanceInquiryRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		BalanceInquiry
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
BalanceInquiryRequest	[1..1]		
PaymentAccountReq	[0..1]		
AccountType	[0..1]		default Default
CardAcquisitionReference	[0..1]		if the card data comes from the previous message pair
TransactionID	[1..1]		
TimeStamp	[1..1]		
PaymentInstrumentData	[0..1]		<i>see PaymentRequest</i>
PaymentInstrumentType	[1..1]		
CardData	[0..1]		<i>see PaymentRequest</i>
EntryMode	[1..1]		
ProtectedCardData	[0..1]		<i>see PaymentRequest</i>
SensitiveCardData	[0..1]		<i>see PaymentRequest</i>
PAN	[0..1]		<i>see PaymentRequest</i>
CardSeqNumb	[0..1]		<i>see PaymentRequest</i>
ExpiryDate	[0..1]		<i>see PaymentRequest</i>
TrackData	[0..4]		<i>see PaymentRequest</i>
TrackNumb	[0..1]		<i>see PaymentRequest</i>
TrackFormat	[0..1]		<i>see PaymentRequest</i>
TrackValue	[1..1]		
MobileData	[0..1]		<i>see PaymentRequest</i>
MobileCountryCode	[0..1]		<i>see PaymentRequest</i>
MobileNetworkCode	[0..1]		<i>see PaymentRequest</i>
MaskedMSISDN	[0..1]		<i>see PaymentRequest</i>
Geolocation	[0..1]		<i>see PaymentRequest</i>
GeographicCoordinates	[0..1]		
Latitude	[1..1]		
Longitude	[1..1]		
UTMCoordinates	[0..1]		
UTMZone	[1..1]		
UTMEastward	[1..1]		
UTMNorthward	[1..1]		
ProtectedMobileData	[0..1]		<i>see PaymentRequest</i>
SensitiveMobileData	[0..1]		<i>see PaymentRequest</i>
MSISDN	[1..1]		
IMSI	[0..1]		<i>see PaymentRequest</i>

<i>BalanceInquiryRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
IMEI	[0..1]		<i>see PaymentRequest</i>
StoredValueAccountID	[0..1]		<i>see PaymentRequest</i>
StoredValueAccountType	[1..1]		
StoredValueProvider	[0..1]		<i>see PaymentRequest</i>
OwnerName	[0..1]		<i>see PaymentRequest</i>
ExpiryDate	[0..1]		<i>see PaymentRequest</i>
EntryMode	[1..1]		
IdentificationType	[1..1]		
StoredValueID	[1..1]		
LoyaltyAccountReq	[0..1]		<i>see PaymentRequest</i>
CardAcquisitionReference	[0..1]		if the card data comes from the previous message pair
TransactionID	[1..1]		
TimeStamp	[1..1]		
LoyaltyAccountID	[0..1]		
EntryMode	[1..1]		
IdentificationType	[1..1]		
LoyaltyID	[1..1]		

4.4.3.3 Balance Inquiry Response Layout

<i>BalanceInquiryResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		BalanceInquiry
MessageType	[1..1]		Response
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
BalanceInquiryResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>
PaymentAccountStatus	[0..1]		If BalanceInquiryRequest. PaymentAccountReq present
PaymentInstrumentData	[0..1]		If payment instrument is analysed
PaymentInstrumentType	[1..1]		
CardData	[0..1]		<i>see PaymentResponse</i>
PaymentBrand	[0..1]		
MaskedPAN	[0..1]		<i>see PaymentResponse</i>
PaymentAccountRef	[0..1]		<i>see PaymentResponse</i>
EntryMode	[0..1]		
CardCountryCode	[0..1]		If available in the card
ProtectedCardData	[0..1]		<i>see PaymentRequest</i>
SensitiveCardData	[0..1]		<i>see PaymentResponse</i>
PAN	[1..1]		
CardSeqNumb	[0..1]		<i>see PaymentResponse</i>
ExpiryDate	[0..1]		<i>see PaymentResponse</i>
TrackData	[0..4]		<i>see PaymentResponse</i>
TrackNumb	[0..1]		default 2
TrackFormat	[0..1]		default ISO
TrackValue	[1..1]		
AllowedProduct	[0..n]		If the card has restrictions on product that can be purchased.
ProductCode	[1..1]		
EanUpc	[0..1]		
ProductLabel	[0..1]		
AdditionalProductInfo	[0..1]		
MobileData	[0..1]		<i>see PaymentResponse</i>
MobileCountryCode	[0..1]		<i>see PaymentResponse</i>
MobileNetworkCode	[0..1]		<i>see PaymentResponse</i>
MaskedMSISDN	[0..1]		<i>see PaymentResponse</i>
Geolocation	[0..1]		<i>see PaymentResponse</i>
GeographicCoordinates	[0..1]		

<i>BalanceInquiryResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
Latitude	[1..1]		
Longitude	[1..1]		
UTMCoordinates	[0..1]		
UTMZone	[1..1]		
UTMEastward	[1..1]		
UTMNorthward	[1..1]		
ProtectedMobileData	[0..1]		<i>see PaymentResponse</i>
SensitiveMobileData	[0..1]		<i>see PaymentResponse</i>
MSISDN	[1..1]		
IMSI	[0..1]		<i>see PaymentResponse</i>
IMEI	[0..1]		<i>see PaymentResponse</i>
StoredValueAccountID	[0..1]		<i>see PaymentResponse</i>
StoredValueAccountType	[1..1]		
StoredValueProvider	[0..1]		<i>see PaymentResponse</i>
OwnerName	[0..1]		<i>see PaymentResponse</i>
ExpiryDate	[0..1]		<i>see PaymentResponse</i>
EntryMode	[1..1]		<i>see PaymentResponse</i>
IdentificationType	[1..1]		
StoredValueID	[1..1]		
CurrentBalance	[0..1]		If PaymentInstrumentData present and Result is Success
Currency	[0..1]		If PaymentInstrumentData present and Result is Success
PaymentAcquirerData	[0..1]		If a card is analysed
AcquirerID	[0..1]		If several Acquirers
MerchantID	[1..1]		
AcquirerPOIID	[1..1]		
AcquirerTransactionID	[0..1]		If provided by the Acquirer
ApprovalCode	[0..1]		If available
LoyaltyAccountStatus	[0..1]		If PaymentInstrumentData absent and loyalty account analysed
LoyaltyAccount	[1..1]		
LoyaltyAccountID	[1..1]		
EntryMode	[1..1]		
IdentificationType	[1..1]		
IdentificationSupport	[1..1]		
LoyaltyID	[1..1]		
LoyaltyBrand	[0..1]		If a card is analysed
CurrentBalance	[0..1]		If Result is Success
LoyaltyUnit	[0..1]		default Point
Currency	[0..1]		If Result is Success and If LoyaltyUnit is "Monetary"
PaymentReceipt	[0..*]		If Basic profile implementation with no printer on the POI.
DocumentQualifier	[1..1]		SaleReceipt or CashierReceipt
IntegratedPrintFlag	[0..1]		same as PrintRequest
RequiredSignatureFlag	[0..1]		default False.

<i>BalanceInquiryResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
OutputContent	[1..1]		
OutputFormat	[1..1]		Text, XHTML
OutputText	[0..n]		same as Display
Text	[1..1]		same as Display
CharacterSet	[0..1]		same as Display
Font	[0..1]		same as Display
StartColumn	[0..1]		same as Display
Color	[0..1]		same as Display
CharacterWidth	[0..1]		same as Display
CharacterHeight	[0..1]		same as Display
CharacterStyle	[0..1]		same as Display
Alignment	[0..1]		same as Display
EndOfLineFlag	[0..1]		same as Display
OutputXHTML	[0..1]		same as Display

4.4.3.4 Balance Processing

Balance Inquiry provides the balance of an account, identified by a card or other identifier. The card could be a payment card, a loyalty card, a stored value card.

- Rule 1: The same card could identify several accounts. Both Balance Inquiry request and response may contain any combination of payment (*PaymentAccountReq* and *PaymentAccountStatus*), loyalty (*LoyaltyAccountReq* and *LoyaltyAccountStatus*) and stored value (*StoredValueAccountID* and *StoredValueAccountStatus*) cards.
- Rule 2: When the card or the account, is read or identified by the Sale Terminal, information is inserted in the Balance Inquiry request message. Then the POI Terminal must provide the balance of the related account(s).
- Rule 3: When *CardAcquisitionReference* is present in the Balance Inquiry request message. The POI Terminal must retrieve the card/account information from the previous transaction. If the transaction could not be found by the POI Terminal, the Balance Inquiry response message is sent with *Result*="Failure" and *ErrorCondition*="NotFound" (see section 4.6.4.4.1 *Transaction Not Found*).
- Rule 4: When the Balance Inquiry request message is empty or contains only *PaymentAccountReq.AccountType*, the card has to be read by the POI Terminal.
- Rule 5: If the Balance Inquiry response is not received by the Sale System for some reason, the Sale System can abort the transaction, but must follow the error resolution of Idempotent Requests, i.e. the Balance Inquiry could be repeated (see section 4.7.5.1 *Error Resolutions Specifications*).

4.4.3.5 Error Cases

When the Balance Inquiry request is successfully processed, the Balance Inquiry response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

UnavailableService

The Balance Inquiry administrative service is not available in the POI System (see section 4.6.4.3.3 *Unavailable Administrative Service*).

DeviceOut

The Balance Inquiry administrative service couldn't be provided by the POI System, e.g. the communication to request the Host (see section 4.6.3.1 *DeviceOut Error*).

Cancel

The user has aborted the transaction on the Customer interface (e.g. the POI Terminal keyboard), because he do not want to continue the transaction, see section 4.6.6.2.1 *User Cancellation*.

InvalidCard

The POI terminates the transaction because no card is entered by the Customer, or the inserted card is not configured in the system and cannot be processed (see section 4.6.7.1.1 *No Card Entered* and section 4.6.7.1 *InvalidCard Error*).

UnreachableHost

The Payment Acquirer is unreachable or has not answered to an online request, so it is considered as temporary unavailable. The Cashier has not forced the transaction, and the payment cannot be accepted (see section 4.6.8.1.1 *Host Unreachable* and section 4.6.8.1.2 *No Host Answer*).

Refusal

The transaction is refused by the payment Acquirer or the rules associated to the card. The Cashier has not forced the transaction, and the payment cannot be repeated. A specific message is normally displayed to the Customer and the Cashier if they are presents, the information below could be logged for further information (see section 4.6.8.2 *Refusal Error*).

4.4.3.6 Example

The Sale Server requests to the POI Server the balance of a payment card, which is also a loyalty card.

MessageHeader		(message example 41)
MessageClass	Service	
MessageCategory	BalanceInquiry	
MessageType	Request	
ServiceID	629	
SaleID	SaleTermA	
POIID	POITerm1	
BalanceInquiryRequest		
MessageHeader		(message example 42)
MessageClass	Service	
MessageCategory	BalanceInquiry	
MessageType	Response	
ServiceID	629	
SaleID	SaleTermA	
POIID	POITerm1	
BalanceInquiryResponse		
Response		
Result	Success	
PaymentAccountStatus		
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	CardPlus	
EntryMode	MagStripe	
SensitiveCardData		
PAN	6011014570541535	
ExpiryDate	0411	
CurrentBalance	832.59	
Currency	EUR	
PaymentAcquirerData		
AcquirerID	400012	
MerchantID	mer77-130209	
AcquirerPOIID	963276433	
ApprovalCode	8347	
LoyaltyAccountStatus		
LoyaltyAccount		
LoyaltyAccountID		
EntryMode	MagStripe	
IdentificationType	PAN	
IdentificationSupport	LinkedCard	
LoyaltyID	8678252755678	
LoyaltyBrand	SuperBonus	
CurrentBalance	112	
LoyaltyUnit	Point	

4.5 Devices Services

4.5.1 Output Content Formats

All messages which contain information to display or print, as the *DisplayRequest* message, carry this information in the data structure *OutputContent*.

OutputContent could contain various formats of information to be displayed. The format is indicated by the *OutputFormat* data element and the related data structure dedicated to the format of the output, which could be:

- *PredefinedContent*, for displaying one message among a set of predefined messages,
- *OutputText*, for displaying a formatted text, i.e. with format characters along indicating the format.
- *OutputXHTML*, for displaying an XHTML document.
- *OutputBarcode*, for displaying a bar code.

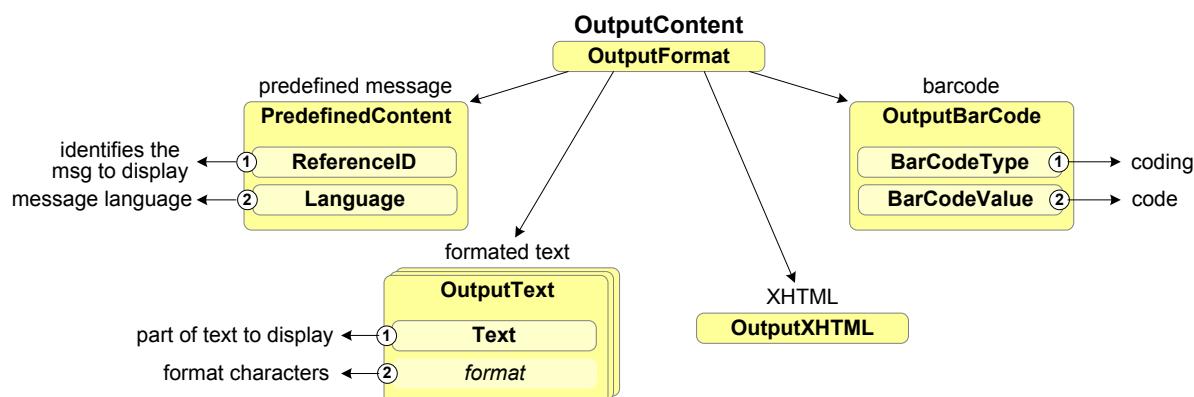


Figure 191: Format of Information to Display or Print

Predefined Message: *PredefinedContent*

A set of predefined messages is configured, the sender of the message has only to send the identifier *ReferenceID* of the message to display. It allows having the exact content of the messages to display under the control of the receiver, the identifier *ReferenceID* in the message representing only the global meaning of the information to display.

In addition, the language of the message (data element *Language*) could be sent along with the identifier.

Formatted Text: *OutputText*

The data structure contains:

- The text to display or print (*Text*) with
- The format of the text specified by format characters:
 - The character encoding (*CharacterSet*), and the font (*Font*),
 - The position of the beginning of the text on the device *StartRow* and *StartColumn*,
 - The color (*Color*),
 - The size of the characters *CharacterWidth* and *CharacterHeight*,
 - The *CharacterStyle* (bold, italic...) and the *Alignment* on the line (left, centred...),
 - If the characters of the displayed *Text* has to be terminated by an end of line (*EndOfLineFlag*)

The data structure *OutputText* could be repeated if the complete text to display contains several lines or contains different formats.

The following example of formatted text:

OutputContent

OutputFormat

 Text

OuputText

 Text

 Please, enter the

OuputText

 Text

 Mileage

 CharacterStyle

 Bold

OuputText

 Text

 as read on the odometer

will display the following text:

“Please, enter the **Mileage** as read on the odometer”

If a text feature is not supported by the receiver of the message, it has to be ignored, i.e. displayed or printed with *Result*="Partial" and *ErrorCondition*="MessageFormat" (see section 4.6.4.3.5 *Unavailable Display Format*).

XHTML Document: *OutputXHTML*

The data structure contains an XHTML document which includes output only, supporting all the functionalities of the formatted text. The content of *OutputXHTML* is the content of the XHTML document body. Images included in the document have to be stored on the receiver.

The following example of formatted text:

OutputContent

OutputFormat	XHTML
OutputXHTML	Please, enter the Mileage as read on the odometer

will display the following text:

“Please, enter the **Mileage** as read on the odometer”

Most of modern XHTML documents use the more flexible format with the CSS (Cascading Style Sheet) defined in the header, than the usage of “inline” style attributes.

The receiver of the *OutputXHTML* has to accept the following tag:

- “b”, “u”, “i”: Text to be rendered as bold, underlined, or italic.
- “h1”, “h2”: Headings (allow use of CSS)
- “small”, “big”, “em”, “strong”: Text to be rendered smaller, larger, emphasized, or strongly emphasized
- “p”, “br”: Paragraph, Break Line
- “pre”: Preformatted text
- “img”: Source images with the attribute “src” to provide the image name.

The receiver of the *OutputXHTML* has to accept the following attributes:

- “class”, to allow use of CSS
- “font-size” with the values “medium”, “smaller”, “larger”.
- “text-align” with the values “left”, “right”, “center”, “justify”.
- “color” with the values “black” “silver” “gray” “white” “maroon” “red” “purple” “fuchsia” “green” “lime” “olive” “yellow” “navy” “blue” “teal” “aqua” (the standard CSS color)

If a tag or attribute feature is not supported by the receiver of the message, it has to be ignored, i.e. displayed or printed with *Result*=“Partial” and *ErrorCondition*=“MessageFormat” (see section 4.6.4.3.5 *Unavailable Display Format*).

BarCode: *OutputBarcode*

This is a barcode to be displayed or printed, with the encoding *BarcodeType*, and the code value *BarcodeValue*.

The following example of barcode:

OutputContent

OutputFormat BarCode

OutputBarcode

BarcodeValue 9782707150189

could be displayed as the following drawing:



Figure 192: Barcode Display Example

4.5.2 Display

4.5.2.1 Processing Overview

The *Display* message pair is used in different environments for various purposes:

- (1) *Display to the Cashier during a POI Transaction*: During the processing of some request from the Sale Terminal, the POI Terminal wants to show some information to the Cashier on a display device managed by the Sale Terminal.
- (2) *Display to the Customer outside a POI Transaction*: The Sale Terminal wants to display information to the Customer when the POI Terminal is not processing a transaction. As in most of the cases, the Customer interface display device is managed by the POI Terminal.
- (3) *Display to the Customer during a POI Transaction*: The POI Terminal uses for its Customer interface a device managed by the Sale Terminal. The POI Terminal uses Device Service as Display to communicate with the Customer.
- (4) *Display to the Cashier outside a POI Transaction*: The Sale Terminal uses for its Cashier interface a display device managed by the POI Terminal. The Sale Terminal uses Device Service as Display to communicate with the Cashier.

These cases are determined by the following data element of the *DisplayOutput* data structure:

- The data element *Device* which indicates the target logical device on which the information have to be displayed: "CashierDisplay" and "CustomerDisplay".
- The data element *InfoQualify* which qualify the information to display: "Display", "Status", "Error" and "POIReplication".

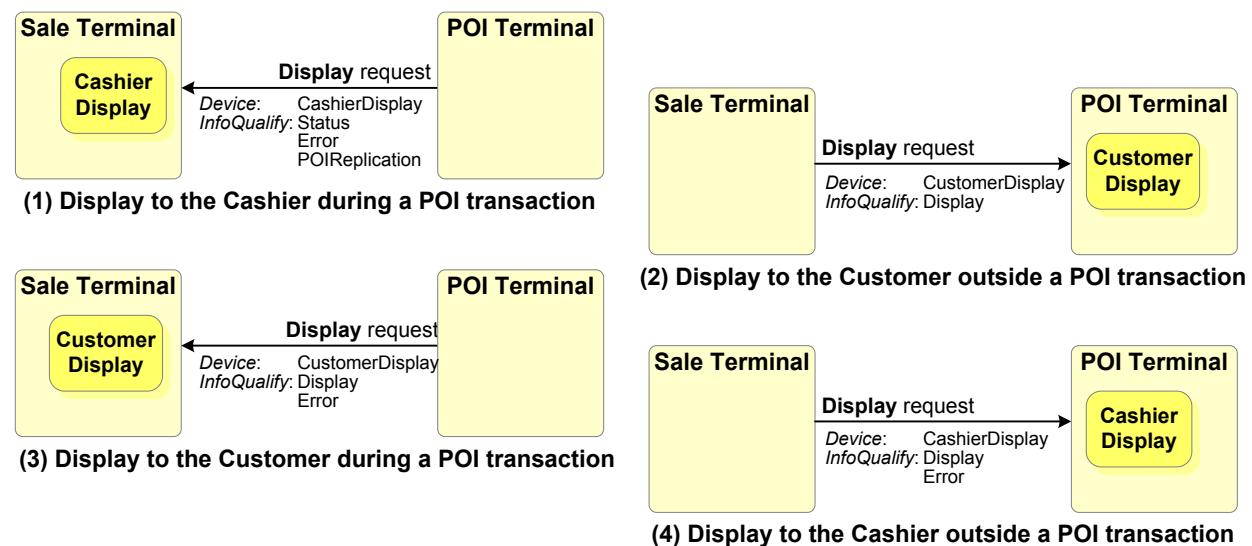


Figure 193: Display Devices and Information Qualify

(1) Display to the Cashier during a POI Transaction

This is the standard case of the display interface that the POI Terminal provides to the Cashier during the processing of an attended transaction.

Information is presented on the logical Device “CashierDisplay” managed by the Sale Terminal with the following value of *InfoQualify*:

- “Status”: the different states of the processing are displayed to the Cashier to communicate the progress of the transaction.
- “Error”: Error situation which has to be known by the Cashier but not necessary by the Customer. For instance when the Acquirer requires to the Merchant to capture the card of the Customer.
- “POIReplication”: The POI replicates to the Cashier what is displayed internally to the Customer by the POI Terminal. The Cashier might assist the Customer conducting the transaction. Depending on the transaction performed by the POI, sensitive information can be not replicated to the Cashier.

(2) Display to the Customer outside a POI Transaction

This is the case of unattended context, when the Sale System displays some information to the Customer before or after a transaction, and the display device is managed by the POI Terminal.

Information is presented on the logical Device “CustomerDisplay” managed by the POI Terminal with only one value of *InfoQualify*:

- “Display”: The Sale System displays some instructions waiting for a Customer on the interface. It could be some instructions as an invitation to enter a card, select an item, to pick up a nozzle.

(3) Display to the Customer during a POI Transaction

This is the case where the Customer interface of the POI Terminal (or at least a part of it for security reasons) is managed by the Sale Terminal. During the processing of a transaction, the POI Terminal uses the *Display* message pair to interface the Customer.

Information is presented on the logical Device “CustomerDisplay” managed by the Sale Terminal with the following value of *InfoQualify*:

- “Display”: the standard information displayed to communicate to the Customer during the transaction.
- “Error”: If the case of the Customer interface uses using a specific device or presentation for error messages.

(4) Display to the Cashier outside a POI Transaction

This is the case where the Cashier interface of the Sale Terminal (or a part of it) is managed by the POI Terminal. During the processing of a Sale transaction and outside a POI transaction, the Sale Terminal uses the *Display* message pair to interface the Cashier.

Information is presented on the logical Device “CashierDisplay” managed by the POI Terminal with the following value of *InfoQualify*:

- “Display”: the standard information displayed to communicate to the Cashier during the transaction.
- “Error”: If the case of the Cashier interface uses using a specific device or presentation for error messages.

4.5.2.2 Presentation of the Display Messages

The Display request message body *DisplayRequest*, contains the following information:

The data structure *DisplayOutput*, containing a complete Display command. This data structure could be repeated to group several display commands in the same message. It includes:

1. The indicator to require or not a Display response message: *ResponseRequiredFlag*, “True” by default.
2. The minimum duration of the display: *MinimumDisplayTime*.
3. The logical device on which the information will be displayed: *Device*.
4. The type of information to display: *InfoQualify*.
5. The content of the information to display in the data structure *OutputContent* (see the section 4.5.1 *Output Content Formats*).
6. Optionally a signature of the content to display: *OutputSignature*.

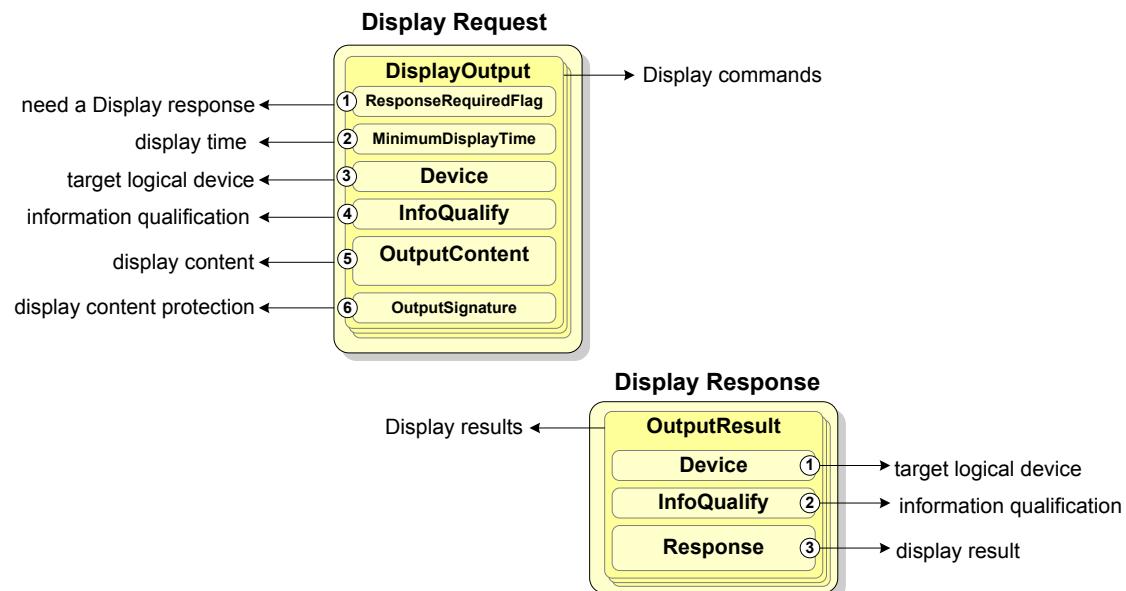


Figure 194: Display Request/Response Information

The Display response message body *DisplayResponse*, contains the following information:

The data structure *OutputResult*, containing the result of a Display command sent in the request message. There is one such data structure per data structure *DisplayOutput* sent in the request and requiring a response. It includes:

1. A copy of the logical device: *Device*.
2. A copy of the type of information: *InfoQualify*.
3. The result of the related Display command: *Response*.

4.5.2.3 Display Request Layout

<i>DisplayRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		“Device”
MessageCategory	[1..1]		Display
MessageType	[1..1]		Request
ServiceID	[0..1]		if requested inside a service
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
DisplayRequest	[1..1]		
DisplayOutput	[1..n]		Complete display content for output devices. At most one DisplayOutput per Device/InfoQualify pair.
ResponseRequiredFlag	[0..1]		default True
MinimumDisplayTime	[0..1]		default 0
Device	[1..1]		CashierDisplay, CustomerDisplay
InfoQualify	[1..1]		Status, Error, Display, POIReplication
OutputContent	[1..1]		
OutputFormat	[1..1]		MessageRef, Text, BarCode, XHTML
PredefinedContent	[0..1]	<i>S from Sale</i>	Mandatory, if OutputFormat is MessageRef Not allowed, otherwise
ReferenceID	[1..1]		
Language	[0..1]		Default language if absent
OutputText	[0..n]	<i>S from POI</i>	Mandatory, if OutputFormat is Text Not allowed, otherwise. One instance of OutputText per shared format
Text	[1..1]	<i>S</i>	
CharacterSet	[0..1]	<i>E</i>	If not present, the settings of the target system or device are used.
Font	[0..1]	<i>E</i>	If not present, the settings of the target system or device are used.
StartRow	[0..1]	<i>E</i>	If not present, the settings of the target system or device are used (e.g. current row position).
StartColumn	[0..1]	<i>E</i>	If not present, the settings of the target system or device are used (e.g. current column position).
Color	[0..1]	<i>E</i>	If not present, default colour used
CharacterWidth	[0..1]	<i>E</i>	If not present, default width used
CharacterHeight	[0..1]	<i>E</i>	If not present, default height used
CharacterStyle	[0..1]	<i>E</i>	If not present, default style used
Alignment	[0..1]	<i>E</i>	If not present, default alignment used
EndOfLineFlag	[0..1]	<i>E</i>	default True
OutputXHTML	[0..1]	<i>E</i>	Mandatory, if OutputFormat is XHTML Not allowed, otherwise
OutputBarcode	[0..1]	<i>E</i>	Mandatory, if OutputFormat is BarCode Not allowed, otherwise
BarcodeType	[0..1]		default EAN13 EAN8, EAN13, UPCA, Code25, Code128, PDF417, QRCode

<i>DisplayRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
BarcodeValue	[1..1]		
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display.

4.5.2.4 Display Response Layout

<i>DisplayResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Display
MessageType	[1..1]		Response
ServiceID	[0..1]		Copy
DeviceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
DisplayResponse	[1..1]		
OutputResult	[1..n]		One per DisplayOutput item of the request, and in the same order.
Device	[1..1]		Copy
InfoQualify	[1..1]		Copy
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>same as PaymentResponse</i>

4.5.2.5 Display Processing

Rule 1: Only the *DisplayResponse.OutputResult* related to the *DisplayRequest.DisplayOutput* components containing *ResponseRequiredFlag* to “True” have to be present.

The *DisplayResponse.OutputResult* components have to be in the same order than the *DisplayRequest.DisplayOutput* components of the related Display request message.

If all the *DisplayRequest.DisplayOutput* contains *ResponseRequiredFlag* to “False”, no Display message response has to be sent.

A *DisplayRequest* message shall contain at most one *DisplayOutput* per *Device/InfoQualify* pair of values.

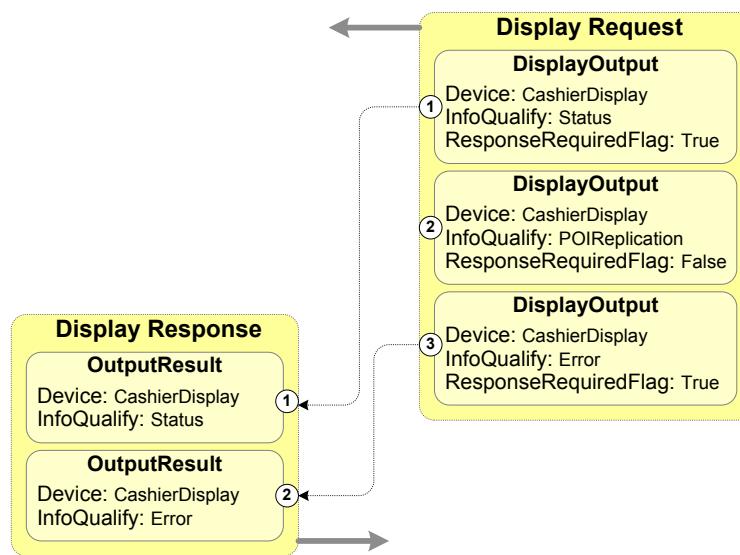


Figure 195: Display Commands Relationships

Rule 2: If *DisplayOutput.MinimumDisplayTime* is greater than zero, the *OutputContent* content has to be displayed at least this period in seconds, but the message response has to be sent immediately.

If a new Display message is sent before the end of this period, the receiver of the message overwrites the current display and send back a Display response with *Result*="Success", if a response is required. Any new command requiring a display on the same device could overwrite before the end of the minimum display time.

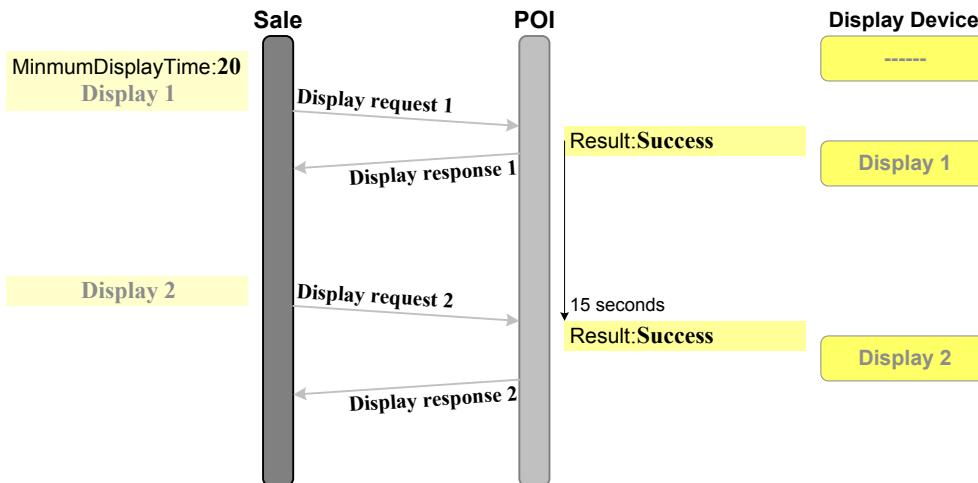


Figure 196: Minimum Display Time Management

Rule 3: If the format of the display content is not supported, the receiver of the message shall send back a Display response with *Result*="Failure" and *ErrorCondition*="UnavailableService" (see section 4.6.4.3.5 *Unavailable Display Format*), if a response is required.

Rule 4: If the *PredefinedContent.ReferenceID* cannot be found, the receiver of the message shall send back a Display response with *Result*="Failure" and *ErrorCondition*="NotFound" (see section 4.6.4.4.2 *Message Not Found*), if a response is required.

Rule 5: If the *PredefinedContent.Language* is not available, the receiver of the message shall use the default language, and send a Display response with *Result*="Partial" and *ErrorCondition*="NotFound" (see section 4.6.4.4.6 *Language Not Supported*), if a response is required.

Rule 6: If formatting requested in *OutputText* is not available by the receiver of the message, it shall display the text without formatting.

4.5.2.6 Error Cases

When the Display request is successfully processed, the Display response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The Display request is received during a Service dialogue or another Device dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

UnavailableDevice

The logical device is not present in the System or the Terminal, the Display request cannot be handled (see section 4.6.3.2 *UnavailableDevice Error*).

UnavailableService

The Display device service is not available in the POI Terminal (see section 4.6.4.3.4 *Unavailable Device Service*).

The format of the display requested in the Display request is not available in the System which manages the display logical device (see section 4.6.4.3.5 *Unavailable Display Format*).

DeviceOut

The display device is temporary or permanently out of service (see section 4.6.3.1.3 *Device Temporary Out* and section 4.6.3.1.4 *Device Permanently Out*).

Busy

The POI or Sale Terminal cannot process a Device Request, because another request is already processed (see section 4.6.5.1.3 *Device Busy*).

NotFound

The information to be displayed in a Display message request has the format “PredefinedContent”, and the identification of the message *ReferenceID* is not found by the receiver of the message. (see section 4.6.4.4.2 *Message Not Found*).

4.5.2.7 Example

This example provides the details of *Figure 195: Display Commands Relationships*.

MessageHeader			(message example 43)
MessageClass	Device		
MessageCategory	Display		
MessageType	Request		
ServiceID	674		
DeviceID	2		
SaleID	SaleTermA		
POIID	POITerm1		
DisplayRequest			
DisplayOutput			
Device	CashierDisplay		
InfoQualify	Status		
OutputContent			
OutputFormat	Text		
OutputText			
Text	Bank Authorization		
DisplayOutput			
ResponseRequiredFlag	False		
Device	CashierDisplay		
InfoQualify	POIReplication		
OutputContent			
OutputFormat	Text		
OutputText			
Text	EMV Payment Processing		
DisplayOutput			
Device	CashierDisplay		
InfoQualify	Error		
OutputContent			
OutputFormat	Text		
OutputText			
Text	Card request online authorisation		

MessageHeader			(message example 44)
MessageClass	Device		
MessageCategory	Display		
MessageType	Response		
ServiceID	674		
DeviceID	2		
SaleID	SaleTermA		
POIID	POITerm1		
DisplayResponse			
OutputResult			
Device	CashierDisplay		
InfoQualify	Status		
Response			
Result	Success		
OutputResult			
Device	CashierDisplay		
InfoQualify	Error		
Response			
Result	Success		

4.5.3 Input

4.5.3.1 Processing Overview

Purpose of the *Input* message pair is to get some information from a user through the Sale to POI protocol.

For requesting information to the user, the *Input* message request contains the component *OutputContent* to be displayed (see section 4.5.1 *Output Content Formats*).

The message request also contains the data element *InputCommand* specifying the kind of requested information and the procedure to obtain it, as summarized in the figure below.

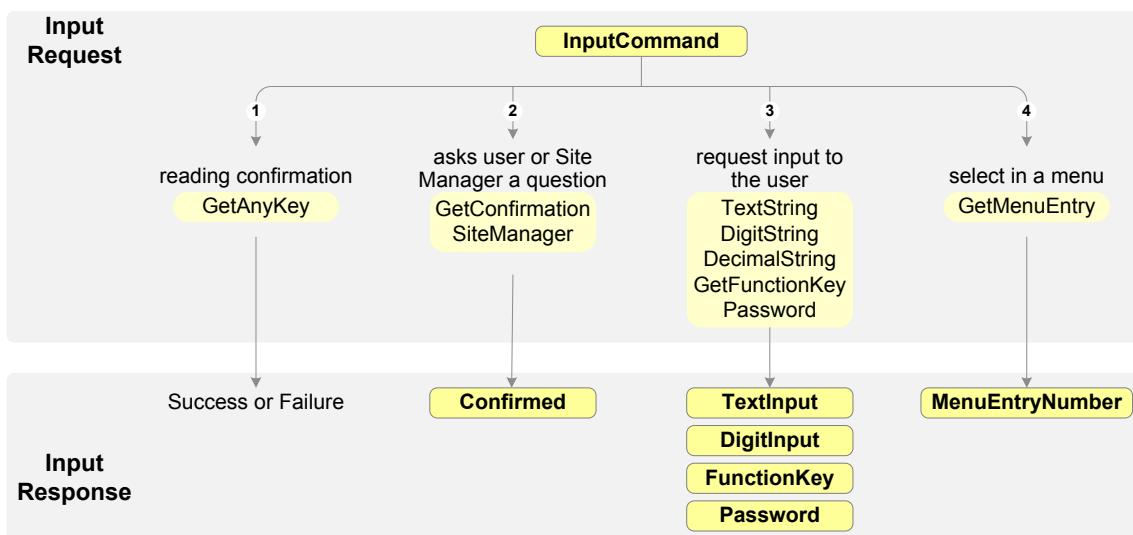


Figure 197: Input Messages Mechanisms

1. *Reading Confirmation* (“*GetAnyKey*”): The Terminal displays the information content *OutputContent* to the user, and wait for a confirmation to be sure that the information has been read. Reception of a successful Input message response gives the reading confirmation. If requested, an immediate response could be sent back, without waiting confirmation from the user.
2. *Asks a Question* (“*GetConfirmation*”, “*SiteManager*”): The Terminal displays a question to the user, and waits for response “Yes” or “No”. “*GetConfirmation*” asks the question to the Cashier or the Customer, and “*SiteManager*” asks the question to the Site Manager. The data element *ConfirmedFlag* of the Input message response gives the answer to the question.
3. *Asks a Question* (“*TextString*”, “*DecimalString*”, “*DigitString*”, “*GetFunctionKey*”, “*Password*”): The Terminal requests an information to the user, and waits for input entering. “*TextString*”, “*DecimalString*”, “*DigitString*”, “*GetFunctionKey*”, and “*Password*” request an alphanumeric character string, a decimal string, a numeric character string, function key, or a password respectively. The data elements *TextInput*, *DigitInput*, *FunctionKey*, and *Password*²⁴ of the Input message response give respectively the user entered input. In addition parameters to enter the string could be specified:
 - The maximum waiting and minimum length of the data could be specified in the request (*MinLength* and *MaxLength* with *MaxDecimalLength* for the decimal part of the decimal string).
 - The end of the character typing when the maximum allowed length is reached (if the user must confirm or not with *WaitUserValidationFlag*).

²⁴ *Password* is sent in the response for a protected password, using the generic CMS data structure *ContentInformationType*. *TextInput* is sent in the response for a plaintext password.

- A string mask to get information requiring a specific format with *StringMask*.
4. *Select an Item in a Menu* (“*GetMenuEntry*”): The Terminal displays the menu provided in the component *OutputContent* for the prompt or the background of the menu, and the component *MenuEntry* for the items of the menu. The user selects an item which is reported in the data element *MenuEntryNumber* of the Input message response.

In addition general parameters for the Input command could be specified in the request:

- The maximum waiting time for entering data, with *MaxInputTime*.
- A default value which could be displayed and overwriting with data, with *DefaultInputString*.
- The notification of an inserted card instead of an input action, with *NotifyCardInputFlag*.
- Length constraints of the entered string, with *MinLength*, *MaxLength*, *MaxDecimalLength*.
- The waiting of user confirmation when the maximum allowed length is reached, with *WaitUserValidationFlag*.
- A default string displayed on the input field before entering the string, with *DefaultInputString*.
- A string mask to get information requiring a specific format, with *StringMask*.
- The display from the right to the left of the entered characters, with *FromRightToLeftFlag*.
- To generate a beep when a key is pressed, with *BeepKeyFlag*.
- The behaviour of the Correct function key, with *GlobalCorrectionFlag*.
- Disabling some function keys, with *DisableCancelFlag*, *DisableCorrectFlag*, *DisableValidFlag*.
- The ability to go back at the upper level or the root of a menu, with *MenuBackFlag*.

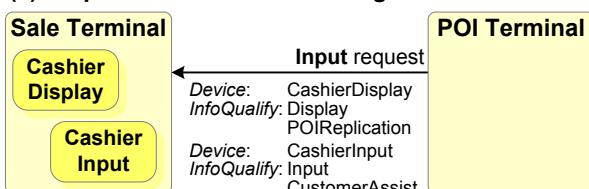
The *Input* message pair is used in different environments:

- (1) *Request Information from the Cashier during a POI Transaction*: During the processing of a service request from the Sale Terminal, the POI Terminal need to request some information from the Cashier using a display/input device pair managed by the Sale Terminal. The Sale terminal handles the Input device request and sends the response to the POI terminal.
- (2) *Request Information from the Customer outside a POI Transaction*: The Sale Terminal requests information from the Customer when the POI Terminal is not being processing a transaction. The Customer interface is managed by the POI Terminal which handles the Input device request and sends the response to the Sale terminal.
- (3) *Request Information from the Customer during a POI Transaction*: The POI Terminal uses for its Customer interface a device managed by the Sale Terminal. The POI Terminal uses Device Service as the Input message pair to communicate with the Customer.
- (4) *Request Information from the Cashier outside a POI Transaction*: The Sale Terminal uses for its Cashier interface an input device managed by the POI Terminal. The Sale Terminal uses Device Service as the Input message pair to communicate with the Cashier.

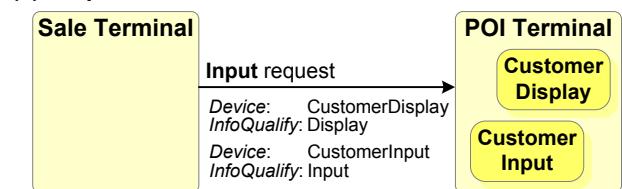
These cases are determined by:

- The following data element of the *DisplayOutput* data structure:
 - The data element *Device* which indicates the target logical device on which the information have to be displayed: "CashierDisplay" and "CustomerDisplay".
 - The data element *InfoQualify* which qualifies the information to display: "Display" and "POIReplication".
- The following data element of the *InputData* data structure:
 - The data element *Device* which indicates the target logical device on which the information have to be entered: "CashierInput" and "CustomerInput".
 - The data element *InfoQualify* which qualifies the information to request: "Input" and "CustomerAssistance".

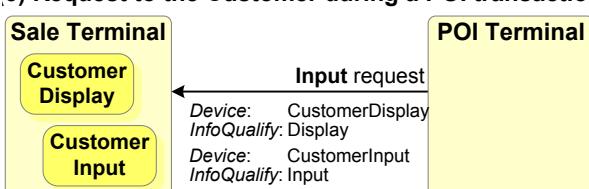
(1) Request to the Cashier during a POI transaction



(2) Request to the Customer outside a POI transaction



(3) Request to the Customer during a POI transaction



(4) Request to the Cashier outside a POI transaction

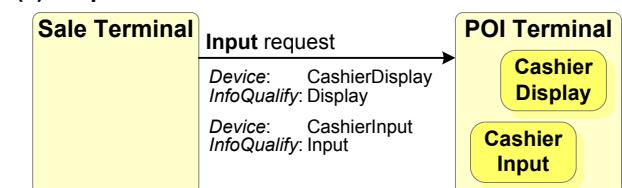


Figure 198: Display/Input Devices and Information Qualify

(1) Request Information from the Cashier During a Transaction

This is the standard case of question or information that the POI Terminal requests from the Cashier during the processing of an attended transaction.

Information is presented on the pair of logical Device “CashierDisplay”/“CashierInput” managed by the Sale Terminal with the following value of *InfoQualify*:

- “Display”/“Input”: to request any Input command directly from the Cashier, or a confirmation to the Site Manager.
- “POIReplication”/“CustomerAssistance”: The POI replicates to the Cashier what is displayed internally to the Customer by the POI Terminal. The Cashier might assist the Customer conducting the transaction, entering some information on behalf of the Customer.

(2) Request Information from the Customer Outside a Transaction

This is the case of unattended context, when the Sale System requests some information from the Customer before or after a transaction, and the display device is managed by the POI Terminal.

Information is presented on the logical Device “CashierDisplay”/“CashierInput” managed by the POI Terminal with only one value of *InfoQualify*:

- “Display”/“Input”: The Sale System requests some information from a Customer on the interface. In addition to the answer, the Sale System could wait for the event of entering a card with the flag *NotifyCardInputFlag* of the Input request.

(3) Request Information from the Customer During a Transaction

This is the case where the Customer interface of the POI Terminal (or at least a part of it for security reasons) is managed by the Sale Terminal.

During the processing of a transaction, the POI Terminal uses the *Input* message pair to interface the Customer with the same elements than the case (2).

(4) Request Information from the Cashier Outside a Transaction

This is the case where the Cashier interface of the Sale Terminal (or a part of it) is managed by the POI Terminal. During the processing of a Sale transaction and outside a POI transaction, the Sale Terminal uses the *Input* message pair to interface the Cashier.

Information is presented on the logical Device “CashierDisplay”/“CashierInput” managed by the POI Terminal with the of *InfoQualify* values “Display”/“Input”.

For some environments, the display device and the input device are not managed by the same system.

In the example (1) of the figure below:

- The keyboard of the POI Terminal is used as the input device of the Customer interface. The POI System uses internal interfaces to request information from the Customer.
- The display device of the Customer interface is managed by the Sale Terminal. The POI Terminal uses Display request messages to display information during the customer input process.

Of course some features of the Input interface are not really optimised, for instance the character by character display of entered information is not very efficient.

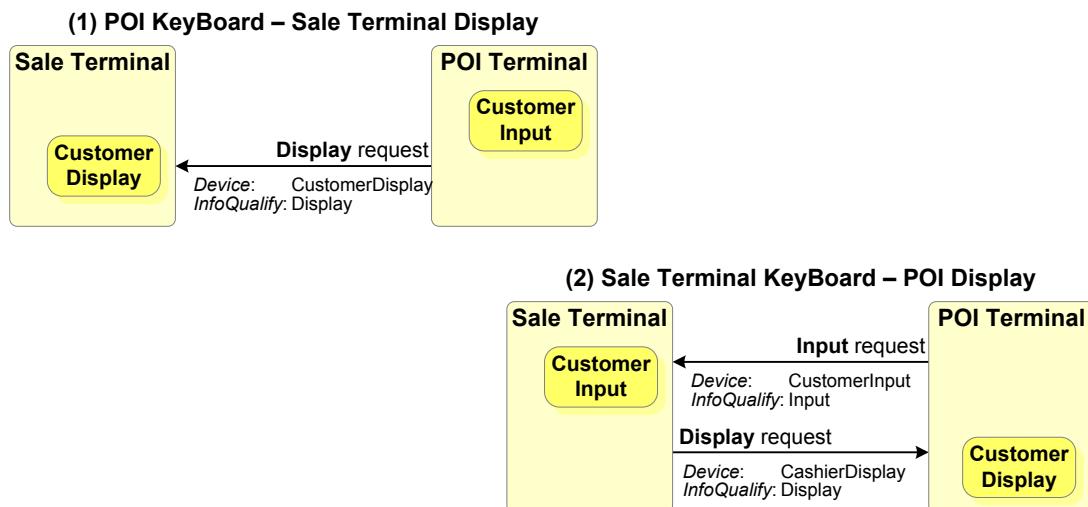


Figure 199: Other Types of Customer Interface Environments

In the example (2) of the figure above:

- The keyboard of the Sale Terminal is used as the input device of the Customer interface. The POI Terminal uses an Input request message to request information from the Customer. This Input request does not include output part, as the display device is managed by the POI Terminal.
- The display device of the Customer interface is managed by the POI Terminal. The POI Terminal uses an internal interface to display information related to the Input request sent to the Sale Terminal. The Sale Terminal uses Display request messages to display information during the customer input process.

This architecture is even less efficient than the first example.

4.5.3.2 Presentation of the Input Messages

The Display request message body *InputRequest*, contains the following information:

1. The data structure *DisplayOutput*, containing the question or the information to display for requesting the input (see the section 4.5.1 *Output Content Formats* for general output format and Rule 14 for the format of menus).
2. The data structure *InputData*, containing the command and its parameters:
 - a. The logical device on which the information will be requested: *Device*.
 - b. The type of information to request: *InfoQualify*.
 - c. The command: *InputCommand* (see previous section).
 - d. An indicator to request notification of the event if a card is entered by the Customer: *NotifyCardInputFlag*.
 - e. Parameters of the input command. (*MaxInputTime*, *ImmediateResponseFlag*, *MinLength*, *MaxLength*, *MaxDecimalLength*, *WaitUserValidationFlag*, *DefaultInputString*, *StringMask*, *BeepKeyFlag*, *GlobalCorrectionFlag*, *DisableCancelFlag*, *DisableCorrectFlag*, *DisableValidFlag*, *MenuBackFlag*)

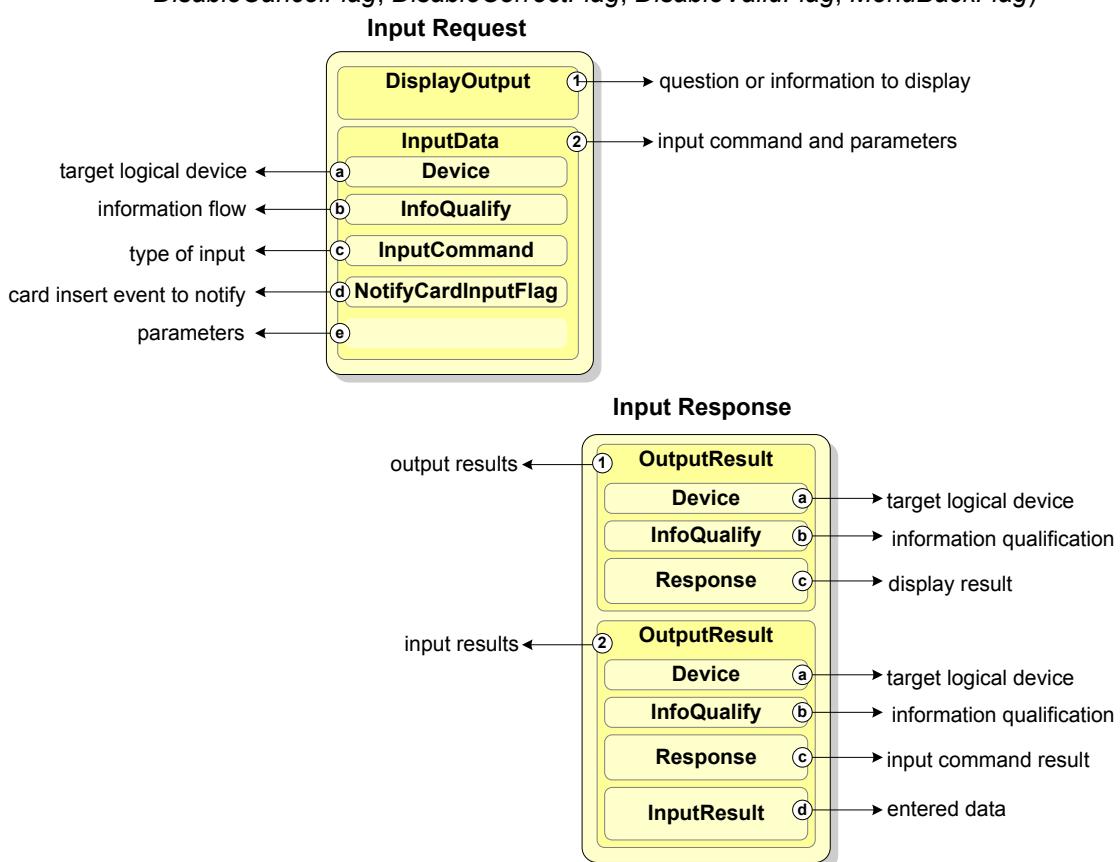


Figure 200: Input Request/Response Information

The Input response message body *InputResponse*, contains the following information:

1. The outcome of the Output request:
 - a. A copy of the output *Device*.
 - b. A copy of the output *InfoQualify*.
 - c. The result of the output: *Response*.
2. The outcome of the Input request:
 - a. A copy of the logical input device: *Device*.

- b. A copy of the input *InfoQualify*.
- c. The result of the input command: *Response*.
- d. The data structure *InputResult* containing the information given by the user related to the input command: *ConfirmedFlag*, *FunctionKey*, *TextInput*, *DigitInput*, *Password*, or *MenuEntryNumber*.

4.5.3.3 Input Request Layout

<i>InputRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		“Device”
MessageCategory	[1..1]		Input
MessageType	[1..1]		Request
ServiceID	[0..1]		if requested inside a service
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
InputRequest	[1..1]		
DisplayOutput	[0..1]		Mandatory if the display device is managed by the receiver.
Device	[1..1]		CashierDisplay, CustomerDisplay
InfoQualify	[1..1]		Display, POIReplication
OutputContent	[1..1]		
OutputFormat	[1..1]		MessageRef, Text, XHTML
PredefinedContent	[0..1]		<i>same as Display</i>
ReferenceID	[1..1]		<i>same as Display</i>
Language	[0..1]		<i>same as Display</i>
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		<i>same as Display</i>
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartRow	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
MenuEntry	[0..n]	S	Mandatory, if InputCommand is GetMenuEntry Not allowed, otherwise One instance of MenuEntry per item to display in the menu at this level
MenuEntryTag	[0..1]		default Selectable
DefaultSelectedFlag	[0..1]		default False.
OutputFormat	[1..1]		MessageRef, Text, XHTML
PredefinedContent	[0..1]		<i>same as Display</i>
ReferenceID	[1..1]		<i>same as Display</i>
Language	[0..1]		<i>same as Display</i>
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		<i>same as Display</i>
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>

<i>InputRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
StartRow	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display.
InputDialog	[1..1]		
Device	[1..1]		CashierInput , CustomerInput
InfoQualify	[1..1]		Input, CustomerAssistance
InputCommand	[1..1]		
NotifyCardInputFlag	[0..1]		default False
MaxInputTime	[0..1]		Time limit for the InputResponse message.
ImmediateResponseFlag	[0..1]		Not allowed if InputCommand is not GetAnyKey. default "True"
MinLength	[0..1]		Not allowed if InputCommand is not TString, DigitsString, DecimalString, Password or GetMenuEntry. Lower or equal to <i>MaxLength</i> .
MaxLength	[0..1]		Not allowed if InputCommand is not TString, DigitsString, DecimalString, Password or GetMenuEntry. Greater or equal to <i>MinLength</i> .
MaxDecimalLength	[0..1]		Not allowed if InputCommand is not DecimalString Greater than <i>MinLength</i> , lower than <i>MaxLength</i> .
WaitUserValidationFlag	[0..1]		default False
DefaultInputString	[0..1]		Not allowed if InputCommand is not TString, DigitsString, DecimalString or Password.
StringMask	[0..1]		Not allowed if InputCommand is not TString, DigitsString, DecimalString or Password.
FromRightToLeftFlag	[0..1]		Not allowed if InputCommand is not TString, DigitsString, DecimalString or Password. default False
MaskCharactersFlag	[0..1]		Not allowed if InputCommand is not TString, DigitsString or Password default False
BeepKeyFlag	[0..1]		default False
GlobalCorrectionFlag	[0..1]		Not allowed if InputCommand is not TString, DigitsString, Password or DecimalString default False
DisableCancelFlag	[0..1]		Not allowed if InputCommand is not GetConfirmation, SiteManager, or GetMenuEntry default False
DisableCorrectFlag	[0..1]		Not allowed if InputCommand is not GetConfirmation, SiteManager, or GetMenuEntry default False
DisableValidFlag	[0..1]		Not allowed if InputCommand is not GetConfirmation, SiteManager, or GetMenuEntry default False

<i>InputRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MenuBackFlag	[0..1]		Allowed for the GetMenuEntry value of InputCommand. default False

4.5.3.4 Input Response Layout

<i>InputResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Input
MessageType	[1..1]		Response
ServiceID	[0..1]		Copy
DeviceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
InputResponse	[1..1]		
OutputResult	[0..1]		If <i>DisplayOutput</i> present in the request.
Device	[1..1]		Copy
InfoQualify	[1..1]		Copy
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>same as PaymentResponse</i>
InputResult	[1..1]		
Device	[1..1]		Copy
InfoQualify	[1..1]		Copy
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>same as PaymentResponse</i>
Input	[0..1]		If Response.Result is Success
InputCommand	[1..1]		Copy
ConfirmedFlag	[0..1]		Mandatory, if InputCommand is GetConfirmation or SiteManager Not allowed, otherwise
FunctionKey	[0..1]		Mandatory, if InputCommand is GetFunctionKey Not allowed, otherwise
TextInput	[0..1]		Mandatory, if InputCommand is TextString or DecimalString Not allowed, otherwise
DigitInput	[0..1]		Mandatory, if InputCommand is DigitString Not allowed, otherwise
Password	[0..1]		Mandatory, if InputCommand is Password Not allowed, otherwise
MenuEntryNumber	[0..n]		Mandatory, if InputCommand is GetMenuEntry Not allowed, otherwise

4.5.3.5 Input Processing

Table below provides the validity of Input request message parameters for each type of input command:

	GetAny Key	GetConfirmation SiteManager	TextString DigitString Password	DecimalString	GetFunctionKey	GetMenuEntry
<i>NotifyCardInputFlag</i>	valid	valid	valid	valid	valid	valid
<i>MaxInputTime</i>	valid	valid	valid	valid	valid	valid
<i>ImmediateResponseFlag</i>	valid					
<i>MinLength</i>			valid	valid		valid
<i>MaxLength</i>			valid	valid		valid
<i>MaxDecimalLength</i>				valid		
<i>WaitUserValidationFlag</i>			valid	valid		
<i>DefaultInputString</i>		valid	valid	valid		
<i>StringMask</i>			valid	valid		
<i>FromRightToLeftFlag</i>			valid	valid		
<i>MaskCharactersFlag</i>			valid			
<i>BeepKeyFlag</i>	valid	valid	valid	valid	valid	valid
<i>GlobalCorrectionFlag</i>			valid	valid		
<i>DisableCancelFlag</i>		valid	valid	valid		valid
<i>DisableCorrectFlag</i>		valid				valid
<i>DisableValidFlag</i>		valid				valid
<i>MenuBackFlag</i>						valid

Table 17: Parameters Validity per Input Command

Rule 1: For the GetAnyKey input command, if *ImmediateResponseFlag* is “True”, the Input response message is sent back immediately, without waiting for the confirmation of the user on what is displayed by the Input request.

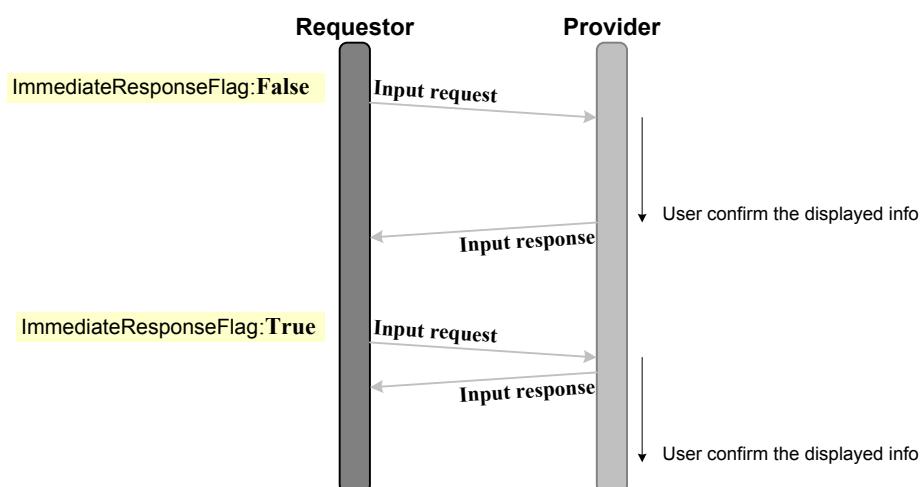


Figure 201: GetAnyKey Input Command Responses

Rule 2: For the *TextString*, *DigitString*, *Password* and *DecimalString* input commands:

If *MinLength* is greater than zero, the string *TextInput*, *DigitInput*, or the decrypted password must have at least this length. The receiver must request at least this length to be entered or cancel the Input.

If *MaxLength* is present, the character string *TextInput*, *DigitInput*, or the decrypted password must have at most this length.

For the *DecimalString* commands, if *MaxDecimalLength* is present, the number of digit after the decimal point (character '.') in *TextInput* or *DigitInput* must be lower or equal to this value.

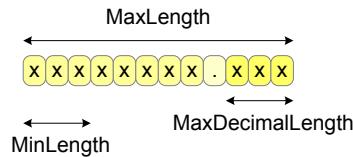


Figure 202: Length Constraint for TextString, DigitString and DecimalString

Rule 3: If *MaxLength* or *MaxDecimalLength* is present, when the user reaches the maximum length of the string or of the decimal part:

If *WaitUserValidationFlag* is False, the entering command shall be automatically stopped.

If *WaitUserValidationFlag* is True, the entering command shall wait for the user confirmation before to be stopped, in order to allow some correction in the entering string.

Rule 4: If the *DefaultInputString* is present and non empty:

For the *TextString*, *DigitString* and *DecimalString* input commands: default string displayed on the input field before entering the string.

For *GetConfirmation* input command: "Y" for yes, "N" for no.

If the *StringMask* is present, the *DefaultInputString* value must contain the separator characters if any.

Rule 5: If the *StringMask* is present for *TextString*, *DigitString* and *DecimalString* input commands, the value of the mask has to compliant with the Input command *TextString*, *DigitString* and *DecimalString* (e.g. for *DigitString*, only digit characters could be entered). If the mask is not consistent, the receiver of the message shall send back an Input response with *Result*="Failure" and *ErrorCondition*="MessageFormat" (see section 4.6.2.1.6 *Unexpected Value*).

Rule 6: If the *StringMask* is present for *TextString*, *DigitString*, *Password* and *DecimalString* input commands, the *TextInput*, *DigitInput*, or the decrypted password value in the Input response message must contain only the characters entered without characters of the mask only displayed.

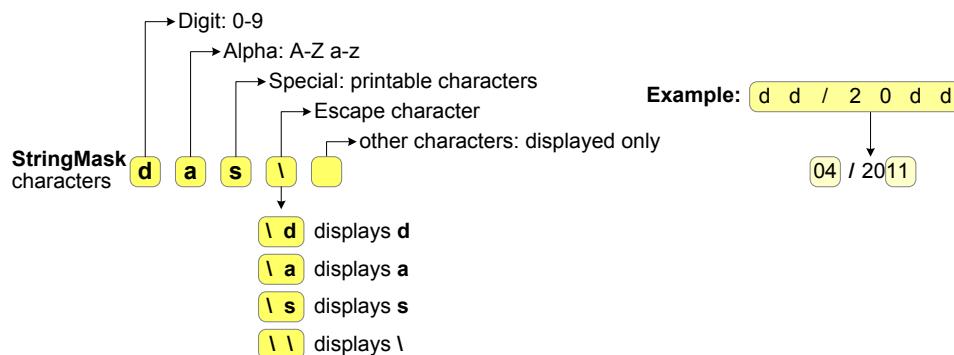


Figure 203: StringMask for Formatted String Input

Rule 7: The user can always cancel the data entering. In this case, the receiver of the message shall send back an Input response with *Result*="Failure" and *ErrorCondition*="Cancel" (see section 4.6.6.2.1 *User Cancellation*).

Rule 8: If the component *MaxInputTime* is present in the Input request message, after this period of time, the input command must automatically be cancelled, and the Input response with *Result*="Failure" and *ErrorCondition*="Cancel" (see section 4.6.6.2.2 *System Cancellation*).

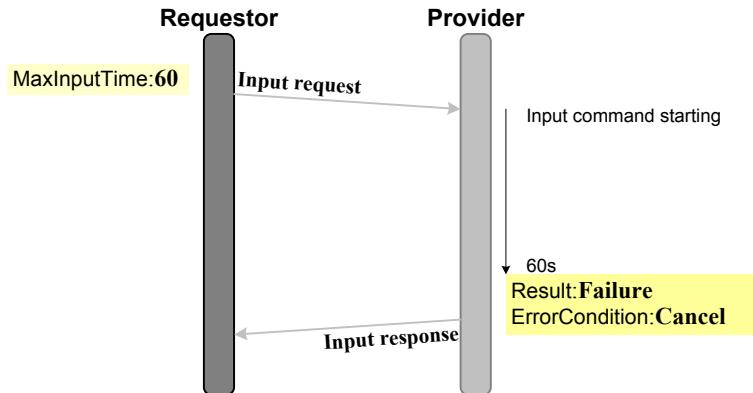


Figure 204: MaxInputTime Limitation

Rule 9: If the input command is not implemented or not feasible (e.g. in an unattended environment), the receiver of the message shall send back an Input response with *Result*="Failure" and *ErrorCondition*="UnavailableService" (see section 4.6.4.3.6 *Unavailable Input Command*).

Rule 10: If the *PredefinedContent.ReferenceID* cannot be found, the receiver of the message shall send back an Output response with *Result*="Failure" and *ErrorCondition*="NotFound" (see section 4.6.4.4.2 *Message Not Found*).

Rule 11: If the *PredefinedContent.Language* is not available, the receiver of the message shall use the default language, and send an Output response with *Result*="Partial" and *ErrorCondition*="NotFound" (see section 4.6.4.4.6 *Language Not Supported*), if a response is required.

Rule 12: If the *NotifyCardInputFlag* is "True" in the *Input* request message, and the Customer inserts a card in the card reader before completing the Input command, the POI send back an Input response with *Result*="Failure" and *ErrorCondition*="InsertedCard" (see section 4.6.5.3 *InsertedCard Error*).

NotifyCardInputFlag is used in unattended environments. For unattended environments, a transaction might start when a Customer manipulates an external interface of the vending machine.

The manipulation could be:

- On a device managed by the Sale Terminal (e.g. pick up the nozzle of a petrol pump),
- An answer to an Input request, because the device for the input command is managed by the POI Terminal (e.g. select an option),
- A card insertion which is managed by an Input request message with the response indicated in the previous rule.

If the Sale System wants to be notified:

- Both from the answer to the input command and a card insertion, it requests the input command and includes the *NotifyCardInputFlag* flag to "True". The Input response message reporting the answer to the input command (request 1 below), or the *ErrorCondition*="InsertedCard" (example 2 below).

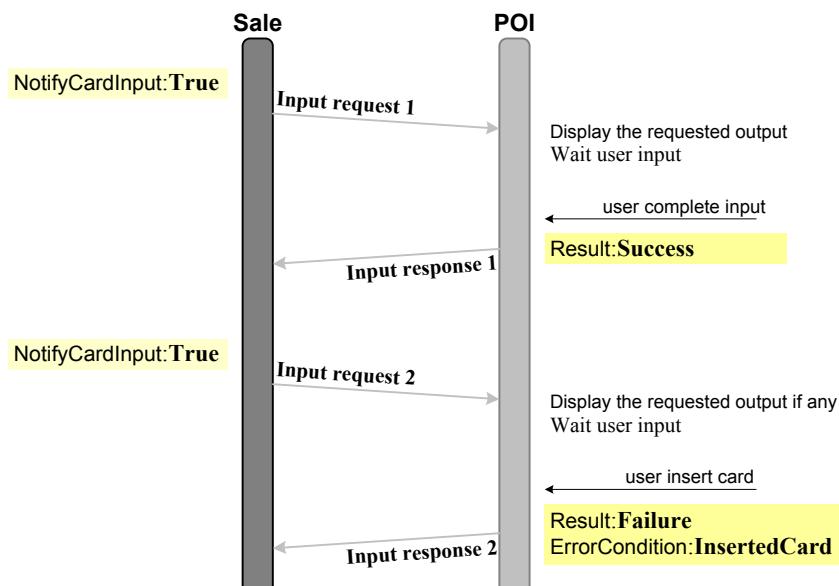


Figure 205: Inserted Card Notification

- Only from card insertion, instead of sending an Input request message, the Sale System waits for an Event notification CardInserted.

Rule 13: The flag *MaskCharactersFlag* has the value "True" when the entered character in an "TextInput", "DigitInput" or "Password" *InputCommand*, are not displayed to the user, but replaced by a standard character as '.'.

It allows to enter user password which are not encrypted as in the "Password" input command, but using the standard "TextInput" or "DigitInput" input commands.

Rule 14: The flag *DisableCancelFlag* (respectively *DisableCorrectFlag*, *DisableValidFlag*) has the value "True" when the Cancel function key (respectively the Correct function key and the Valid function key) cannot be used. The function key is then not active and not displayed to the user.

Rule 15: Get menu entry Input command.

When *InputCommand* has the value "GetMenuEntry", the Input request message must include:

- *OutputContent* which contains a prompt, the background of the menu, or any information to display, related to menu but not to a specific item of the menu,
- An occurrence of *MenuEntry* for each choice of the menu, which contains:
 - *MenuEntryTag* to specify if the menu item is selectable by the user or not, and if the menu item is a submenu of the current menu.
 - *OutputFormat* to indicates the output format of the menu item.
 - *PredefinedContent*, *OutputText* or *OutputXHTML*, depending of the *OutputFormat* value.

The various occurrences of *MenuEntry/OutputFormat* may have different values.

The following example of menu:

```
DisplayOutput
Device          CustomerDisplay
InfoQualify    Display
OutputContent
  OutputFormat  Text
OutputText
  Text          Choose the type of report:
MenuEntry
  OutputFormat  Text
OutputText
  Text          1: Daily Payment Report
MenuEntry
  MenuEntryTag  NonSelectable
  OutputFormat   Text
OutputText
  Text          2: Daily Loyalty Report
MenuEntry
  MenuEntryTag  SubMenu
  OutputFormat   Text
OutputText
  Text          3: Other Reports
```

could be displayed as the following menu:



Figure 206: Menu Display Example

Rule 16: Back/Home browsing functions in a Menu.

The boolean *MenuBackFlag* may be present in the Input request message only for the value "GetMenuEntry" of the *InputCommand*. It has the value:

- "True" when the menu to be displayed is not the home of the menu, i.e. has a parent menu,
- "False" when the menu to be displayed has no parent menu, i.e. this the home of the menu.

If *MenuBackFlag* has the value "True":

- When the "Back" function key is pressed by the user, the Input response message is returned with *MenuEntryNumber* set to the value -1, and a new Input request must be sent with the immediate upper level of the menu.
- When the "Home" function key is pressed by the user, the Input response message is returned with *MenuEntryNumber* set to the value 0, and a new Input request must be sent with the home of the menu.

The "Back" and "Home" function keys may be any function keys, which are clearly identified as such function keys.

Rule 17: Multiple items selection in a Menu (checklist).

When *InputCommand* has the value "GetMenuEntry", several menu entries may be selected if and only if at least one of *MinLength* and *MaxLength* elements is present:

- If present, *MinLength* specifies the minimum number of *MenuEntry* occurrences which have to be selected.
- If *MinLength* is absent, the user may select no *MenuEntry* occurrences.
- If present, *MaxLength* specifies the maximum number of *MenuEntry* occurrences which may to be selected.

In the Input response, one occurrence of *MenuEntryNumber* is present per *MenuEntry* selected. *MenuEntryNumber* values must be in increasing order.

Rule 18: Default or pre-selected items in a Menu.

When *InputCommand* has the value "GetMenuEntry", and one or several occurrences of *MenuEntry* contain *DefaultSelectedFlag* with the value True:

- If only one item of the menu can be selected (see previous Rule), only one occurrence of *MenuEntry* may contain *DefaultSelectedFlag* with the value True. This entry is then the selected entry displayed before any action of the user.
- If several items of the menu can be selected (see previous Rule), several occurrences of *MenuEntry* may contain *DefaultSelectedFlag* with the value True. These entries are then selected entries which are displayed before any action of the user.

4.5.3.6 Error Cases

When the Input request is successfully processed, the Input response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The Display request is received during a Service dialogue or another Device dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

UnavailableDevice

The logical device is not present in the System or the Terminal, the Device request cannot be handled (see section 4.6.3.2 *UnavailableDevice Error*).

UnavailableService

The PIN device service is not available in the POI Terminal (see section 4.6.4.3.4 *Unavailable Device Service*).

The requested input command in the Input request is not available in the System which manages the display logical device (see section 4.6.4.3.6 *Unavailable Input Command*).

The format of the display requested in the Input request is not available in the System which manages the display logical device (see section 4.6.4.3.5 *Unavailable Display Format*).

DeviceOut

The display/input device is temporary or permanently out of service (see section 4.6.3.1.3 *Device Temporary Out* and section 4.6.3.1.4 *Device Permanently Out*).

Busy

The POI or Sale Terminal cannot process the Input request, because another request is already processed on this display/input device (see section 4.6.5.1.3 *Device Busy*).

NotFound

The information to be displayed in a Device message request has the format “PredefinedContent”, and the identification of the message *ReferenceID* is not found by the receiver of the message (see section 4.6.4.4.2 *Message Not Found*).

Cancel

The user has aborted the input command, or a timeout occurs waiting for the user input (see section 4.6.6.2 Cancel Error).

EnterCard

The Input requested a *NotifyCardInputFlag* and the Customer enters a card in the card reader before answering the Input command. The POI has aborted (see section 4.6.5.3 *InsertedCard Error*).

4.5.3.7 Examples

a) Display Confirmation

This is an example of the “GetAnyKey” input command asked to by the Sale System to the Customer to confirm a car wash.

MessageHeader		(message example 45)
MessageClass	Device	
MessageCategory	Input	
MessageType	Request	
DeviceID	1374	
SaleID	SaleTermA	
POIID	POITerm1	
InputRequest		
DisplayOutput		
Device	CustomerDisplay	
InfoQualify	Display	
OutputContent		
OutputFormat	Text	
OutputText		
Text	Confirm the Car Wash	
InputData		
Device	CustomerInput	
InfoQualify	Input	
InputCommand	GetAnyKey	
MessageHeader		(message example 46)
MessageClass	Device	
MessageCategory	Input	
MessageType	Response	
DeviceID	1374	
SaleID	SaleTermA	
POIID	POITerm1	
InputResponse		
InputResult		
Device	CustomerInput	
InfoQualify	Input	
Response		
Result	Success	
Input		
InputCommand	GetAnyKey	

b) Information Request

This is an example of the “DigitString” command asked to by the Sale System to the Customer to enter the mileage of the vehicle.

MessageHeader		(message example 47)
MessageClass	Device	
MessageCategory	Input	
MessageType	Request	
DeviceID	1375	
SaleID	SaleTermA	
POIID	POITerm1	
InputRequest		
DisplayOutput		
Device	CustomerDisplay	
InfoQualify	Display	
OutputContent		
OutputFormat	Text	
OutputText		
Text	Enter the mileage in miles :	
InputData		
Device	CustomerInput	
InfoQualify	Input	
InputCommand	DigitString	
MessageHeader		(message example 48)
MessageClass	Device	
MessageCategory	Input	
MessageType	Response	
DeviceID	1375	
SaleID	SaleTermA	
POIID	POITerm1	
InputResponse		
InputResult		
Device	CustomerInput	
InfoQualify	Input	
Response		
Result	Success	
Input		
InputCommand	DigitString	
DigitInput	43824	

c) Menu Item Selection

This is an example of the “GetMenuEntry” input command asked by the POI Terminal to the Cashier.

MessageHeader		(message example 49)
MessageClass	Device	
MessageCategory	Input	
MessageType	Request	
ServiceID	675	
DeviceID	4	
SaleID	SaleTermA	
POIID	POITerm1	
InputRequest		
DisplayOutput		
Device	CustomerDisplay	
InfoQualify	Display	
OutputContent		
OutputFormat	Text	
OutputText		
Text	Choose the report:	
MenuEntry		
OutputFormat	Text	
OutputText		
Text	1: Daily Payment Report	
MenuEntry		
OutputFormat	Text	
OutputText		
Text	2: Daily Loyalty Report	
MenuEntryTag	NonSelectable	
MenuEntry		
MenuEntryTag	SubMenu	
OutputFormat	Text	
OutputText		
Text	3: Other Reports	
InputData		
Device	CustomerInput	
InfoQualify	Input	
InputCommand	GetMenuEntry	

MessageHeader		(message example 50)
MessageClass	Device	
MessageCategory	Input	
MessageType	Response	
ServiceID	675	
DeviceID	4	
SaleID	SaleTermA	
POIID	POITerm1	
InputResponse		
InputResult		
Device	CustomerInput	
InfoQualify	Input	
Response		
Result	Success	
Input		
InputCommand	GetMenuEntry	
MenuEntryNumber	1	

4.5.4 Input Update

4.5.4.1 Presentation of the Input Update Message

The Input Update message body *InputUpdate*, contains the following components:

1. The reference to the Input to update: *MessageReference*.
2. The display of the Input to update: *OutputContent*.
3. The menu entries to update: *MenuEntry*.
4. The signature of the information to display: *OutputSignature*.

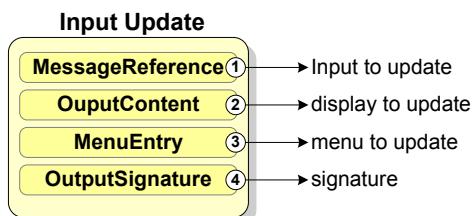


Figure 207: Input Update Information

4.5.4.2 Input Update Layout

<i>InputUpdate Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		“Device”
MessageCategory	[1..1]		InputUpdate
MessageType	[1..1]		Request
ServiceID	[0..1]		if requested inside a service
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
InputUpdate	[1..1]		
MessageReference	[1..1]		Reference to the Input request message to update
ServiceID	[0..1]		Copy if present in the Input header to update.
DeviceID	[0..1]		Copy if present in the Input header to update.
OutputContent	[1..1]		
OutputFormat	[1..1]		Copy
PredefinedContent	[0..1]		<i>same as Display</i>
ReferenceID	[1..1]		<i>same as Display</i>
Language	[0..1]		<i>same as Display</i>
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		<i>same as Display</i>
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartRow	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
MenuEntry	[0..n]	S	Mandatory for InputCommand = GetMenuEntry, otherwise absent.
MenuEntryTag	[0..1]		default Selectable
DefaultSelectedFlag	[0..1]		default False.
OutputFormat	[1..1]		Copy
PredefinedContent	[0..1]		<i>same as Display</i>
ReferenceID	[1..1]		<i>same as Display</i>
Language	[0..1]		<i>same as Display</i>
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		<i>same as Display</i>
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartRow	[0..1]		<i>same as Display</i>

<i>InputUpdate Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display.
MinLength	[0..1]		If present in the Input to update.
MaxLength	[0..1]		If present in the Input to update.
MaxDecimalLength	[0..1]		If present in the Input to update.

4.5.4.3 Input Update Processing

An Input Update message is sent by the initiator of an Input request message when an event occurred on the initiator of the Input request message, which required to change the information displayed to the user answering to the Input command,

Examples of such situation are:

- The Sale System has sent an Input Request to display a menu entry to the customer with the prompt "Select your car wash program".
The customer inserting coins in a cash machine handled by the Sale System and the Sale System wants to update the prompt to display the total amount of inserted coins.
The Sale System sends an Input Update with the prompt "Select your car wash program (credit=5 euros)".
- The Sale System has sent an Input Request to display a menu where the menu items are disabled until the necessary amount of coins are inserted.
The menu entry becomes selectable with Input Update messages, when enough coins are inserted.

Rule 1: The Input Update message must be sent after the Input request message it updates, and before the related Input response message.

The *MessageReference* is mandatory to identify the Input request to update: *ServiceID* and *DeviceID* are a copy of the same field of the Input request header to update.

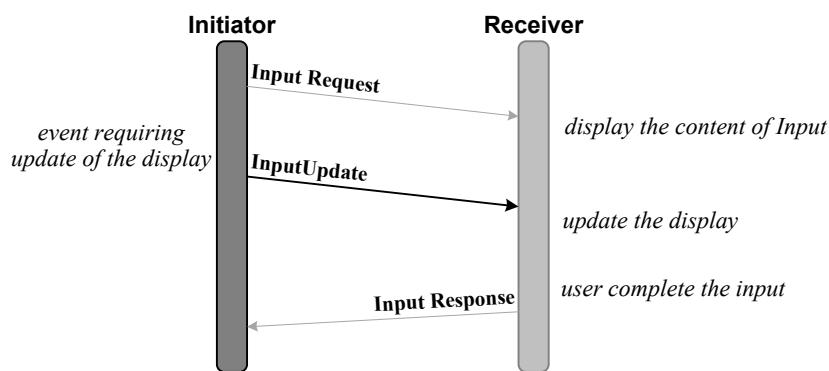


Figure 208: Input Update Sequence Flow

Rule 2: When the Input Update message is received after the Input response message it want to update, the Input Update message is ignored.

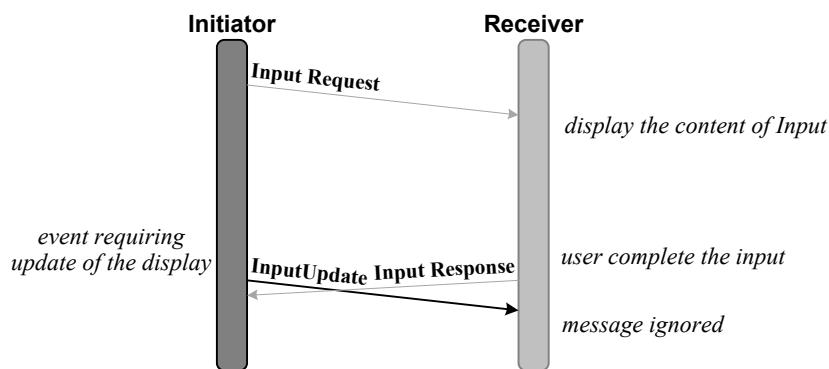


Figure 209: Too Late Input Update

When the *ServiceID* and *DeviceID* of *MessageReference* of the Input Update are different from the Input request in progress, the Input Update is ignored.

In particular when the Input Update is received during the second Input processing of two successives Input exchanges.

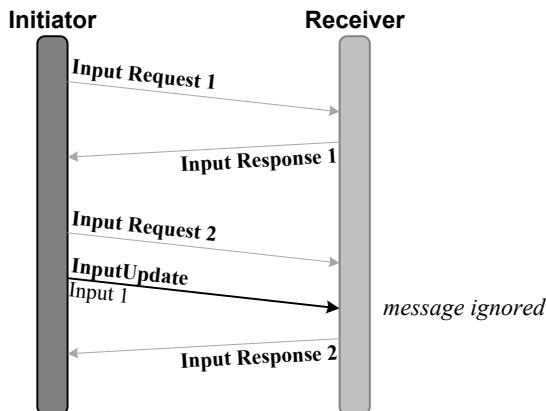


Figure 210: Incorrect Input Message Reference

- Rule 3: *InputUpdate.OutputContent* is mandatory, even if the value is not updated. The *OutputFormat* must be a copy of the Input request message to update. If the format is different, the Input Update is ignored.
- Rule 4: The information contained in the *InputData* data structure of the Input request to update apply in the update at the exception of:
- If *MinLength* was present in the Input request, it must be present in the Input Update, even if the value has not changed.
 - If *MaxLength* was present in the Input request, it must be present in the Input Update, even if the value has not changed.
 - If *MaxDecimalLength* was present in the Input request, it must be present in the Input Update, even if the value has not changed.
- Rule 5: If the *InputCommand* of the Input request message to update is "GetMenuEntry":
- The *InputUpdate.OutputContent* must be present even if the prompt of the menu is not updated, and the previous rule applies.
 - The *InputUpdate.MenuEntry* must be present even if no menu items are updated.
 - For each occurrence of *InputUpdate.MenuEntry*:
 - The number of occurrences must be the same than the Input to update, otherwise the Input Update must be ignored,
 - The *OutputFormat* must have the same value than the related menu entry in the Input to update, otherwise the Input Update must be ignored,
 - The remaining fields *MenuEntryTag*, *DefaultSelectedFlag*, *PredefinedContent*, and *OutputXHTML* are present even if they have the same value than the Input to update.
 - When some of these remaining fields are present, if they are inconsistent the Input Update must be ignored (see section 4.5.3.5 *Input Processing*), otherwise the related menu entry display is updated.

- Rule 6: During the processing of an Input request by the POI system, when the POI sends of an Event because of a new language chosen by the customer (`EventToNotify = "CustomerLanguage"`), or a key that is pressed requires a display update (`EventToNotify = "KeyPressed"`), the Sale system need to send an Input Update to update the display accordingly.

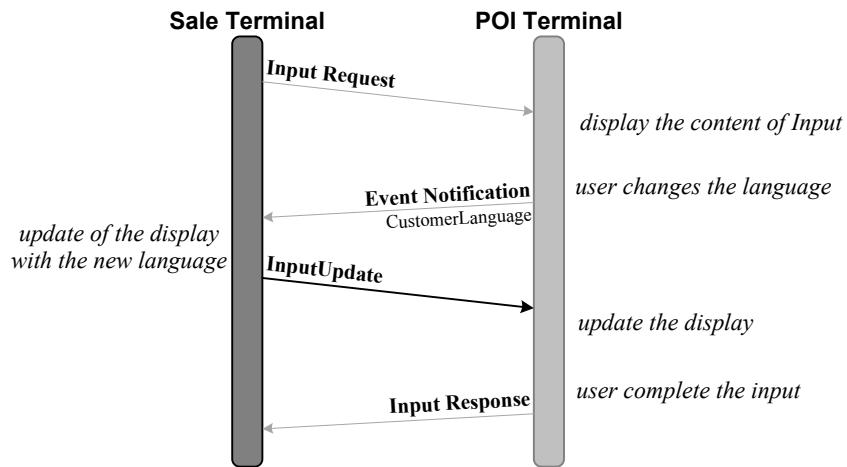


Figure 211: Language Input Update

4.5.4.4 Example

MessageHeader		(message example 51)
MessageClass	Device	
MessageCategory	InputUpdate	
MessageType	Request	
DeviceID	260	
SaleID	SaleTermA	
POIID	POITerm1	
InputUpdate		
MessageReference		
DeviceID	259	
OutputContent		
OutputFormat	Text	
OutputText		
Text	Select your car wash program (credit=5 euros)	

4.5.5 Print

4.5.5.1 Processing Overview

The *Print* message pair is used in different environments:

- (1) *Print a document on the Sale Terminal during a POI Transaction*: During the processing of a request from the Sale Terminal, the POI Terminal must print a document on a printer managed by the Sale Terminal.
- (2) *Print a document on the POI Terminal outside a POI Transaction*: The printer used by the Sale Terminal is managed by the POI Terminal.

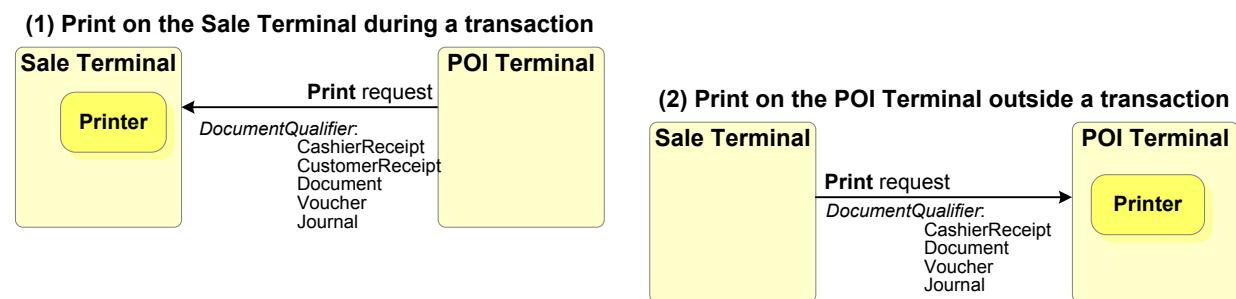


Figure 212: Printer Location

4.5.5.2 Presentation of the Print Messages

The Print request message body *PrintRequest*, contains the data structure *PrintOutput*, including the document to print with the printing parameters:

1. The type of document to print: *DocumentQualifier* (CashierReceipt, CustomerReceipt, Document, Voucher, Journal).
2. The type of response to the Print request message: *ResponseMode* (no response, immediate response, or response after the print ending).
3. For a receipt to print, an indicator to integrate the POI receipt to the Sale receipt: *IntegratedPrintFlag*.
4. For a receipt to print, the flag: *RequiredSignatureFlag*, to indicate that the receipt requires a physical signature by the Customer.
5. The content of the information to print (see the section 4.5.1 Output Content Formats) *OutputContent*, restricted to the format: Text, XHTML and Barcode.

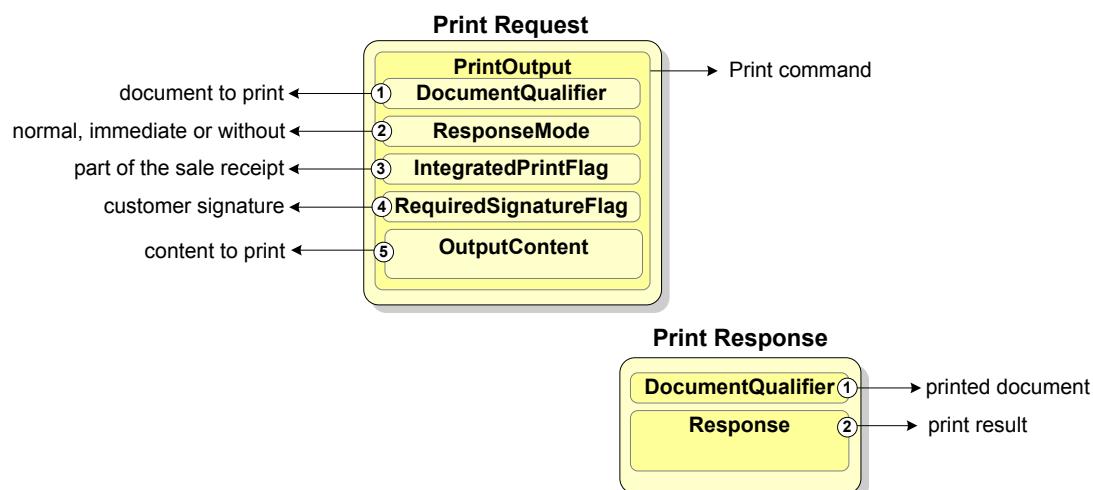


Figure 213: Print Request/Response Information

The Print response message body *PrintResponse*, contains the following information:

1. A copy of the type of document to print: *DocumentQualifier*.
2. The outcome of the Print request: *Response*.

4.5.5.3 Print Request Layout

<i>PrintRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		“Device”
MessageCategory	[1..1]		Print
MessageType	[1..1]		Request
ServiceID	[0..1]		if requested inside a Service dialogue
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
PrintRequest	[1..1]		
PrintOutput	[1..1]		
DocumentQualifier	[1..1]		
ResponseMode	[1..1]		
IntegratedPrintFlag	[0..1]		default False. Not allowed if DocumentQualifier is not “CashierReceipt” or “CustomerReceipt”.
RequiredSignatureFlag	[0..1]		default False.
OutputContent	[1..1]		
OutputFormat	[1..1]		MessageRef, Text, BarCode, XHTML
PredefinedContent	[0..1]		<i>same as Display</i>
ReferenceID	[1..1]		<i>same as Display</i>
Language	[0..1]		<i>same as Display</i>
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		<i>same as Display</i>
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
EndOfLineFlag	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
OutputBarcode	[0..1]		<i>same as Display</i>
BarcodeType	[0..1]		<i>same as Display</i>
BarcodeValue	[1..1]		<i>same as Display</i>

4.5.5.4 Print Response Layout

<i>PrintResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Print
MessageType	[1..1]		Response
ServiceID	[0..1]		Copy
DeviceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
PrintResponse	[1..1]		
DocumentQualifier	[1..1]		Copy
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>same as PaymentResponse</i>

4.5.5.5 Print Processing

Most of the transactions generate one or several documents which are printed on a printer owned by the POI Terminal or the Sale Terminal.

Mode of Response to the Print request messages

To allow various kinds of synchronisations between systems for document printing, the Print message request contains in the *ResponseMode* data component, how the Print message response has to be sent:

- (1) "PrintEnd": the Print response message is sent at the end of the print, when the sender has to ensure the successful end of the printing.
- (2) "Immediate": the Print response message is sent after taking into account the print command. The sender wants its printing request acknowledged, but the failure during the printing could be resolved in another way.
- (3) "NotRequired": the Print response message is not sent.

The figure below presents these modes, taking the example of a Print request from a POI Terminal.

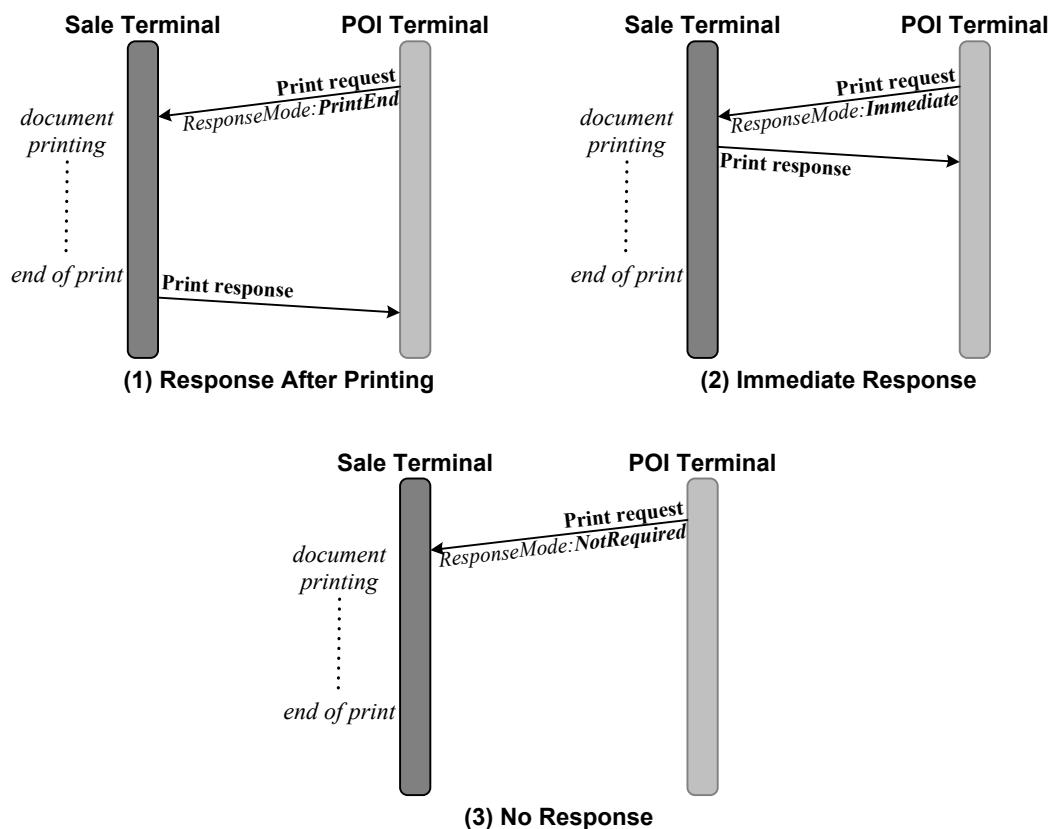


Figure 214: Print Response Message Modes

Rule 1: The Print request messages are executed in the order of the message reception, without any kind of priority.

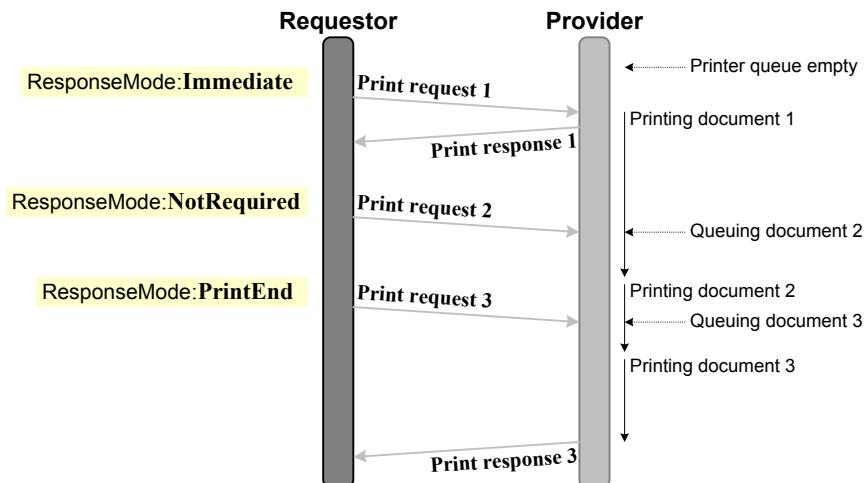


Figure 215: Print Processing Sequence Example

Rule 2: If the *ResponseMode* is “NotRequired” or “Immediate” and the print buffer is full, the receiver of the message shall send back a Print response with *Result*=“Failure” and *ErrorCondition*=“Busy” (see section 4.6.5.1.3 *Device Busy*).

Rule 3: The three *ResponseMode* "NotRequired", "Immediate" and "PrintEnd" have to be supported by all implementations of the Print request message.

Type of Documents to Print

The type of document to print is carried out by the *DocumentQualifier* component. It could have an influence about the physical printer or the type of paper to use.

Rule 4: If the *DocumentQualifier* is not supported by the receiver, the Print response message shall contain *Result*=“Failure” and *ErrorCondition*=“UnavailableService” (see section 4.6.4.3.7 *Unavailable Printing Mode*).

Among the types of document to print specified by *DocumentQualifier*, the “Receipt” document supports dedicated services. There are two types of receipts:

1. The *Sale Receipt*, which contains all the details of the sale,
2. The *POI Receipt(s)*, which contains all the details of the transaction realised by the POI: payment receipt, loyalty receipt or other receipt.

Rule 5: For a Sale printer, the values “CashierReceipt” and “CustomerReceipt” of *DocumentQualifier* represent the POI receipt copies for the Cashier and the Customer respectively. For a POI printer, the values “SaleReceipt” of *DocumentQualifier* represent the Sale receipt for the Cashier.

Depending on the card brand rules, and the implementation of the Sale application:

- The POI receipt could be printed separately. The POI (or the Sale) has in charge the layout of the receipt and the validation of the printing result. The Sale System (or the POI System) must print this transaction receipt without any change, and independently of the sale receipt.
- The POI receipt is integrated to the Sale Receipt by the application. This integration is required when the component *IntegratedPrintFlag* has the value "True" in the *Print* request message.

If the component *IntegratedPrintFlag* has the value "True" in the Print request message, a *ResponseMode* value "PrintEnd" is considered as "Immediate".

4.5.5.6 Error Cases

When the Print request is successfully processed, the Print response message, if present, gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The Print request is received during a Service dialogue or another Device dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

UnavailableDevice

The printer adapted to the type of document to print is not present in the System or the Terminal, the Device request cannot be handled (see section 4.6.3.2 *UnavailableDevice Error*).

UnavailableService

The print device service is not available in the Sale or POI Terminal (see section 4.6.4.3.4 *Unavailable Device Service*).

The printing mode, *ResponseMode* or *DocumentQualifier*, is not supported by the printer or the driver (see section 4.6.4.3.7 *Unavailable Printing Mode*).

DeviceOut

The printer device is temporary or permanently out of service (see section 4.6.3.1.3 *Device Temporary Out* and section 4.6.3.1.4 *Device Permanently Out*).

Busy

The POI or Sale Terminal cannot process the Print request, because one or several requests are already processed on this print device and the print buffer is full (see section 4.6.5.1.1 *Component Unavailable*).

4.5.5.7 Examples

This is an example of payment receipt to print.

MessageHeader		(message example 52)
MessageClass	Device	
MessageCategory	Print	
MessageType	Request	
ServiceID	675	
DeviceID	1375	
SaleID	SaleTermA	
POIID	POITerm1	
PrintRequest		
PrintOutput		
DocumentQualifier	CustomerReceipt	
ResponseMode	PrintEnd	
OutputContent		
OutputFormat	XHTML	
OutputXHTML	<big>Papadeaux Seafood Kitchen</big><p>1079 Davis Dr. - Alpharetta GA 30022</p><p>12:54:07 04/30/2009</p><p>Visa *****5973</p><big><p>Amount: \$18.00</p><p>Auth Code: 237162</p></big>><p>Customer Copy</p>	

The printed receipt looks like below:

Papadeaux Seafood Kitchen

1079 Davis Dr. - Alpharetta GA 30022
12:54:07 04/30/2009
Visa *****5973

Amount: \$18.00

Auth Code: 237162

Customer Copy

As requested, the response message is sent at the end of the print.

MessageHeader		(message example 53)
MessageClass	Device	
MessageCategory	Print	
MessageType	Response	
ServiceID	675	
DeviceID	1374	
SaleID	SaleTermA	
POIID	POITerm1	
PrintResponse		
DocumentQualifier	CustomerReceipt	
Response		
Result	Success	

4.5.6 Sound

4.5.6.1 Usage Overview

The *Sound* message pair is used in different environments for various purposes:

- (1) *Play a sound to the Cashier during a POI Transaction*: During the processing of some request from the Sale Terminal, the POI Terminal wants to play a sound to the Cashier on a display or input device managed by the Sale Terminal.
- (2) *Play a sound to the Customer outside a POI Transaction*: The Sale Terminal wants to play a sound to the Customer when the POI Terminal is not being processing a transaction. As in most of the cases, the Customer interface display or input device is managed by the POI Terminal.
- (3) *Play a sound to the Customer during a POI Transaction*: The POI Terminal uses for its Customer interface a device managed by the Sale Terminal.
- (4) *Play a sound to the Cashier outside a POI Transaction*: The Sale Terminal uses for its Cashier interface a display or input device managed by the POI Terminal.

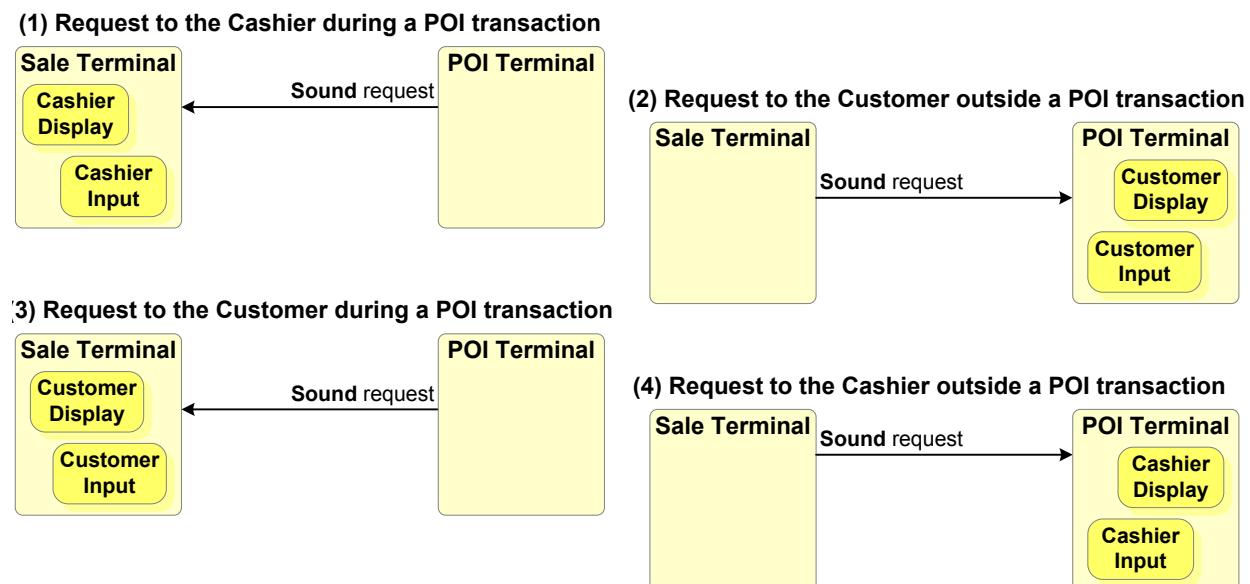


Figure 216: Sound Usage

4.5.6.2 Presentation of the Sound Messages

The Sound request message body *SoundRequest*, contains the following information:

1. The way to answer to the Sound request message: *ResponseMode*, “NotRequired” by default.
2. The type of action on the sound to play: *SoundAction*.
3. The volume of the sound to play: *SoundVolume*.
4. The sound to play: *SoundContent*.

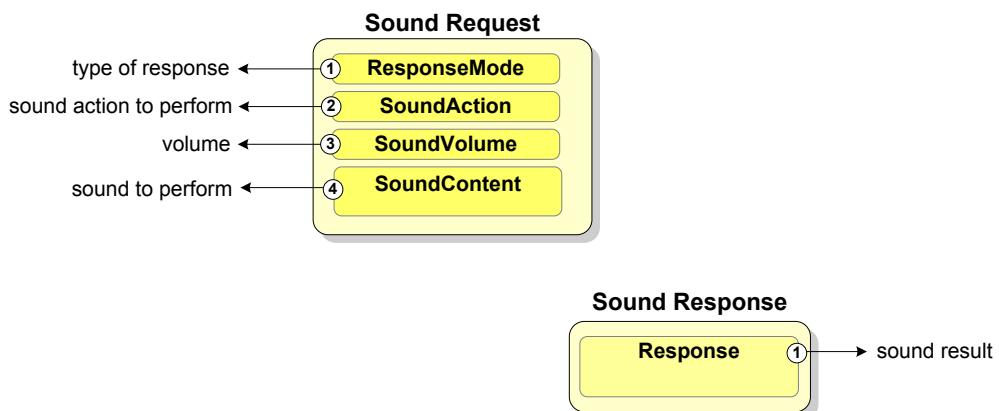


Figure 217: Sound Request/Response Information

The Sound response message body *SoundResponse*, contains the following information:

1. The result of the related Sound action: *Response*.

4.5.6.3 Sound Request Layout

<i>SoundRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		“Device”
MessageCategory	[1..1]		Sound
MessageType	[1..1]		Request
ServiceID	[0..1]		if requested inside a service
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
SoundRequest	[1..1]		
ResponseMode	[0..1]		default NotRequired. Allowed values: NotRequired, Immediate, SoundEnd
SoundAction	[1..1]		StartSound, StopSound, SetDefaultVolume
SoundVolume	[0..1]		Mandatory if SoundAction is SetDefaultVolume
SoundContent	[0..1]		Absent if SoundAction is SetDefaultVolume, otherwise mandatory.
SoundFormat	[1..1]		SoundRef, MessageRef, Text
Language	[0..1]		
ReferenceID	[0..1]		Mandatory if SoundFormat is SoundRef or MessageRef
Text	[0..1]		Mandatory if SoundFormat is Text

4.5.6.4 Sound Response Layout

<i>DisplayResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Sound
MessageType	[1..1]		Response
ServiceID	[0..1]		Copy
DeviceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
SoundResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>same as PaymentResponse</i>

4.5.6.5 Sound Processing

- Rule 1: If *ResponseMode* is equal to "NotRequired", no Sound response message has to be sent.
 If *ResponseMode* is equal to "Immediate", a Sound response message must be sent at the beginning of the generated sound.
 If *ResponseMode* is equal to "SoundEnd", a Sound response message must be sent at the end of the generated sound.

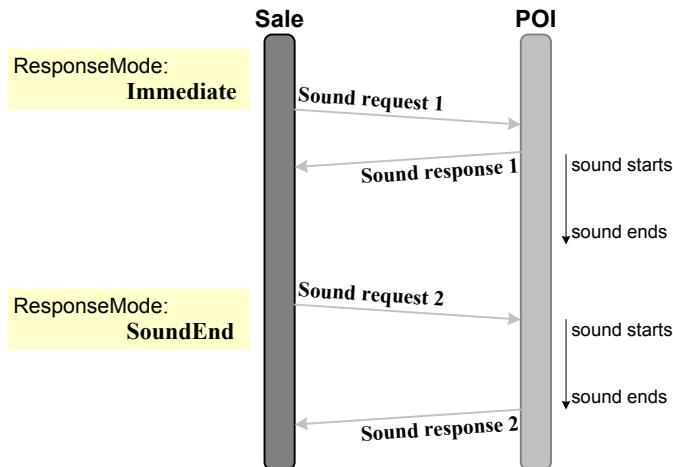


Figure 218: Sound Response Mode

- Rule 2: When *SoundAction* = "SetDefaultVolume", *SoundVolume* is mandatory, *SoundContent* must be absent. If not respected, the receiver of the message shall send back a Sound response message with *Result*="Failure" and *ErrorCondition*="MessageFormat" (see section 4.6.2.1 *Message Format*).

If the receiver of the Sound request message is configured to deliver specific sound volumes per period of time, the Sound request message is ineffective, and a Sound response with *Result*="Success" is sent if required by *ResponseMode*.

- Rule 3: When *SoundAction* = "StartSound":

- If *SoundVolume* is present, the sound must be played at this volume level.
- *SoundContent* must be present. If not respected, the receiver of the message shall send back a Sound response with *Result*="Failure" and *ErrorCondition*="MessageFormat" (see section 4.6.2.1 *Message Format*).

If the receiver of the Sound request message is configured to deliver specific sound volumes per period of time, the Sound request message is ineffective, and a Sound response with *Result*="Success" is sent if required by *ResponseMode*.

- Rule 4: If the format *SoundFormat* of *SoundContent* is not supported, the receiver of the message shall send back a Sound response with *Result*="Failure" and *ErrorCondition*="UnavailableService" (see section 4.6.4.3.9 *Unavailable Sound Format*), if a response is required.

- Rule 5: If *SoundContent.Language* is absent from the Sound request with the *SoundFormat* values "MessageRef" and "Text", the last customer language defined by the Sale system or the POI system must be used.

If the language is not available, the receiver of the message shall use the default language,

and send a Sound response with *Result*="Failure" and *ErrorCondition*="NotFound" (see section 4.6.4.4.6 *Language Not Supported*).

Rule 6: If the *SoundContent.ReferenceID* cannot be found, the receiver of the message shall send back a Sound response with *Result*="Failure" and *ErrorCondition*="NotFound" (see section 4.6.4.4.2 *Message Not Found*), if a response is required.

Rule 7: When *SoundAction* = "StopSound", *SoundVolume* and *SoundContent* must be absent. The sound playing in progress must be stopped.

4.5.6.6 Error Cases

When the Sound request is successfully processed, the Sound response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The Sound request is received during a Service dialogue or another Device dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

UnavailableDevice

The logical device is not present in the System or the Terminal, the Sound request cannot be handled (see section 4.6.3.2 *UnavailableDevice Error*).

UnavailableService

The Sound device service is not available in the POI Terminal (see section 4.6.4.3.4 *Unavailable Device Service*).

The format of the display requested in the Sound request is not available in the System which manages the display logical device (see section 4.6.4.3.5 *Unavailable Display Format*).

DeviceOut

The sound device is temporary or permanently out of service (see section 4.6.3.1.3 *Device Temporary Out* and section 4.6.3.1.4 *Device Permanently Out*).

Busy

The POI or Sale Terminal cannot process a Device Request, because another request is already processed (see section 4.6.5.1.3 *Device Busy*).

NotFound

The information to be displayed in a Sound message request has the format “PredefinedContent”, and the identification of the message *ReferenceID* is not found by the receiver of the message. (see section 4.6.4.4.2 *Message Not Found*).

4.5.6.7 Example

This is the example of a Sale terminal "SaleTermA" requesting to the POI terminal POITerm1 to play a sound file referenced "ambiance02.mp3".

MessageHeader		(message example 54)
MessageClass	Device	
MessageCategory	Sound	
MessageType	Request	
DeviceID	259	
SaleID	SaleTermA	
POIID	POITerm1	
SoundRequest		
ResponseMode	Immediate	
SoundAction	StartSound	
SoundVolume	50	
SoundContent		
SoundFormat	SoundRef	
ReferenceID	ambiance02.mp3	
MessageHeader		(message example 55)
MessageClass	Device	
MessageCategory	Sound	
MessageType	Response	
DeviceID	259	
SaleID	SaleTermA	
POIID	POITerm1	
SoundResponse		
Response		
Result	Success	

4.5.7 PIN

4.5.7.1 PIN Processing

The P/N Device service is provided by a POI Terminal to a Sale Terminal for PIN checking of local cards managed by the Sale System owned by a Customer or a Cashier.

It allows:

1. Acquisition and encryption of the PIN by the POI Terminal,
2. PIN verification by the card or the POI Terminal, or
3. Acquisition and verification of the PIN.

The devices and the PIN services are always located on the POI Terminal, and the PIN request is always made by the Sale Terminal outside a transaction.

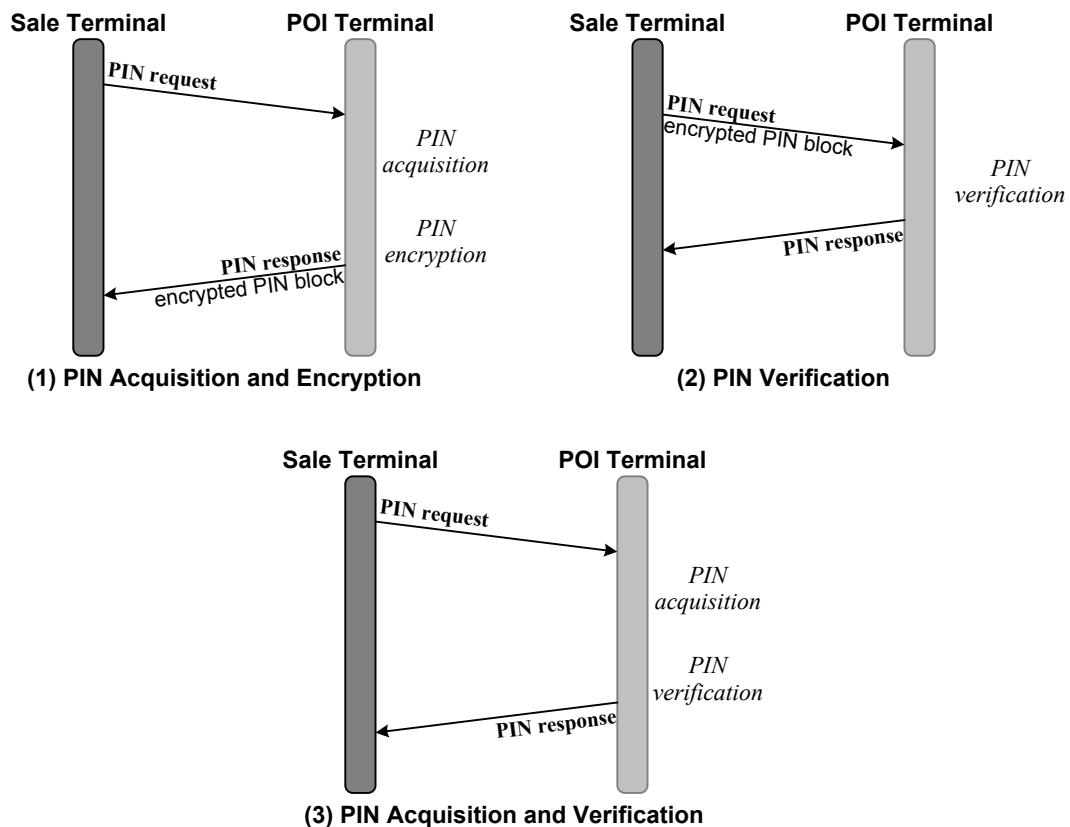


Figure 219: PIN Exchange Overview

4.5.7.2 Presentation of the PIN Messages

The PIN request message body *PINRequest*, contains the following components:

1. The request: *PINRequestType*, “PINEnter” for acquisition and encryption, “PinVerifyOnly” for checking, or “PinVerify” acquisition and checking).
2. The PIN verification algorithm and key: *PINVerifMethod*.
3. Additional input to PIN verification algorithm: *AdditionalInput*.
4. Algorithm to use for encrypting the PIN: *PINEncAlgorithm*.
5. The format of the PIN block before encryption: *PINFormat*.
6. The identification of the PIN encryption key: *KeyReference*.
7. The maximum time to enter the PIN: *MaxWaitingTime*.
8. The encrypted PIN: *CardholderPIN*.

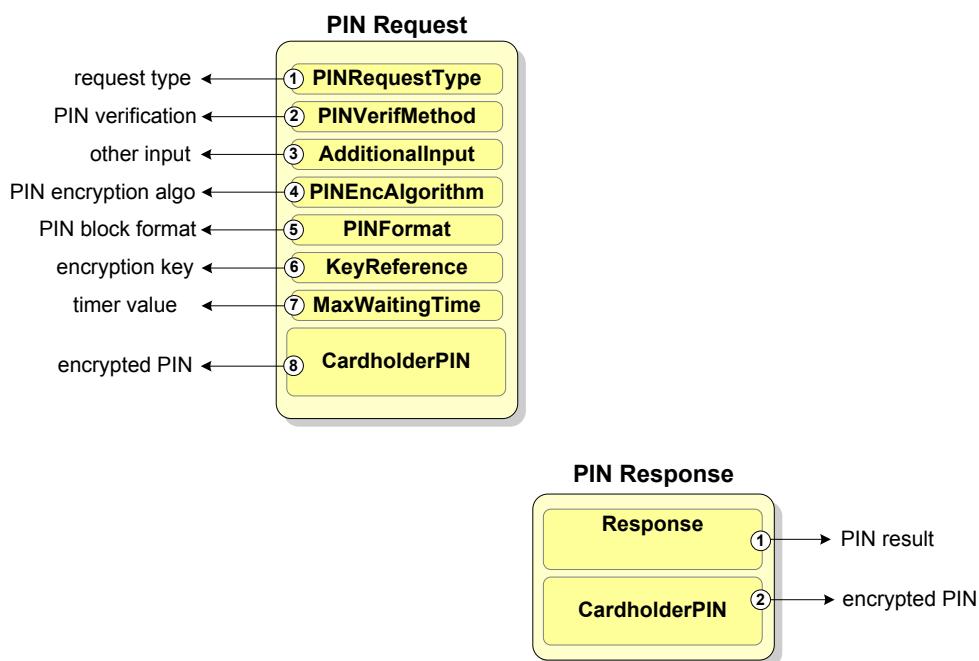


Figure 220: PIN Request/Response Information

The PIN response message body *PINResponse*, contains the following information:

1. The outcome of the PIN request: *Response*.
2. The encrypted PIN: *CardholderPIN*.

4.5.7.3 PIN Request Layout

<i>PINRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		“Device”
MessageCategory	[1..1]		PIN
MessageType	[1..1]		Request
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
PINRequest	[1..1]		
PINRequestType	[1..1]		
PINVerifMethod	[0..1]		Optional if PINRequestType is PINVerify or PINVerifyOnly, absent otherwise
AdditionalInput	[0..1]		Optional if PINRequestType is PINEnter or PINVerify, absent otherwise
PINEncAlgorithm	[0..1]		Optional if PINRequestType is PINEnter, absent otherwise
PINFormat	[0..1]		Optional if PINRequestType is PINEnter, absent otherwise
KeyReference	[0..1]		Optional if PINRequestType is PINEnter, absent otherwise
MaxWaitingTime	[0..1]		Absent if PINRequestType is PINVerify or PINVerifyOnly, optional if PINRequestType is PINEnter
BeepKeyFlag	[0..1]		default False
CardholderPIN	[0..1]		Optional if PINRequestType is PINVerifyOnly, absent otherwise <i>Conformed to EPAS Acquirer protocol.</i>
EncrPINBlock	[1..1]		
PINFormat	[1..1]		
AdditionalInput	[0..1]		If PIN verification requires additional information to verify the PIN

4.5.7.4 PIN Response Layout

<i>PINResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		PIN
MessageType	[1..1]		Response
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
PINResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>
CardholderPIN	[0..1]		If PINRequestType is PINEnter
EncrPINBlock	[1..1]		
PINFormat	[1..1]		
AdditionalInput	[0..1]		If PINEncAlgorithm requires additional information to verify the PIN

4.5.7.5 Error Cases

When the PIN request is successfully processed, the PIN response message, if present, gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The PIN request is received during a Service dialogue or another Device dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

UnavailableService

The format of the format requested in the Print request is not available in the System which manages the printer (see section 4.6.4.3.5 *Unavailable Display Format*).

The PIN verification method requested in the PIN request is not available in the POI (see section 4.6.4.3.8 *Unavailable PIN Verification Method*).

DeviceOut

The display/input device is temporary or permanently out of service (see section 4.6.3.1.3 *Device Temporary Out* and section 4.6.3.1.4 *Device Permanently Out*).

Busy

The POI or Sale Terminal cannot process the PIN request, because another request is already processed on the devices dedicated to the PIN processing (see section 4.6.5.1.3 *Device Busy*).

Wrong PIN

The Cardholder has entered its PIN on the PED keyboard and the verification fails (see section 4.6.7.2 *WrongPIN Error*).

Not Found

The key identified in the component *KeyDataInfo* or *KeyReference* of the message request is not found (see section 4.6.4.4.4 *Key Reference Not Found*).

Cancel

The user has aborted the PIN process on the Customer interface (i.e. the PED keyboard), because he do not want to continue the transaction (e.g., problem of PIN remembering, chooses another payment mean, stop the purchase on a vending machine), or a timeout occurs waiting or during the PIN entering see section 4.6.6.2.1 *User Cancellation*.

4.5.7.6 Examples

This is an example of PIN acquisition and verification.

MessageHeader*(message example 56)*

MessageClass	Device
MessageCategory	PIN
MessageType	Request
DeviceID	375
SaleID	SaleTermA
POIID	POITerm1

PINRequest

PINRequestType	PINVerify
PINVerifMethod	Local1
AdditionalInput	05543

MessageHeader*(message example 57)*

MessageClass	Device
MessageCategory	PIN
MessageType	Response
DeviceID	375
SaleID	SaleTermA
POIID	POITerm1

PINResponse**Response**

Result	Success
--------	---------

4.5.8 Card Reader

4.5.8.1 Processing Overview

The *Card Reader* Device services are provided by a POI Terminal to a Sale Terminal to read a magnetic stripe card or a smart card.

These services are distributed in three messages pairs:

1. *Card Reader Init*: to read a magnetic stripe card or initialise smart card reading with the *CardInit* message pair,
2. *Card Reader APDU*: to exchange APDU with a chip card with the *CardReaderAPDU* message pair,
3. *Card Reader Power-off*: to terminate the smart card dialogue with the *CardReaderPowerOff* message pair.

The card reader device and the Card Reader services are always located on the POI Terminal, and the Card Reader request is always made by the Sale Terminal outside a transaction.

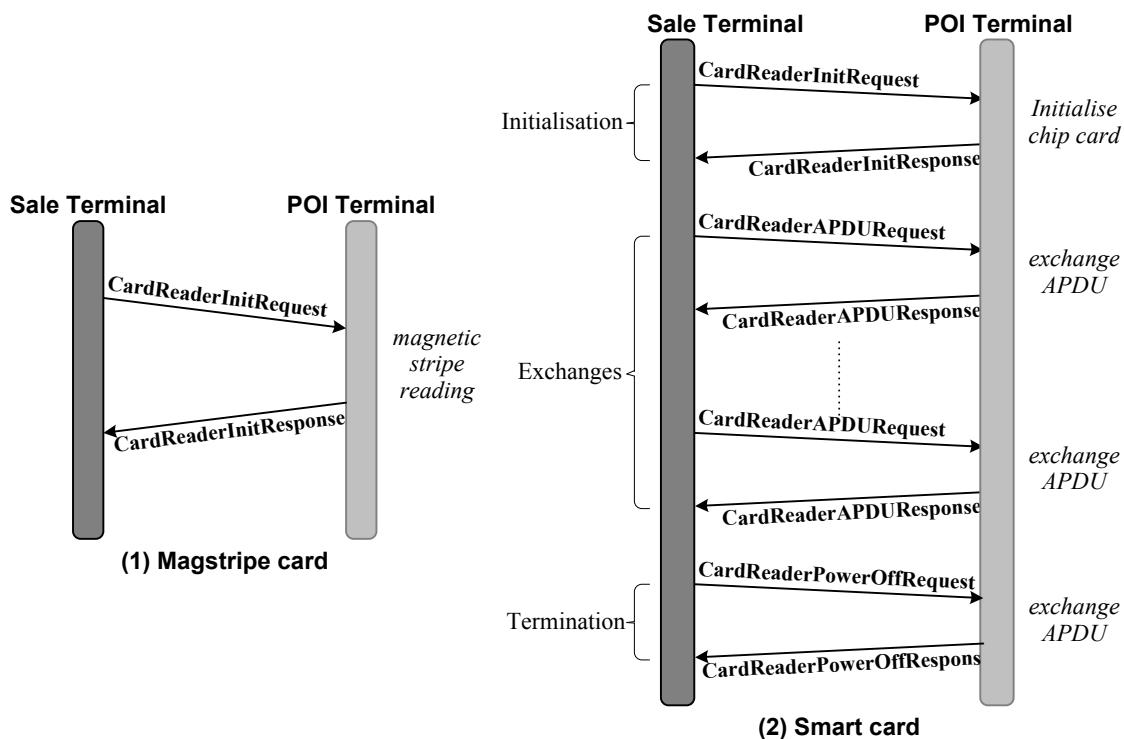


Figure 221: Card Reader Exchanges Overview

Card Reader Init

The *CardInit* message pair enables the entering of a card in a card reader, and initialises a card when inserted in the card reader. When the card is inserted by a Cardholder or a Cashier,

- For magnetic stripe card, the card tracks are read, the Cardholder or the Cashier may be asked to remove the card, and the POI passes tracks to the Sale System. The card reader becomes available for a new device request or a service request from the Sale System, except if the card is also a smart card and the *CardInit* request message ask to keep the card to work with the smartcard after the magstripe reading.
- For smart card²⁵, the chip is initialised (powered on and reset or warm-reset). Initialisation information is forwarded to the Sale System. The card reader becomes busy, and the Sale System can send:
 - A new CardReaderInit command for a warm-reset,
 - One or several CardReaderAPDU commands finished by a CardReaderPowerOff command which frees the card.

Card Reader APDU

The *CardReaderAPDU* message pair performs with the chip of an initialised smart card, an exchange of APDU-Q and APDU-R, conforming to ISO 7816.

Several APDU commands can be grouped in a single CardReaderAPDU request message. The commands shall be processed by the POI terminal in order of appearance and the CardReaderAPDU response shall contain the corresponding command responses in the same order.

Card Reader Power-off

This *CardReaderPowerOff* message pair processes a power off to the chip of an initialised smart card, ejects the card if necessary (motorised card reader), and request to the Customer to remove the card.

²⁵ For a contactless card, the card must remain in the field for requesting APDU dialogues. So “tap and hold” is required.

4.5.8.2 Presentation of the Card Reader Messages

The Card Reader Init request message body *CardReaderInitRequest*, contains the following components:

1. The indicator requiring making a warm reset on an already initialised chip card: *WarmResetFlag*.
2. Restriction on the card reader to enable: *ForceEntryMode*.
3. Request to leave the card in the reader: *LeaveCardFlag*.
4. The maximum time to enter a card: *MaxWaitingTime*.
5. An invitation message to the Customer or the Cashier for entering the card: *DisplayOutput*.

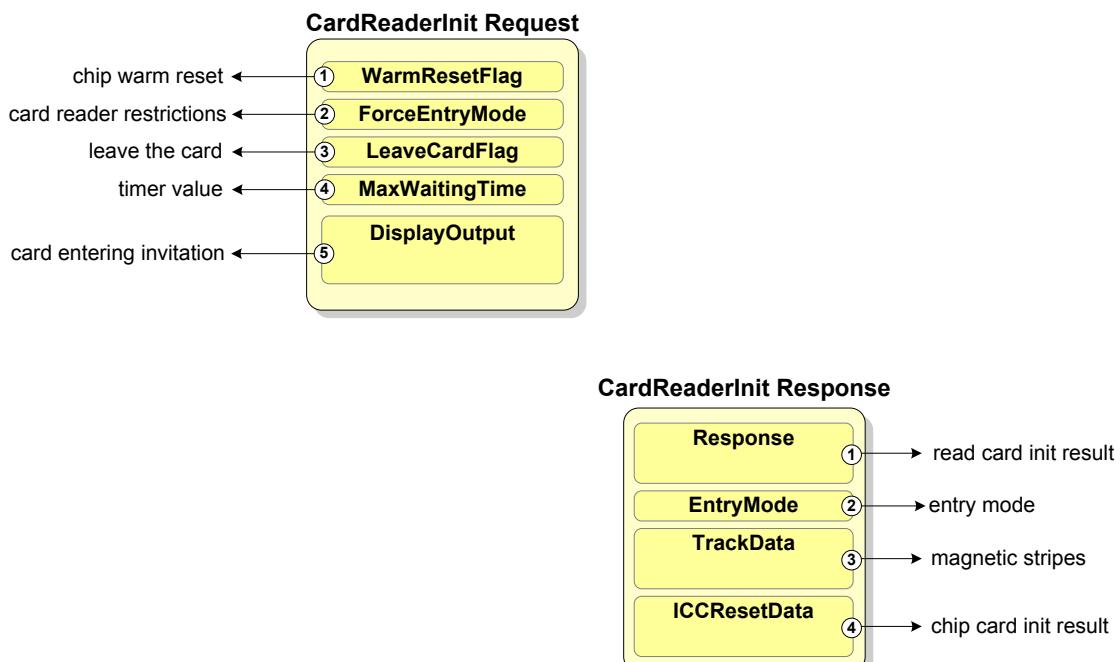


Figure 222: Card Reader Init Request/Response Information

The Card Reader Init response message body *CardReaderInitResponse*, contains the following information:

1. The result of the Card Reader Init: *Response*.
2. The type of card reader: *EntryMode*.
3. The magnetic stripes for a magstripe card reader: *TrackData*.
4. The result of the reset for a smart card (ATR and status words): *ICCResetData*.

The Card Reader APDU request message body *CardReaderAPDUREquest*, contains the standard data to send an ISO 7816 APDU: *APDUClass*, *APDUIInstruction*, *APDUPar1*, *APDUPar2*, *APDUData* and *APDUExpectedLength*.

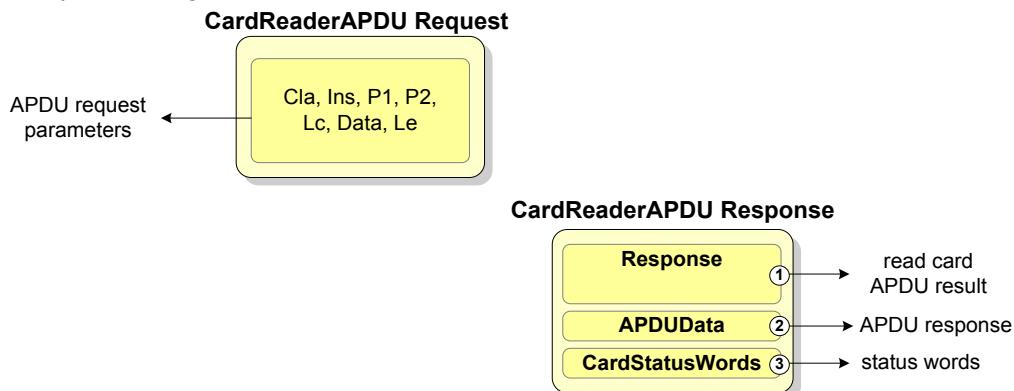


Figure 223: Card Reader APDU Request/Response Information

The Card Reader APDU response message body *CardReaderAPDUDResponse*, contains the following information:

1. The result of the Card Read APDU: *Response*.
2. The APDU response: *APDUData*.
3. The result of the command carried out by the APDU: *CardStatusWords*.

The Card Reader Power Off request message body *CardReaderPowerOffRequest*, contains the following components:

1. The maximum time to remove a card: *MaxWaitingTime*.
2. An invitation message to the Customer or the Cashier for removing the card: *DisplayOutput*.

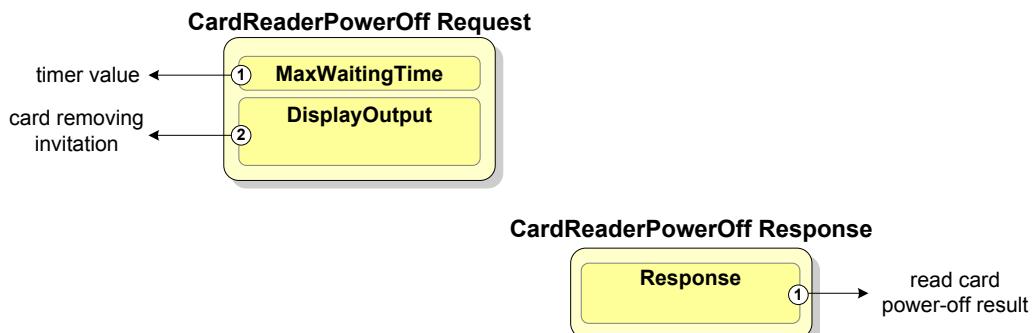


Figure 224: Card Reader Power-Off Request/Response Information

The Card Reader Power Off response message body *CardReaderPowerOffResponse*, contains the following information:

1. The result of the Card Reader Init: *Response*.

4.5.8.3 Card Reader Init Request Layout

<i>CardReaderInitRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		"Device"
MessageCategory	[1..1]		CardReaderInit
MessageType	[1..1]		Request
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
CardReaderInitRequest	[1..1]		
WarmResetFlag	[0..1]		default False
ForceEntryMode	[0..n]		To restrict card modes: "RFID", "MagStripe", "ICC", "SynchronousICC" or "Contactless"
LeaveCardFlag	[0..1]		default True
MaxWaitingTime	[0..1]		
DisplayOutput	[0..1]		Override the prompt message
Device	[1..1]		CashierDisplay or CustomerDisplay
InfoQualify	[1..1]		Display
OutputContent	[1..1]		
OutputFormat	[1..1]		
PredefinedContent	[0..1]		<i>same as Display</i>
ReferenceID	[1..1]		
Language	[0..1]		<i>same as Display</i>
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartRow	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
EndOfLineFlag	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display.

4.5.8.4 Card Reader Init Response Layout

<i>CardReaderInitResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		CardReaderInit
MessageType	[1..1]		Response
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
CardReaderInitResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>
EntryMode	[0..1]		Mandatory if Result is Success Values: RFID, MagStripe, ICC, Contactless or SynchronousICC
TrackData	[0..4]		if EntryMode is RFID or MagStripe
TrackNumb	[0..1]		default 2
TrackFormat	[0..1]		default ISO
TrackValue	[1..1]		
ICCResetData	[0..1]		if EntryMode is ICC, EMVContactless or SynchronousICC
ATRValue	[0..1]		if available
CardStatusWords	[0..1]		if available

4.5.8.5 Card Reader APDU Request Layout

<i>CardReaderAPDUREquest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		“Device”
MessageCategory	[1..1]		CardReaderAPDU
MessageType	[1..1]		Request
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
CardReaderAPDUREquest	[1..n]		
APDUClass	[1..1]		
APDUIstruction	[1..1]		
APDUPar1	[1..1]		
APDUPar2	[1..1]		
APDUData	[0..1]		Mandatory if APDUIstruction requires data. Not allowed otherwise (Lc=0)
APDUExpectedLength	[0..1]		Mandatory if expected data length in response is limited in size. Default value indicates the maximum number of bytes available is required.

4.5.8.6 Card Reader APDU Response Layout

<i>CardReaderAPDUREsponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		CardReaderAPDU
MessageType	[1..1]		Response
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
CardReaderAPDUREsponse	[1..n]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		see <i>PaymentResponse</i>
AdditionalResponse	[0..1]		see <i>PaymentResponse</i>
APDUData	[0..1]		Not present if Result is not OK. Optional depending on the APDU and values of CardStatusWords
CardStatusWords	[1..1]		

4.5.8.7 Card Reader Power-Off Request Layout

<i>CardReaderPowerOffRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		“Device”
MessageCategory	[1..1]		CardReaderPowerOff
MessageType	[1..1]		Request
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
CardReaderPowerOffRequest	[1..1]		
MaxWaitingTime	[0..1]		
DisplayOutput	[0..1]		To override the prompt message
Device	[1..1]		CashierDisplay or CustomerDisplay
InfoQualify	[1..1]		Display
OutputContent	[1..1]		
OutputFormat	[1..1]		
PredefinedContent	[0..1]		<i>same as Display</i>
ReferenceID	[1..1]		
Language	[0..1]		<i>same as Display</i>
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartRow	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
EndOfLineFlag	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display.

4.5.8.8 Card Reader Power-Off Response Layout

<i>CardReaderPowerOffResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		CardReaderPowerOff
MessageType	[1..1]		Response
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
CardReaderPowerOffResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>

4.5.8.9 Card Reader Processing

The figure below presents on the Sale to POI interface, the global process flow of the Card Reader exchanges on the Sale and POI Terminals.

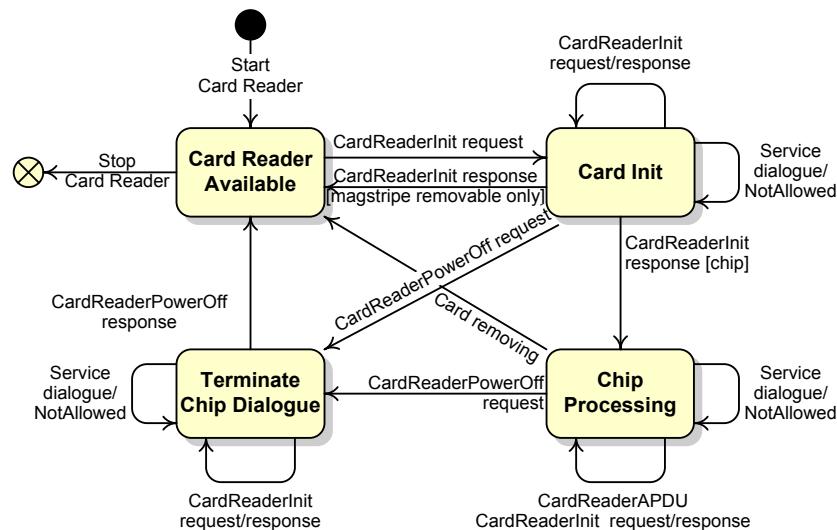


Figure 225: Card Reader Device State Diagram

The Card Reader component(s) of the POI Terminal has to be started-up before allowing the performing of functions on these Card Reader(s).

A Card Reader function cannot be activated by the Sale System when the POI Terminal performs a Service dialogue, or when the Card Reader is busy, waiting for the completion of operations on the smart card.

When the Card Reader is available, the Sale System can activate the insertion and initialisation of a card (CardInit message pair). If this is a magnetic stripe card, requested tracks are read, and the Card Reader become free at the end of the processing.

If the inserted and initialised card is a smart card, the Card Reader becomes busy (Chip Processing state in the diagram). The Sale to POI interface Card Reader remains in this state while CardReaderAPDU requests are performed. The Card Reader becomes free after a chip power off or card removing.

- Rule 1: If the Customer removes the card when the card reader is in the “Chip Processing” state, the POI answer to the next CardReaderAPDU or CardReaderPowerOff a response with *Result*=“Failure” and *ErrorCondition*=“NotFound” (see section 4.6.4.4.7 *Card Removed*).
- Rule 2: If the POI System receives a CardReaderAPDU or a CardReaderPowerOff request in a wrong state, it shall send back a response with *Result*=“Failure” and *ErrorCondition*=“NotAllowed” (see section 4.6.4.1.7 *Forbidden CardReader Sequence*).
- Rule 3: If the POI System receives a CardReaderAPDU request which is forbidden for the card by the POI (e.g. PIN Verify for a general purpose payment card not allowed and filtered by the POI), it shall send back a response with *Result*=“Failure” and *ErrorCondition*=“NotAllowed” (see section 4.6.4.1.8 *Forbidden CardReader APDU Request*).

- Rule 4: If ForceEntryMode is not present, all entry modes supported by POI terminal are activated. For a magstripe card, the CardReaderInit response contains the tracks and the *EntryMode* "MagStripe".
- Rule 5: When the card is removed by the user in the ChipProcessing state, the POI shall send an unsolicited event to the Sale Terminal, EventNotification message with *EventToNotify* = "CardRemoved".
- Rule 6: If the component *MaxWaitingTime* is present in the CardReaderInit or CardReaderPowerOff request message, after this period of time, the card reader command must automatically be cancelled, and the CardReaderInit or CardReaderPowerOff response with *Result*="Failure" and *ErrorCondition*="Cancel" (see section 4.6.6.2.2 *System Cancellation*).

4.5.8.10 Error Cases

When the Card Reader request is successfully processed, the Card Reader response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The Card Reader request is received during a Service dialogue or another Device dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

The POI System receives a Card Reader message request from the Sale System, with cannot be accepted in this state (see section 4.6.4.1.7 *Forbidden CardReader Sequence*).

The POI System receives a Card Reader APDU message request from the Sale System, with cannot be accepted for the card (see section 4.6.4.1.8 *Forbidden CardReader APDU Request*).

UnavailableDevice

The card reader is not present in the System or the Terminal, the Device request cannot be handled (see section 4.6.3.2 *UnavailableDevice Error*).

UnavailableService

The Card Reader service is not available in the POI Terminal (see section 4.6.4.3.4 *Unavailable Device Service*).

The format of the display requested in the Card Reader request is not available in the System which manages the display logical device (see section 4.6.4.3.5 *Unavailable Display Format*).

DeviceOut

The card reader is temporary or permanently out of service (see section 4.6.3.1.3 *Device Temporary Out* and section 4.6.3.1.4 *Device Permanently Out*).

Busy

The POI or Sale Terminal cannot process the Input request, because another request is already processed on this card reader device (see section 4.6.5.1.3 *Device Busy*).

NotFound

The information to be displayed in a Device message request has the format “PredefinedContent”, and the identification of the message *ReferenceID* is not found by the receiver of the message (see section 4.6.4.4.2 *Message Not Found*).

The smart card has been removed since the answer to the last message (see section 4.6.4.4.7 *Card Removed*).

Cancel

The user has aborted the Card Reader command, or a timeout occurs waiting for the entering or removing of the card (see section 4.6.6.2.1 *User Cancellation*).

4.5.8.11 Examples

a) Magstripe Card Reading

The Sale Terminal *SaleTermA* requests to the POI Terminal *POITerm1* a CardReader Init Request with a display message for the Customer to enter his card.

MessageHeader		(message example 58)
MessageClass	Device	
MessageCategory	CardReaderInit	
MessageType	Request	
DeviceID	385	
SaleID	SaleTermA	
POIID	POITerm1	
CardReaderInitRequest		
DisplayOutput		
Device	CustomerDisplay	
InfoQualify	Display	
OutputContent		
OutputFormat	Text	
OutputText		
Text	Please Enter Your Card	

When a magnetic stripe card has been inserted by the Customer, the CardReader Init Response is sent by the POI Terminal *POITerm1* to the Sale Terminal *SaleTermA*.

b) ICC Card Reading

The Sale Terminal *SaleTermA* requests to the POI Terminal *POITerm1* a CardReader Init Request with a display message for the Customer to enter his card.

MessageHeader		(message example 60)
MessageClass	Device	
MessageCategory	CardReaderInit	
MessageType	Request	
DeviceID	421	
SaleID	SaleTermA	
POIID	POITerm1	
CardReaderInitRequest		
DisplayOutput		
Device	CustomerDisplay	
InfoQualify	Display	
OutputContent		
OutputFormat	Text	
OutputText		
Text	Plese Enter Your Card	

When an ICC card has been inserted by the Customer, the CardReader Init Response is sent by the POI Terminal *POITerm1* to the Sale Terminal *SaleTermA*.

c) Card Reader APDU

After initialization of the ICC card, the Sale Terminal requests a CardReader APDU Request with the ISO 7816 command: SELECT

MessageHeader		(message example 62)
MessageClass	Device	
MessageCategory	CardReaderAPDU	
MessageType	Request	
DeviceID	422	
SaleID	SaleTermA	
POIID	POITerm1	
CardReaderAPDURequest		
APDUClass	00	
APDUIInstruction	A4	
APDUPar1	04	
APDUPar2	00	
APDUData	A0000000421010	
APDUExpectedLength	07	

The SELECT APDU request is sent to the ICC by the POI Terminal, processed by the ICC card, the response is sent to the POI Terminal, which sends CardReader APDU Response to the Sale Terminal.

MessageHeader		(message example 63)
MessageClass	Device	
MessageCategory	CardReaderAPDU	
MessageType	Response	
DeviceID	422	
SaleID	SaleTermA	
POIID	POITerm1	
CardReaderAPDUResponse		
Response		
Result	Success	
APDUData	6F2D8407A0000000421010A522500243428701019F1207544553542043429F110101BF0C05DF60020B145F2D026672	
CardStatusWords	9000	

d) Card Reader Power-Off

Then the Sale Terminal requests to the POI Terminal a CardReader Power-Off Request with a display message for the Customer to remove the card.

MessageHeader		(message example 64)
MessageClass	Device	
MessageCategory	CardReaderPowerOff	
MessageType	Request	
DeviceID	423	
SaleID	SaleTermA	
POIID	POITerm1	
CardReaderPowerOffRequest		
DisplayOutput		
Device	CustomerDisplay	
InfoQualify	Display	
OutputContent		
OutputFormat	Text	
OutputText		
Text	Please Remove Your Card	

When the ICC has been powered off by the POI Terminal and removed by the Customer, the CardReader Power-Off Response is sent by the POI Terminal to the Sale Terminal.

MessageHeader		(message example 65)
MessageClass	Device	
MessageCategory	CardReaderPowerOff	
MessageType	Response	
DeviceID	423	
SaleID	SaleTermA	
POIID	POITerm1	
CardReaderPowerOffResponse		
Response		
Result	Success	

4.5.9 Transmit

4.5.9.1 Presentation of the Transmit Messages

The Transmit request message body *TransmitRequest*, contains:

1. A flag *WaitResponseFlag* to indicate to wait a response to the transmitted message.
2. The maximum time for the transmission: *MaximumTransmitTime*.
3. The address where to send the message: *DestinationAddress*.
4. The message to send *Message*:

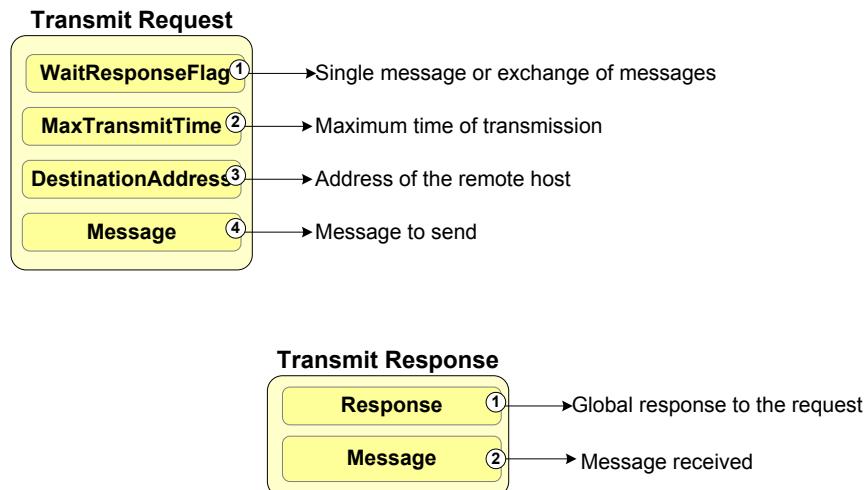


Figure 226: Transmit Information

The Transmit response message body *TransmitResponse* contains:

1. The result of the transaction: *Response*.
2. The received message if any: *Message*.

4.5.9.2 Transmit Request Layout

<i>Transmit Request Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		"Device"
MessageCategory	[1..1]		Transmit
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
TransmitRequest	[1..1]		
WaitResponseFlag	[0..1]		default True
MaximumTransmitTime	[1..1]		
DestinationAddress	[1..1]		
Message	[1..1]		

4.5.9.3 Transmit Response Layout

<i>Transmit Response Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Transmit
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
TransmitResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>see PaymentResponse</i>
AdditionalResponse	[0..1]		<i>see PaymentResponse</i>
Message	[0..1]		

4.5.9.4 Transmit Processing

The *Transmit* message pair is used to request the sending of a message, and the receipt of a response, using the other party as a gateway.

A typical usage is in an MPos environment, where a mobile phone, the Sale terminal, has an extension, the POI terminal that is used to perform the payment transaction:

- 1) The mobile phone sends a Payment request message to the extension to perform a card payment.
- 2) The extension starts the card payment transaction.
- 3) The transaction requests an online authorisation. As the extension has no communication device, it sends the authorisation in a transmit request device message to the mobile phone.
- 4) The Transmit response message conveys the authorisation response.
- 5) The extension completes the card payment transaction and sends the Payment response to the mobile phone.

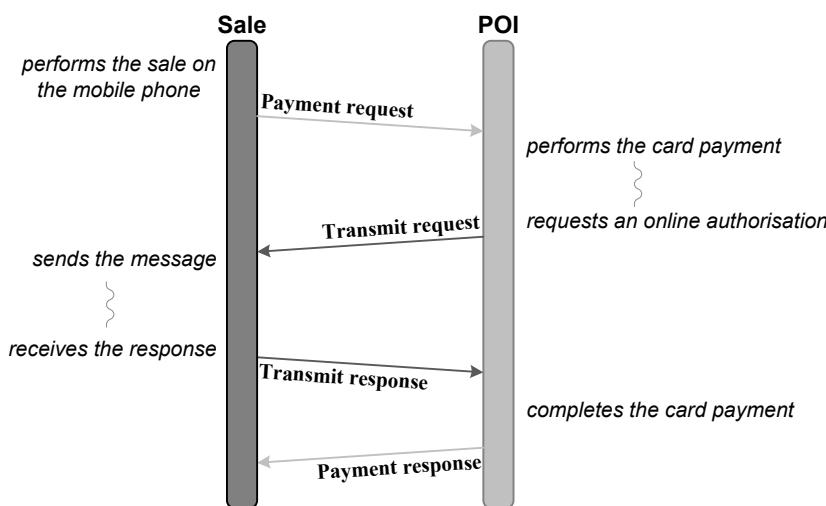


Figure 227: Transmit Exchange in a MPos Architecture

- Rule 1: The Transmit request message uses an implicit communication infrastructure provided by the receiver of the message, and do not allow the choice of several communication infrastructures.
With this limitation, the destination *DestinationAddress* which contains either an IP address with an optional port number, or a phone number is not completed by a type of destination address which is implicit.
The sender of the *TransmitRequest* do not need to know the type of communication used to send the *Message*.
- Rule 2: When the destination *DestinationAddress* cannot be reached or the message *Message* cannot be sent, the receiver must send back a response with *Result*="Failure" and *ErrorCondition*="NotFound" (see section 4.6.8.1.1 *Host Unreachable*).
- Rule 3: When a response must be received (i.e. *WaitResponseFlag*="True"), and no response was received after *MaximumTransmitTime* seconds, the receiver must send back a response with *Result*="Failure" and *ErrorCondition*="NoHostAnswer" (see section 4.6.8.1.2 *No Host Answer*).

4.5.9.5 Error Cases

When the Transmit request is successfully processed, the Transmit response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The Transmit request is received during a Service dialogue or another Device dialogue (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.1 *Forbidden Dialogue*).

UnavailableDevice

The communication device is not present in the System or the Terminal, the Device request cannot be handled (see section 4.6.3.2 *UnavailableDevice Error*).

UnavailableService

The Transmit service is not available in the POI Terminal (see section 4.6.4.3.4 *Unavailable Device Service*).

The format of the display requested in the Transmit request is not available in the System which manages the display logical device (see section 4.6.4.3.5 *Unavailable Display Format*).

DeviceOut

The communication device is temporary or permanently out of service (see section 4.6.3.1.3 *Device Temporary Out* and section 4.6.3.1.4 *Device Permanently Out*).

Cancel

The system has aborted the Transmit command because a problem of access to the address, or a timeout occurs during the transmission (see section 4.6.6.2.2 *System Cancellation*).

HostUnreachable

The destination for the requested service cannot be reached, so it is considered as temporary unavailable (see section 4.6.8.1.1 *Host Unreachable*).

NoHostAnswer

The connected host has not answered to an online request (see section 4.6.8.1.2 *No Host Answer*).

4.5.9.6 Example

The payment application *POITerm1* requests to the mobile *SaleTermA* to send an authorisation request message.

The message to send is:

```
<AuthReq Amount="1000" PAN="916465"></AuthReq>
```

Providing the following message in hexadecimal:

```
3C4175746852657120416D6F756E743D2231303030222050414E3D22393136343635223E3C2F4  
17574685265713E
```

MessageHeader		(message example 66)
MessageClass	Device	
MessageCategory	Transmit	
MessageType	Request	
ServiceID	675	
DeviceID	1	
SaleID	SaleTermA	
POIID	POITerm1	
TransmitRequest		
MaximumTransmitTime	30	
DestinationAddress	acquirer.com:8080	
Message	3C4175746852657120416D6F756E743D2231303030222050414E3D22393136343635223E3 C2F417574685265713E	

The authorisation response message is sent to the payment application *POITerm1* by the mobile *SaleTermA*.

MessageHeader		(message example 67)
MessageClass	Device	
MessageCategory	Transmit	
MessageType	Response	
ServiceID	675	
DeviceID	1	
SaleID	SaleTermA	
POIID	POITerm1	
TransmitResponse		
Response		
Result	Success	
Message	3C4175746852657020416D6F756E743D22313030302220526573706F6E73653D224170707 26F766564222041757468436F64653D22373934352020223E3C2F417574685265703E	

The receive message in hexadecimal is:

```
3C4175746852657020416D6F756E743D22313030302220526573706F6E73653D22417070726F7  
66564222041757468436F64653D22373934352020223E3C2F417574685265703E
```

Providing the following message response:

```
<AuthRep Amount="1000" Response="Approved" AuthCode="7945" "></AuthRep>
```

4.6 Error Reporting and Error Cases

This chapter presents all the error cases of the Retailer protocol by category, and the standard framework to report these errors.

The section 1 describes the data elements of the message responses which contain the result of the processing made by the receiver of the request, and particularly the error reporting.

Following sections define each error case and the information to report in the message response.

4.6.1 Outcome of Message Processing

The message body holds a different content for every message type. However, the first element of the body of a message response is the data structure *Response* including the result of the processing requested in the message request:

- *Result*, global result of the processing of the message request (success, partial or failure).
- *ErrorCondition*, condition that has produced a failure, allowing resolution of the failure by the requestor. This component could also be present to declare a warning if the error is not fatal for the processing of the request, in order to react during the testing phase or correct the problem in the field.
- *AdditionalResponse*, additional information related to the result of a message request processing.

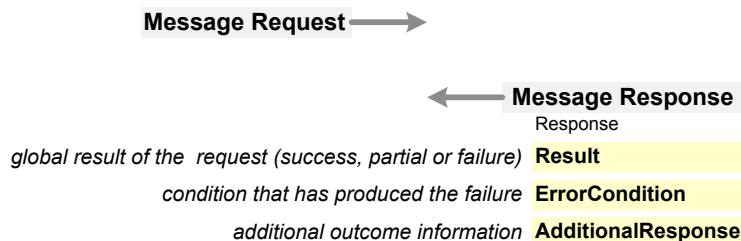


Figure 228: Outcome of Message Processing

The presence condition of these data elements is presented in the table below.

Data Element	Presence Condition
Result	Mandatory for all response messages.
ErrorCondition	Mandatory, if Result is "Failure".
AdditionalResponse	Mandatory, if Result is "Failure".

Table 18: Response Presence Condition

The messages *Display Response*, *Input Response*, and *Print Response* messages do not contain the *Response* data structure directly as the first element of the message body. As the request could contain several display, input or print requests, the result of these multiple requests (in the *Response* data structure), is included in a global structure which could be repeated (the data structures *OutputResult* and *InputResult*).

Rule 1: If *Result* has the value "Failure" in a message response, the receiver of the response (Sale or POI) has to log the error with the content of the data elements *ErrorCondition* and *AdditionalResponse*.

Following sections define, for each value of the *ErrorCondition* data element, the most standard cases for errors, and provide associated format framework for the *AdditionalResponse* data element.

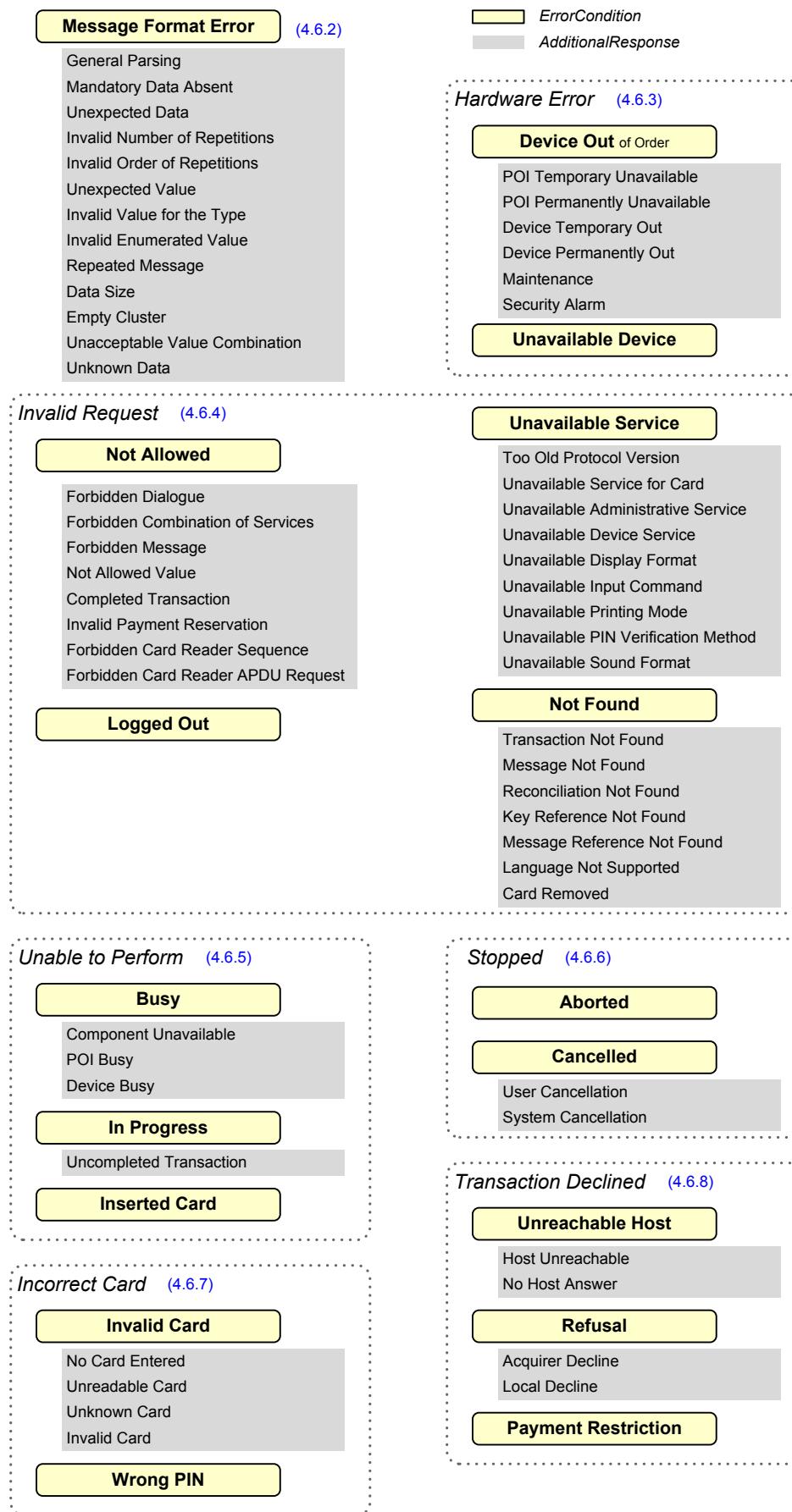


Figure 229: ErrorCondition - AdditionalResponse

4.6.2 Message Format Errors

4.6.2.1 Message Format

4.6.2.1.1 General Parsing Error

The parsing of the message has failed:

General Parsing Error: <Absolute Data Name> <Error>

Absolute Data Name: absolute data name if any (e.g. LoginRequest.SaleSoftware, MessageHeader.MessageClass)

Error: parsing error if any (e.g. unexpected SaleSoftware tag, MessageHeader, LoginRequest or SecurityTrailer expected)

4.6.2.1.2 Mandatory Data Absent

A data structure or a data element which is mandatory in the definition of the message or its processing is absent:

Mandatory Data Element or Structure Absent: <Absolute Data Name>

Absolute Data Name: absolute data name (e.g. LoginRequest.SaleSoftware, MessageHeader.MessageClass)

4.6.2.1.3 Unexpected Data

A data structure or a data element which need to be absent in the definition of the message is present in the message:

Unexpected Data Element or Structure: <Absolute Data Name> - <Reason>

Absolute Data Element Name: absolute data name (e.g. MessageHeader.DeviceID)

Reason: reason for which the system does not attend this data (e.g. not a Sale Terminal login)

4.6.2.1.4 Invalid Number of Repetitions

The maximum number of repetitions of a data structure or a data element which could be repeated is exceeded. The message could not be processed:

Too many Repetitions of the Component: <Absolute Data Name> - <Reason>

Absolute Data Element Name: absolute data name (e.g. DisplayRequest.OutputContent)

Reason: reason or maximum number of repetitions (e.g. two CashierDisplay-Error requests)

4.6.2.1.5 Invalid Order of Repetitions

The order of repetitions of a data structure or a data element is invalid (e.g. the repetitions must be ordered by the value of one of the components):

Invalid Repetitions Order of the Component: <Absolute Data Name> - <Reason>

Absolute Data Element Name: absolute data

Reason: reason

4.6.2.1.6 Unexpected Value

A data element is sent in the message with a value which is not accepted by the receiver because the rule associated to the component specifies another value:

Unexpected Data Element Value: <Absolute Data Name> - <Expected Value(s) or Reason>

Absolute Data Element Name: absolute data name (e.g. MessageHeader.MessageType)

Expected Value: value required by the rule associated to this data element if any (e.g. PaymentRequest, because the body of the message do not correspond to the MessageType in the header, or the value of the data element in the response message has to be copied from the request message). It could be also the reason why the value is unexpected (e.g. the StringMask specifies alphabetic characters, and the input command request a digit string).

4.6.2.1.7 Invalid Value for the Type

A data element has a value which does not respect the syntax of its defined type (e.g. an Integer data element value contains non numeric characters in XML/Schema, or the data element is a date written with two digits for the year instead of four):

<Absolute Data Name> Invalid Value <Value> for the Type <Type> [and Format <Format>]

Absolute Data Name: absolute data name (e.g. MessageHeader.MessageType)

Value: value sent in the message.

Type: type of the data element.

Format: if this is a problem of format, the expected format.

4.6.2.1.8 Invalid Enumerated Value

A data element has a value which is not one of the enumerations defined for the type Enumerated or Cluster:

<Absolute Data Name> Value is not part of the Enumerated or Cluster Type <Type>

Absolute Data Name: absolute data name (e.g. MessageHeader.MessageType)

Type: enumerated or cluster type of the data element.

4.6.2.1.9 Repeated Message

In the header of a message request, the value of the ServiceID or DeviceID date element is the same than a previous message request sent recently:

Repeated Message: <ID Name> - <Value>

ID Name: ServiceID or DeviceID

Value: repeated value of the ServiceID or DeviceID

4.6.2.1.10 Data Size

A data element exceeds the defined size limit or a message is too big for the application:

Data Size: <Absolute Data Name> - <Limit>

Absolute Data Element Name: absolute data name (e.g. LoginRequest.POISystemData)

Limit: maximum size of the data element that could accept the implementation.

4.6.2.1.11 Empty Cluster

A data element which has the type Cluster does not contain any value (empty XML/Schema list, or null ASN.1 bit string):

Empty Cluster: <Absolute Data Name>

Absolute Data Element Name: absolute data name (e.g. LoginRequest.POISystemData)

4.6.2.1.12 Unacceptable Value Combination

The combination of values for a set of data element is unacceptable:

Unacceptable Value Combination: [<Absolute Data Name> : <Value>]*

Absolute Data Element Name: absolute data name

Value: Value of the Data Element

4.6.2.1.13 Unknown Data

A data structure or a data element the receiver of the message does not recognize. This is generally the case when the sender does not uses the same version of the protocol. This data has to be ignored, and the response may contain this information as a warning, useful in the testing phase:

Unknown Data Element or Structure: <Absolute Data Name>

Absolute Data Element Name: absolute data name (e.g. MessageHeader.DeviceID)

4.6.3 Hardware Error

4.6.3.1 DeviceOut Error

4.6.3.1.1 POI Temporary Unavailable

The POI element cannot start to work, because of a temporary error.

POI is Temporary Unavailable: <Reason>

Reason: reason for which the device is not available (e.g. maintenance)

4.6.3.1.2 POI Permanently Unavailable

The POI element cannot start to work, because of a permanent error on a device (e.g. card reader out of order).

<Device> **POI is Permanently Unavailable:** <Reason>

Device: Device of the POI Terminal (e.g. Card Reader, Printer...)

Reason: reason for which the device is not available (e.g. out of order, absent, unconnected)

4.6.3.1.3 Device Temporary Out

The device request cannot be handled, because of a temporary error on a device (e.g. printer out of paper).

<Device> **is Temporary Out of Service:** <Reason>

Device: Device of the Terminal (e.g. Card Reader, Printer...)

Reason: reason for which the device is not available (e.g. no more paper)

4.6.3.1.4 Device Permanently Out

The device request cannot be handled, because of a permanent error on a device (e.g. problem of connection with the display).

<Device> **is Permanently Out of Service:** <Reason>

Device: Device of the Terminal (e.g. Card Reader, Printer...)

Reason: reason for which the device is not available (e.g. out of order, absent, unconnected)

4.6.3.1.5 Maintenance

A POI Terminal or a POI device is entering or has finished maintenance operation.

<Device> **Maintenance:** <Type>

Device: Device of the Terminal (e.g. Card Reader, Printer...)

Type: Type of maintenance (e.g. parameter update), and expected time if available.

4.6.3.1.6 Security Alarm

A security alarm is activated on a POI Terminal or the POI System.

Security Alarm: <Alarm>

Device: Device of the Terminal (e.g. Card Reader, Printer...)

Alarm: Type of alarm (e.g. cryptographic key unknown, invalid cryptographic key, internal alarm sensor activated).

4.6.3.2 ***UnavailableDevice Error***

The logical device is not present in the System or the Terminal, the Device request cannot be handled.

<DeviceType> <InfoQualify> Device Unavailable Absent or not Configured

DeviceType: A value of the Device data element

InfoQualify: A value of the InfoQualify data element

4.6.4 Invalid Request

4.6.4.1 NotAllowed Error

4.6.4.1.1 Forbidden Dialogue

The POI System receives a message request from the Sale System, which could not be accepted and processed because a Service or a Device dialogue is in progress.

Forbidden Request, Service Dialogue <Service> is in Progress

Service: Type of service in progress, Service or Device

4.6.4.1.2 Forbidden Combination of Service

The POI System receives a message request from the Sale System, which could not be accepted and processed because it contains an incompatible combination of services.

Forbidden Combination of Service: <Services>

Services: requested services (e.g. split payment and reservation)

4.6.4.1.3 Forbidden Message

A message which is received by the POI System or the Sale System could not be accepted and processed.

Forbidden Message <Message>

Message: Type of message (e.g. Abort, Event...)

4.6.4.1.4 NotAllowed Value

The POI or the Sale System receives a message request, with the value of a component which could not be accepted for some reason.

<ComponentName>: NotAllowed Value: <Value>, Reason: <Reason>

ComponentName: name of the component, qualified name if the component name alone is ambiguous (e.g. RequestedAmount)

Value: value of the component (e.g. 800.00)

Reason: reason to deny this value (e.g. OneTimeReservation maximum amount of 700.00)

4.6.4.1.5 Completed Transaction

An Enable Service message is received by the POI System to abort an already finished transaction.

Completed Transaction

4.6.4.1.6 Invalid Payment Reservation

The POI System receives a Payment request with the *PaymentType* set to “Completion” for a reservation transaction, and the period is closed or the amount is too high.

Invalid Payment Reservation: <Reason>

Reason: the reservation period is closed, or the amount is too high.

4.6.4.1.7 Forbidden CardReader Sequence

The POI System receives a Card Reader message request from the Sale System, which cannot be accepted in this state:

<Message>: NotAllowed in the State: <State>

Message: message name

State: Card Reader state (Card Reader Available)

4.6.4.1.8 Forbidden CardReader APDU Request

The POI System receives a Card Reader APDU message request from the Sale System, which cannot be accepted for the card (e.g. "Verify PIN" APDU for a general purpose card):

<APDU> APDU Request NotAllowed for the Card: <CardBrand>

APDU: command name (e.g. PIN Verify)

CardBrand: Card brand in the card reader

4.6.4.2 LoggedOut Error

The Sale Terminal (resp. Server) has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal (resp. Server):

<SaleID> Never Login Since Last <Event> at <Time>

SaleID: Identifier of the Sale Terminal or Server

Event: "Logout" or "Start-up"

Time: date and time of the last Logout or Start-up

4.6.4.3 Unavailable Service Error

4.6.4.3.1 Too Old Protocol Version

The *ProtocolVersion* sent by the Sale System is too old to be manageable by the version of protocol implemented in the POI.

Sale Protocol Version <Version> Too Old, Version implemented: <Version>

Version: protocol version implemented (e.g. 1.0)

4.6.4.3.2 Unavailable Service for the Card

The Service requested by the Sale System is not available in the POI for this type of card.

Unavailable Service <Service> for Card: <CardBrand>

Service: service not available: (non exhaustive list)

Cashback, Split Payment

CardBrand: card brand (same value than *PaymentBrand* or *LoyaltyBrand* component of the Payment response message)

4.6.4.3.3 Unavailable Administrative Service

The Administrative Service requested by the Sale System is not available in the POI System.

Unavailable Administrative Service

4.6.4.3.4 Unavailable Device Service

The Device Service requested by the Sale System (resp. POI System) is not available in the POI System (resp. Sale System).

Unavailable Device Service

4.6.4.3.5 Unavailable Display Format

The format of the display requested in the Device request is not available in the System which manages the display logical device.

Unavailable Display Format: <Format> Feature: <Feature>

Format: display format

Message Reference

Text

XHTML

BarCode

Feature: If a certain feature of the format is not supported (e.g. "Font" for Text)

4.6.4.3.6 Unavailable Input Command

The format of the display requested in the Device request is not available in the System which manages the display logical device.

Unavailable Input Command: <Command>

Command: input command (see *InputCommand* data element)

4.6.4.3.7 Unavailable Printing Mode

The type of document to print is not supported by the printer or the driver.

Unavailable Printing Mode: <Parameter> <Value>

Parameter: parameter of the Print request (DocumentQualifier, ResponseMode)

Value: value of the parameter

4.6.4.3.8 Unavailable PIN Verification Method

The PIN verification method requested in the PIN request is not available in the POI.

Unavailable PIN Verification Method: <Method>

Method: PIN verification method

4.6.4.3.9 Unavailable Sound Format

The format of the sound requested in the Device request is not available in the System which manages the sound logical device.

Unavailable Sound Format: <Format>

Format: sound format

Message Reference

Sound Reference

Text

4.6.4.4 **NotFound Error**

4.6.4.4.1 Transaction Not Found

The POI System receives a message request from the Sale System, with a link to a transaction which has been realised with another pair of messages. The transaction is not stored in the POI System, because the transaction cannot be found in the transaction log or the transaction identification is incorrect.

Transaction <TransactionID> is not Found in the <Component> Link

TransactionID: identification of the transaction with time stamp

Component: CardAcquisitionReference or OriginalPOITransaction

4.6.4.4.2 Message Not Found

The POI System receives a Transaction Status request from the Sale System, with a *MessageReference* not stored in the POI System or not in progress in the POI Terminal.

Message not Found, last <MessageCategory> has ID <ID>

MessageCategory: the message type requested

ID: last ServiceID or DeviceID processed for this message type

4.6.4.4.3 Reconciliation Not Found

The Sale System has requested the result of a previous reconciliation period which cannot be found by the POI System.

Reconciliation <POIReconciliationID> is not Found: <Reason>

POIReconciliationID: Identification of the reconciliation.

Reason: reason of the decline (e.g. too old, nonexisting, or uncompleted period)

4.6.4.4.4 Key Reference Not Found

The key identified in the message request is not found.

Key Reference <KeyRef> is not Found

KeyRef: identification of the key

4.6.4.4.5 Predefined Message Reference Not Found

The information to be displayed in a Device message request has the format “PredefinedContent”, and the identification of the message *ReferenceID* is not found by the receiver of the message.

Message Reference <MessageID> is not Found

MessageID: identification of the message

4.6.4.4.6 Language Not Supported

The language requested in the message is not available for the purpose: Cashier language, Customer language, or language of predefined message to display or print.

Language <Language> is not Supported

Language: language requested for the Cashier, the Customer or a predefined message to display or print.

4.6.4.4.7 Card Removed

A smart card has been removed since the answer to the last message.

Card Removed by the Customer <LastMessage>

LastMessage: identification of the last message (e.g. CardAcquisition, CardReaderInit, CardReaderAPDU)

4.6.5 Unable to Perform

4.6.5.1 Busy Error

4.6.5.1.1 Component Unavailable

The POI System, the POI Terminal or a POI component is temporary not available. Reason could be:

- Initialisation in progress,
- Maintenance in progress,
- Device busy (i.e. already used for another task)

POI <Component> Temporary Unavailable: <Reason>

Component: "System", "Terminal" or a specific component

Reason: reason for which the component is not available (e.g. Maintenance in progress, Keyboard busy, Printer buffer full)

4.6.5.1.2 POI Busy

The POI Terminal is temporary not available, because another request is processed by the POI.

POI Terminal Busy to Process another Request: <Request>

Request: message request (e.g. InputRequest, DisplayRequest...)

4.6.5.1.3 Device Busy

The POI or Sale Terminal cannot process a Device Request, because another request is already processed on this device.

<System>Terminal <Device>Busy to Process the Device Request

System: type of System (i.e. POI or Sale)

Device: type of logical device, i.e. of the data element *Device* (e.g. CashierDisplay)

4.6.5.2 InProgress Error

4.6.5.2.1 Uncompleted Transaction

The POI Terminal cannot provide the response of the message in the Transaction Status response message, because the processing is not complete and still in progress.

Uncompleted Transaction

4.6.5.3 InsertedCard Error

The Input Device requested a *NotifyCardInputFlag* and the Customer inserts a card in the card reader without answering the Input command. The POI has aborted the Input command processing, and answers this ErrorCondition value in the Input response message.

There is no AdditionalResponse in this case.

4.6.6 Stopped

4.6.6.1 Aborted Error

The Sale System sent an Abort request message before the end of the Service request processing. The POI Terminal has aborted the transaction, and sends the Service response message with the *ErrorCondition* containing the value "Aborted" and *AdditionalResponse* containing the form below to report the result of the aborted transaction.

Service Aborted during <Status>- Reason: <AbortReason> - from: <SaleID> - MessageID: <ServiceID>

Status: Status of the transaction at the time of the abort (e.g. "Payment Card Selection", "PIN Entering", "Online Authorization")

AbortReason: Reason of the abort provided by the Sale System in the component AbortReason of the Abort request message.

SaleID: Identification of the Sale Terminal or Server that have sent the Abort request message.

Service: Identification of the Abort request message.

4.6.6.2 Cancel Error

4.6.6.2.1 User Cancellation

The user has aborted the transaction on the Customer interface (e.g. by using the key for cancellation on the POI Terminal keyboard), because he does not want to continue the transaction (e.g., problem of PIN remembering, chooses another payment mean or card, stop the purchase on a vending machine). The cancellation by the Customer has to conform to the rules of the chosen card, e.g. it could be forbidden to cancel a transaction after approval by the Issuer with an online authorisation, only a Customer error by removing the card too early may cancel the transaction.

User Cancellation during <Status>

Status: Status of the transaction at the time of the cancellation (e.g. "Payment Card Selection", "PIN Entering", "Online Authorization")

4.6.6.2.2 System Cancellation

The System has forced the cancellation of the command because the user or the system didn't react on time to the application solicitation.

System Cancellation during <Reason>

Reason: reason of the cancellation (e.g. "Timeout")

4.6.7 Incorrect Card

4.6.7.1 InvalidCard Error

4.6.7.1.1 No Card Entered

The POI terminates the transaction because no card is entered by the Customer:

No Card Entered after <Time> Seconds

Time: waiting card time in seconds

4.6.7.1.2 Unreadable Card

The card cannot be read:

Unreadable Card <Reason>

Reason: Error on card reading (e.g. wrong Answer To Reset of the card)

4.6.7.1.3 Unknown Card

The POI terminates the transaction because the inserted card is not configured in the system and cannot be processed:

Unknown Card <BIN, AID...>

BIN or AID: BIN or AID of the card “BIN: val” or “AID: val”

4.6.7.1.4 Invalid Card

The POI terminates the transaction because the inserted card cannot be used for some reason:

Invalid Card < Reason >

Reason: type of invalidity (e.g. card expired, card not allowed for the transaction, suspicion of fraud)

4.6.7.2 WrongPIN Error

The Cardholder has entered his PIN on the PED keyboard and the verification fails.

Wrong PIN <RetryNb> Retries – Remaining <RemainingRetries>

RetryNb: Number of retries made by the Cardholder during the PIN command processing

RemainingRetries: Number of remaining retries for PIN verification (if provided by the chip card or the Host)

4.6.8 Transaction Declined

4.6.8.1 UnreachableHost Error

4.6.8.1.1 Host Unreachable

A host necessary for the requested service cannot be reached, so it is considered as temporary unavailable. The transaction is declined, because it cannot be authorised online and has not been forced by the Cashier.

Host Unreachable: <Reason>

Reason: transport connection error (see 3.2.4 Transport Error Handling):

- Unable to Establish a Transport Connection (ERTR01)
- Unable to Send a Message (ERTR03)
- Max Global Number of Connections (ERTR06)

4.6.8.1.2 No Host Answer

The connected host has not answered to an online request, so it is considered as temporary unavailable. The Cashier has not forced the transaction, which cannot be accepted.

No Host Answer: <Reason>

Reason: transport connection error (see 3.2.4 Transport Error Handling):

- Transport Connection Broken (ERTR02)
- Message Too Big (ERTR04)
- Late Arrival (ERTR05)
- Incomplete Application Message (ERTR07)

4.6.8.2 Refusal Error

The transaction is refused by the Acquirer (or the Host) or the rules associated to the card. The Cashier has not forced the transaction, and the transaction cannot be repeated. A specific message is normally displayed to the Customer and the Cashier if they are present, the information below could be logged for further information:

4.6.8.2.1 Acquirer Decline

<Transaction> Refused by Acquirer: <Acquirer> Reason: <Reason> Code: <Code>

Transaction: type of transaction refused (e.g. payment, reservation...)

Acquirer: name or identification of the Acquirer which has refused

Reason: reason of the refusal

Code: code returned by the Acquirer

4.6.8.2.2 Local Decline

<Transaction> Refused Locally - Reason: <Reason> [Code: <Code>]

Transaction: type of transaction refused (e.g. payment, reservation...)

Reason: reason of the refusal (e.g. PIN error)

Code: internal code if required

4.6.8.3 PaymentRestriction Error

The Customer has used a card restricted on the products the card may pay. Some of the items provided in the *SaleItem* of the request are products that the card cannot pay. The products of the *SaleItem* sequence which could be paid by the card are reported in the *AllowedProductCode* of the Payment response.

<Number> Products not Payable by the <Brand> Card

Number: number of products that the card cannot pay

Brand: brand of the card provided by the Customer

4.7 Error Management

4.7.1 Transaction Status Messages

4.7.1.1 Presentation of the Messages

The Transaction Status request message body *TransactionStatusRequest*, contains the following information:

1. The message on which the status is required by the Sale System: *MessageReference*, to allow the POI identifying this message:
 - a. The category of the original message: *MessageCategory*, which could be Payment, Loyalty, StoredValue, Reversal, CardAcquisition, or CardReaderAPDU
 - b. The identification of the original message: *ServiceID* and *DeviceID*.
 - c. The identification of the sender *SaleID*, and the receiver *POIID* of the original message.
2. A possible flag *ReceiptReprintFlag*, to request the POI to print again the receipt that the original transaction should generate, with the receipt to duplicate in *DocumentQualifier*.

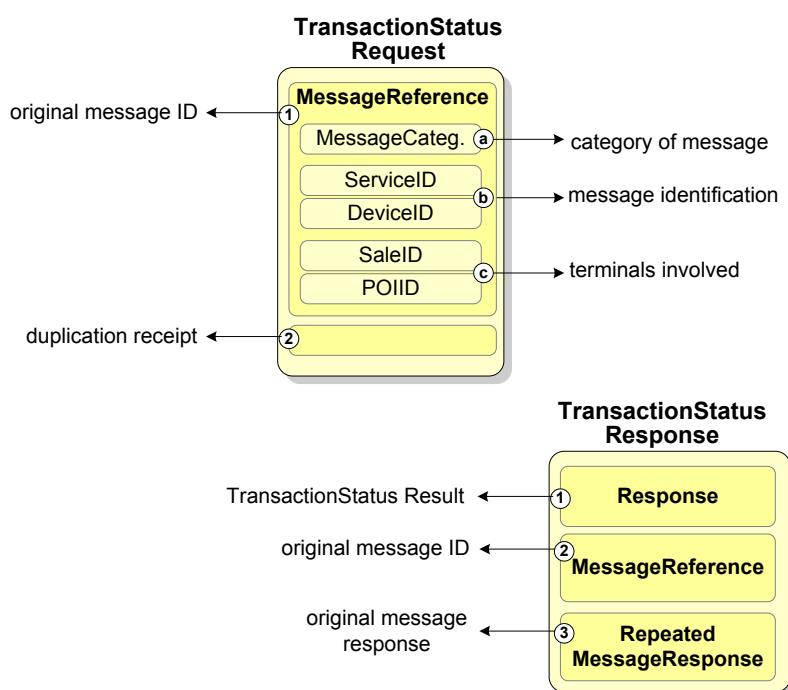


Figure 230: TransactionStatus Request/Response Information

The Transaction Status response message body *TransactionStatusResponse*, contains the following information:

1. The result of the Transaction Status: *Response*.
2. A copy of the *MessageReference* sent in the message request, which may be completed by the POI.
3. A copy of the original response message if the process of the original message is completed.

4.7.1.2 TransactionStatus Request Layout

<i>TransactionStatusRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		TransactionStatus
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
TransactionStatusRequest	[1..1]		
MessageReference	[0..1]		Present if it contains any data
MessageCategory	[0..1]		Payment, Loyalty, StoredValue, Reversal, CardAcquisition, Batch, Reconciliation, or CardReaderAPDU
ServiceID	[0..1]		
DeviceID	[0..1]		
SaleID	[0..1]		default MessageHeader.SaleID
POIID	[0..1]		default MessageHeader.POID
ReceiptReprintFlag	[0..1]		default False
DocumentQualifier	[0..2]		CustomerReceipt or CashierReceipt. Mandatory if ReceiptReprintFlag is True, otherwise absent.

4.7.1.3 TransactionStatus Response Layout

<i>TransactionStatusResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		TransactionStatus
MessageType	[1..1]		Response
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
TransactionStatusResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		<i>same as PaymentResponse</i>
AdditionalResponse	[0..1]		<i>same as PaymentResponse</i>
MessageReference	[0..1]		if Response.Result is Success (process in progress), and present in the request message
MessageCategory	[1..1]		
ServiceID	[0..1]		ServiceID of the requested transaction, if still in progress (to have the possibility to abort it from another Sale Terminal)
DeviceID	[0..1]		DeviceID of the requested transaction, if still in progress (to have the possibility to abort it from another Sale Terminal)
SaleID	[0..1]		Copy
POIID	[0..1]		Copy
RepeatedMessageResponse	[0..1]		If Result is Success (process completed)
MessageHeader	[1..1]		
RepeatedResponseMessageBody	[1..1]		
LoyaltyResponse	[0..1]		
PaymentResponse	[0..1]		
ReversalResponse	[0..1]		
StoredValueResponse	[0..1]		
CardAcquisitionResponse	[0..1]		
CardReaderAPDUREsponse	[0..1]		

4.7.1.4 Transaction Status Processing

Transaction Status message is initiated by the Sale System to get the result of a previous transaction, because the response message was not received by the Sale System.

The previous transaction is identified by identifiers of the message request which has initiated the transaction, this message is also called original message.

Typical use of this message pair is illustrated in the figure below: the payment transaction is terminated but the Payment response message has not been received by the Sale Terminal. In the Transaction Status response message, the POI put the outstanding Payment response message.

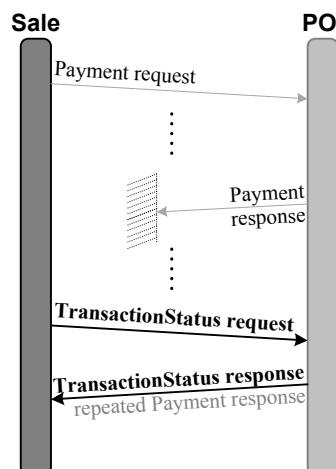


Figure 231: Transaction Status Typical Exchange

Rule 1: The POI Terminal shall be ready to answer to the Transaction Status of the last transaction processed or in progress by the POI Terminal.

Rule 2: If the transaction identified by the *MessageReference* of the Transaction Status request is not found by the POI Terminal, the Transaction Status response is sent with *Result*="Failure" and *ErrorCondition*="NotFound" (see section 4.6.4.4.2 *Message Not Found*).

Rule 3: In the Transaction Status request message, in the component *MessageReference*:

- *SaleID*, identification of the sender of the original message is mandatory if the Transaction Status is not sent by the same Sale Terminal.
- *POIID*, identification of the receiver of the original message is mandatory if the Transaction Status is not sent to the same POI Terminal.

Rule 4: If they are present in the TransactionStatus message, the data elements *MessageReference.MessageCategory*, *ServiceID* and *DeviceID* are copied from the original request.
If they are absent from the message request, the Transaction Status of the last message request processed by the POI Terminal for this Sale Terminal will be delivered.

This rule allows the Sale System to request the status of the last message processed by the POI Terminal for this Sale Terminal. Only one message request may be processed at a time for a pair of Sale/POI Terminals.

Rule 5: If the Transaction Status request contains the *ReceiptReprintFlag* set to “True”, the POI must initiate a reprint of the original transaction:

- (a) The original transaction did not include a payment receipt: A Transaction Status response is sent with *Result*=“NotAllowed” and *ErrorCondition*=“InProgress” (see section 4.6.4.1.4 *NotAllowed Value*).
- (b) The payment receipt is printed on the POI terminal:
 - The POI prints a duplicate of the payment receipt on its printer.
 - The POI sends Transaction Status response is sent with *Result*=“Success” and *RepeatedMessageResponse* absent, whatever the result of the printing.
- (c) The payment receipt is printed on the Sale terminal, and the POI terminal may send *PrintRequest* messages:
 - The POI sends a Print request message to duplicate of the payment receipt on the Sale printer.
 - The POI sends Transaction Status response is sent with *Result*=“Success” and *RepeatedMessageResponse* absent, whatever the result of the printing.
- (d) The payment receipt is printed on the Sale terminal, and the POI terminal is not able to send *PrintRequest* messages:
 - The POI sends Transaction Status response with *Result*=“Success” and *RepeatedMessageResponse* containing the payment receipt in *PaymentReceipt*.
 - The Sale prints a duplicate of the payment receipt on its printer.

The choice of the payment receipt, CustomerReceipt, CashierReceipt or both, is provided in *DocumentQualifier*.

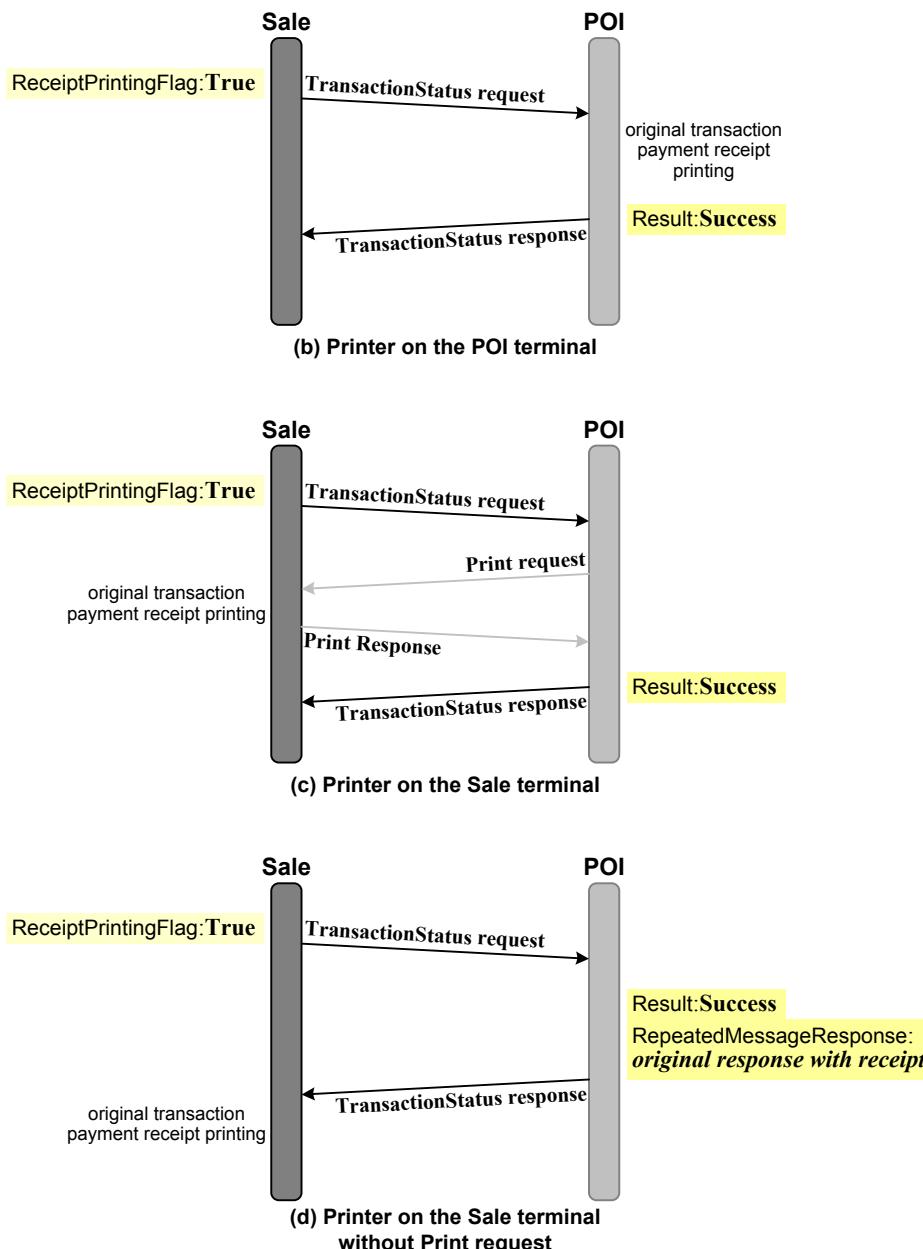


Figure 232: Duplicate Payment Receipt with Transaction Status

Rule 6: If the Transaction Status request is received during the processing of the original message, the Transaction Status response is sent with *Result*="Failure" and *ErrorCondition*="InProgress" (see section 4.6.5.2.1 *Uncompleted Transaction*).

4.7.1.5 Error Cases

When the Transaction Status request is successfully processed, the Transaction Status response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotFound

The POI System receives a Transaction Status request from the Sale System, with a *MessageReference* not stored in the POI System or not in progress in the POI Terminal. (see section 4.6.4.4.2 *Message Not Found*).

InProgress

The POI Terminal cannot provide the response of the message in the Transaction Status response message, because the processing is not complete and still in progress (see section 4.6.5.2.1 *Uncompleted Transaction*).

4.7.1.6 Example

The Sale Terminal SaleTermA requests a payment to the POI Terminal POITerm1.

MessageHeader		(message example 68)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	642	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	579	
TimeStamp	2009-06-07T23:08:42.4+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	104.11	
TransactionConditions		
LoyaltyHandling	Forbidden	
PaymentData		
PaymentType	Normal	

After a crash of the SaleTermA, the Sale System does not receive a response to the payment. Then the Sale Terminal SaleTermB sends a Transaction Status for the previous payment.

MessageHeader		(message example 69)
MessageClass	Service	
MessageCategory	TransactionStatus	
MessageType	Request	
ServiceID	498	
SaleID	SaleTermB	
POIID	POITerm1	
TransactionStatusRequest		
MessageReference		
MessageCategory	Payment	
ServiceID	642	
SaleID	SaleTermA	

MessageHeader		(message example 70)
MessageClass	Service	
MessageCategory	TransactionStatus	
MessageType	Response	
ServiceID	498	
SaleID	SaleTermB	
POIID	POITerm1	
TransactionStatusResponse		
 Response		
Result	Success	
 MessageReference		
MessageCategory	Payment	
ServiceID	642	
SaleID	SaleTermA	
 RepeatedMessageResponse		
 MessageHeader		
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	642	
SaleID	SaleTermA	
POIID	POITerm1	
 PaymentResponse		
 Response		
Result	Success	
 SaleData		
 SaleTransactionID		
TransactionID	579	
TimeStamp	2009-06-07T23:08:42.4+01:00	
 POIData		
 POITransactionID		
TransactionID	481	
TimeStamp	2009-03-10T23:08:42.4+01:00	
POIReconciliationID	200903101	
 PaymentResult		
PaymentType	Normal	
 PaymentInstrumentData		
PaymentInstrumentType	Card	
 CardData		
PaymentBrand	CardPlus	
EntryMode	MagStripe	
 SensitiveCardData		
PAN	0011014570541535	
ExpiryDate	0411	
 AmountsResp		
AuthorizedAmount	104.11	
 PaymentAcquirerData		
AcquirerID	400012	
MerchantID	mer77-130209	
AcquirerPOIID	963276433	
ApprovalCode	8347	

4.7.2 Abort Message

4.7.2.1 Presentation of the Abort Message

The Abort request message body *AbortRequest*, contains the following components:

1. The message on which we are searching the status: *MessageReference*, to allow the POI identifying this message:
 - a. The type of the original message: *MessageCategory*,
 - b. The identification of the original message: *ServiceID* and *DeviceID*.
 - c. The identification of the sender *SaleID*, and the receiver *POIID* of the original message.
2. The reason of aborting the transaction or the device processing, for logging purposes: *AbortReason*.
3. An optional message to display to the Customer: *DisplayOutput* (see the section on Display).

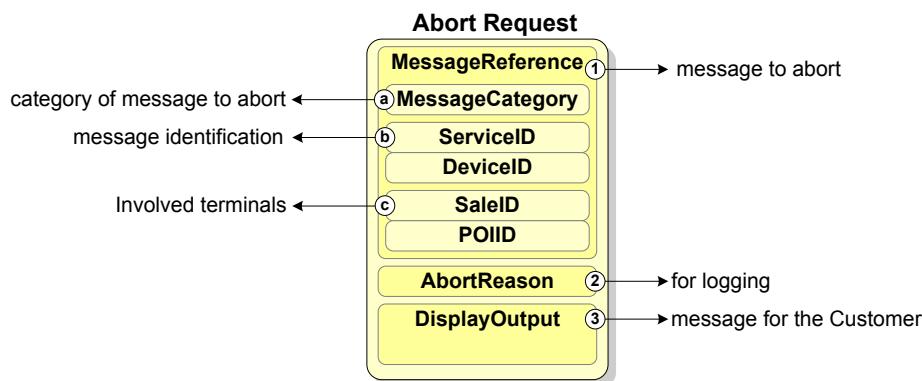


Figure 233: Abort Request Information

4.7.2.2 Abort Request Layout

<i>AbortRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Abort
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
AbortRequest	[1..1]		
MessageReference	[1..1]		
MessageCategory	[0..1]		BalanceInquiry, CardAcquisition, CardReaderInit, CardReaderPowerOff, GetTotals, Input, Loyalty, Payment, Reconciliation, Reversal, StoredValue
ServiceID	[0..1]		Must be present if <i>DeviceID</i> absent.
DeviceID	[0..1]		Must be present if <i>ServiceID</i> absent.
SaleID	[1..1]		default MessageHeader.SaleID
POIID	[1..1]		default MessageHeader.POID
AbortReason	[1..1]		
DisplayOutput	[0..1]		To display an abort message to the Customer
Device	[1..1]		CustomerDisplay
InfoQualify	[1..1]		Error
OutputContent	[1..1]		
OutputFormat	[1..1]		
PredefinedContent	[0..1]		<i>same as Display</i>
ReferenceID	[1..1]		
Language	[0..1]		<i>same as Display</i>
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartRow	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display.

4.7.2.3 Abort Processing

The Abort message allows a Sale Terminal or the Sale Server to halt and terminate prematurely the processing of a message. Most of the time a message is aborted, because the processing is too long or the Sale System is resolving error situation.

Nearly all the message requests the Sale System can send might be aborted:

- The Service requests: CardAcquisitionRequest, LoyaltyRequest, PaymentRequest, ReversalRequest, StoredValueRequest.
- The Device requests: InputRequest, PINRequest, ReadCardInitRequest, ReadCardPowerOffRequest.
- A limited number of Administrative requests: BalanceInquiryRequest, ReconciliationRequest, GetTotals.

Rule 1: When the Sale System sends an Abort request which cannot be accepted (e.g. MessageFormat error):

- a) The POI sends an Event Notification containing the EventToNotify “Reject” and EventDetails, with the information normally located in the AdditionalResponse component.
- b) The Abort request is ignored.

Rule 2: POI Terminals or the POI Server is not allowed to send an Abort Request to the Sale System. The receipt of an Abort request message by the Sale System is ignored.

Rule 3: In the Abort request component *MessageReference*:

- The data elements *MessageCategory*, *ServiceID* and *DeviceID* are copied from the original request,
If one of these mandatory fields is absent, an Event Notification is sent, containing the EventToNotify “Reject” and EventDetails described in section 4.6.2.1.2 Mandatory Data Absent).
- *SaleID*, and *POIID* are not used, an Abort is always sent by the same couple of Sale/POI Terminal. If *SaleID* or *POIID* are present, they must have the same value than the related fields in the header.

Rule 4: If the *transaction identified by the MessageReference* of the Abort request is not found, the POI Terminal sends an Event Notification containing the EventToNotify “Reject” and EventDetails described in the section 4.6.4.4.2 *Message Not Found*.

Rule 5: When the Sale System sends an Abort request on a message in progress, the standard process to respect is the following:

- The processing of the message to abort is stopped as soon as possible.
- All the operations completed during the processing of the message have to be undone by the POI.
- The POI sends the response of the message to abort, with *Result* set to "Failure", *ErrorCondition* set to "Aborted", see section 4.6.6.1 *Aborted Error*.
There is never an Abort response message sent by the POI.

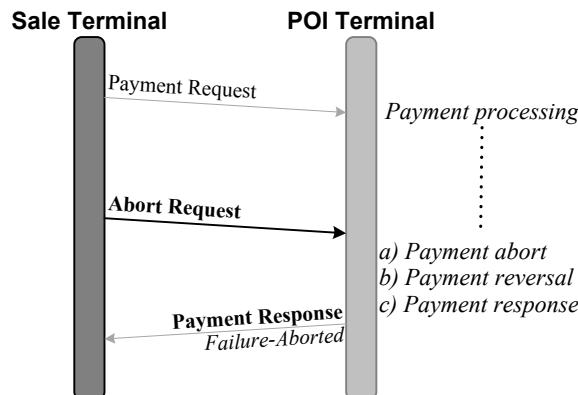


Figure 234: Standard Processing of an Abort

Rule 6: When the Sale System sends an Abort request on a message in progress and the abort is not possible, the POI continues the processing and sends the response to the original message.

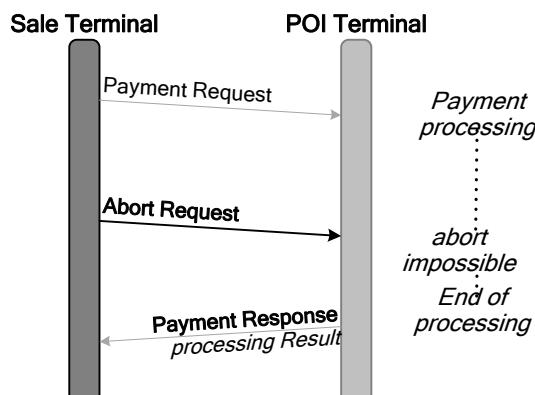


Figure 235: Abort Processing Impossible

Rule 7: When the Sale System sends an Abort request on a message which is completed, the POI sends an Event Notification containing the *EventToNotify* "CompletedMessage" and *EventDetails*, with the information described in section 4.6.6.1 *Aborted Error*.

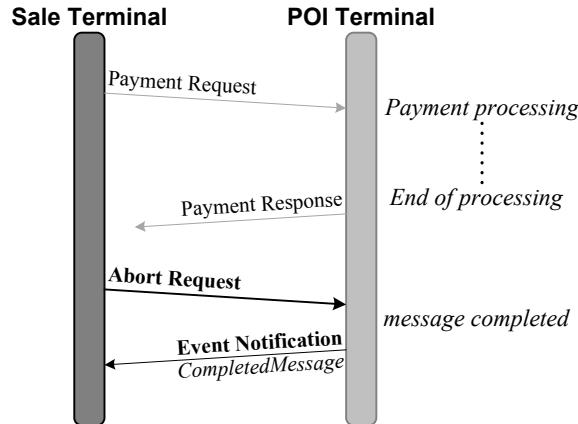


Figure 236: Abort of a Terminated Processing

This rule is valid for crossing of the Abort request and Service/Device response message. So the Sale System has to be ready to receive a response message and an Event notification after *sending* of an Abort request.

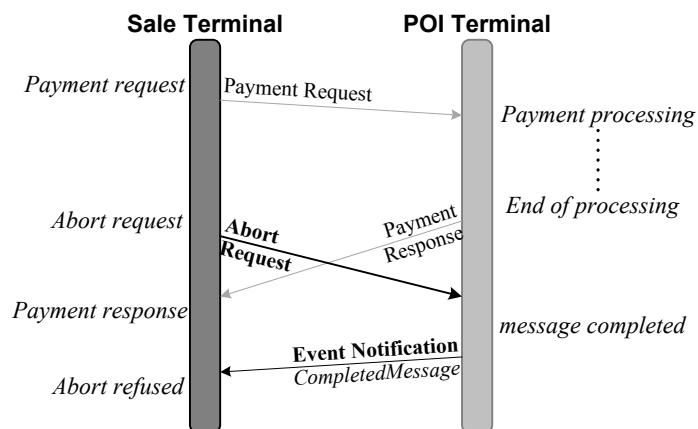


Figure 237: Abort Response Message Crossing

4.7.2.4 Error Cases

The Abort request never has a response. In case of error an Event notification message is sent with the “Reject” *EventToNotify* and the *EventDetails*. These *EventDetails* are enumerated below, listed by the related value of the *ErrorCondition*.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

NotAllowed

The Abort request is to the Sale System by the POI System (see section 3.4.2 *Dialogue Management*, and section 4.6.4.1.3 *Forbidden Message*).

Not Found

The POI System receives an Abort request from the Sale System, with a *MessageReference* not stored in the POI System (see section 4.6.4.4.2 *Message Not Found*).

4.7.2.5 Examples

The Sale Terminal requests a payment to the POI Terminal.

MessageHeader		(message example 71)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Request	
ServiceID	642	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentRequest		
SaleData		
SaleTransactionID		
TransactionID	579	
TimeStamp	2009-06-09T23:12:42.4+01:00	
PaymentTransaction		
AmountsReq		
Currency	EUR	
RequestedAmount	104.11	
TransactionConditions		
LoyaltyHandling	Forbidden	
PaymentData		
PaymentType	Normal	

After a long waiting time, the Sale System does not receive a response to the payment and the Cashier decides to abort the payment transaction.

MessageHeader		(message example 72)
MessageClass	Service	
MessageCategory	Abort	
MessageType	Request	
ServiceID	498	
SaleID	SaleTermA	
POIID	POITerm1	
AbortRequest		
MessageReference		
MessageCategory	Payment	
ServiceID	642	
AbortReason	Waiting time	

The POI aborts the payment and sends the Payment response below to the Sale Terminal.

MessageHeader		(message example 73)
MessageClass	Service	
MessageCategory	Payment	
MessageType	Response	
ServiceID	642	
SaleID	SaleTermA	
POIID	POITerm1	
PaymentResponse		
Response		
Result	Failure	
ErrorCondition	Aborted	
AdditionalResponse	Service Aborted during Online Authorization - Reason: Waiting time - from: SaleTermA - MessageID: 498	
SaleData		
SaleTransactionID		
TransactionID	579	
TimeStamp	2009-06-09T23:12:42.4+01:00	
POIData		
POITransactionID		
TransactionID	481	
TimeStamp	2009-06-09T23:15:12.4+01:00	
POIReconciliationID	200906091	
PaymentResult		
PaymentType	Normal	
PaymentInstrumentData		
PaymentInstrumentType	Card	
CardData		
PaymentBrand	CardPlus	
EntryMode	MagStripe	
SensitiveCardData		
PAN	0011014570541535	
ExpiryDate	0411	
AmountsResp		
AuthorizedAmount	104.11	

4.7.3 Event Notification Message

4.7.3.1 Presentation of the Event Notification Message

The Event notification message body *EventNotification*, contains the following components:

1. The date and time when the event has occurred: *TimeStamp*.
2. The type of event that has occurred on the POI: *EventToNotify*.
3. Additional information on the event: *EventDetails*, which has to be logged and may contain for some events the same information than in *AdditionalResponse* (see section 4.6 Error Reporting and Error Cases).
4. Specific information related to event as: *RejectedMessage*, *MaintenanceRequiredFlag*, *CustomerLanguage*.
5. An optional message to display to the Cashier or other operator: *DisplayOutput*.

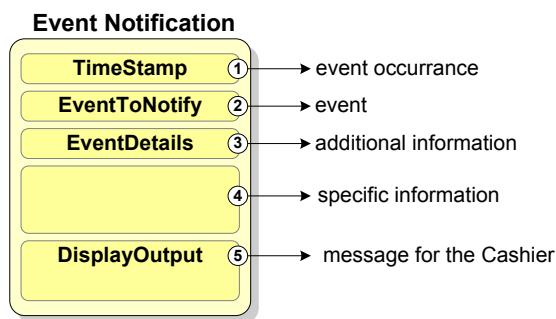


Figure 238: Event Notification Information

4.7.3.2 Event Notification Layout

<i>EventNotification Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
MessageClass	[1..1]		Event
MessageCategory	[1..1]		Event
MessageType	[1..1]		Notification
DeviceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
EventNotification	[1..1]		
TimeStamp	[1..1]		
EventToNotify	[1..1]		
EventDetails	[0..1]		Mandatory for transaction identification when <i>EventToNotify</i> = "SaleWakeUp", "FunctionKeyPressed" or "SaleAdmin". Otherwise if present, the Sale logs it for further examination
RejectedMessage	[0..1]		Mandatory if <i>EventToNotify</i> is "Reject", absent in other cases
MaintenanceRequiredFlag	[0..1]		default False
CustomerLanguage	[0..1]		Mandatory when <i>EventToNotify</i> = "CustomerLanguage", otherwise absent.
DisplayOutput	[0..n]		To display an event message
Device	[1..1]		CashierDisplay
InfoQualify	[1..1]		Error
OutputContent	[1..1]		
OutputFormat	[1..1]		Text, XHTML
OutputText	[0..n]		<i>same as Display</i>
Text	[1..1]		<i>same as Display</i>
CharacterSet	[0..1]		<i>same as Display</i>
Font	[0..1]		<i>same as Display</i>
StartRow	[0..1]		<i>same as Display</i>
StartColumn	[0..1]		<i>same as Display</i>
Color	[0..1]		<i>same as Display</i>
CharacterWidth	[0..1]		<i>same as Display</i>
CharacterHeight	[0..1]		<i>same as Display</i>
CharacterStyle	[0..1]		<i>same as Display</i>
Alignment	[0..1]		<i>same as Display</i>
OutputXHTML	[0..1]		<i>same as Display</i>
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display.

4.7.3.3 Event Notification Processing

An Event Notification message is sent by the POI System, a POI Terminal or the POI Server, when an event occurs which could disturb the POI availability, and the POI cannot inform the Sale System because:

- The POI is not processing a request from the Sale System,
- The POI is processing a request from the Sale System, but the event couldn't be reported in the response message,
- The message request sent by the Sale System does not support response message or the request message is not understood and cannot be answered.

The Sale System has to be aware that the POI can send event only when a transport connection is available. The transport connection(s) to be used for this purpose depends on the configuration of the logical connections.

Rule 1: The flag *MaintenanceRequiredFlag* could be sent with the following events: Shutdown, OutOfOrder or SecurityAlarm.

Rule 2: The notified events BeginMaintenance and EndMaintenance are sent with EventDetails containing the message described in section 4.6.3.1.5 *Maintenance*. Every request message sent to the POI between these two events will receive a response message with Result to "Failure", ErrorCondition to "DeviceOut", see sections 4.6.3.1.1 *POI Temporary Unavailable* and 4.6.3.1.3 *Device Temporary Out*.

Rule 4: The notified event Reject is sent with EventDetails containing one of the messages described in section 4.6.2.1 *Message Format*. If the sender of the message cannot be decoded, the event Reject is sent to the last logged Sale Terminal. The rejected message is placed in the component *RejectedMessage* which is mandatory for a reject.

Rule 5: The notified event SecurityAlarm is sent with EventDetails containing the message described in section 4.6.3.1.6 *Security Alarm*.

Rule 6: The event "Initialised" shall be sent after every initialisation of the Sale to POI protocol application on the POI. The event "Shutdown" shall be sent at every termination if possible.

Rule 7: The notified event OutOfOrder is sent with EventDetails containing one of the messages described in section 4.6.3.1 *DeviceOut Error*.

Rule 8: The notified events CardInserted and CardRemoved are sent to the logged Sale Terminals if they are configured on the POI Terminal. This event is sent when a card is inserted in the card reader, outside a Service or Device dialogue²⁶.

²⁶ Outside a Service or a Device dialogue, the customer cannot process a transaction, and the insertion and removing of a card appears to not have any effect on the POI terminal.

Rule 9: The event “StopAssistance” is sent to the Sale Terminal which has received an Input request with the CustomerAssistance InfoQualify and the Customer has completed the input before the response from the Sale Terminal.

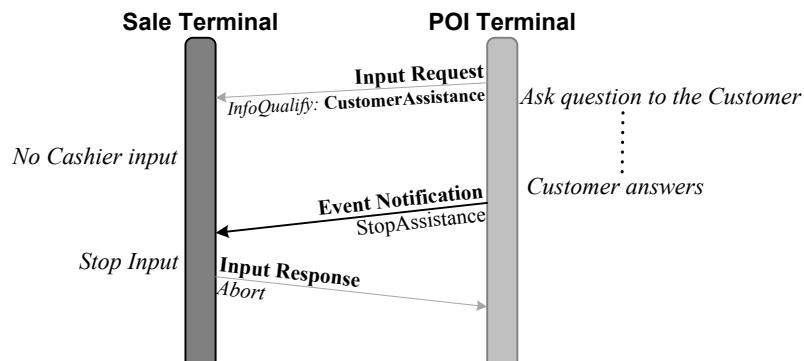


Figure 239: StopAssistance Event

Rule 10: During the processing of a Service request sent by the Sale System, the POI can send Device requests to this Sale Terminal to achieve the processing. If the POI stop the processing of the service before receiving the response of the device request (e.g. user cancellation), the POI has to send an event Abort to the Sale Terminal before to send the service message response.

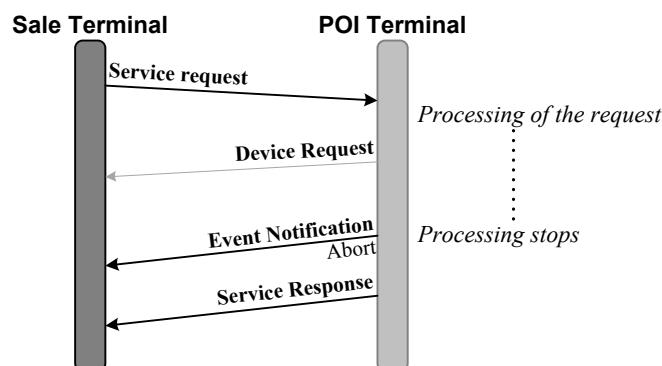


Figure 240: Abort Event

Rule 11: A configuration parameter indicates if the POI System (i.e. the POI Server and the POI Terminals) may send Event notification outside a session.

Rule 12: The notified event SaleWakeUp is sent by the POI to ask the payment of the transaction identified by the value of *EventDetails*. A detailed use-case for this event is described in section 4.3.8.1 *Pay at the Table*.

Rule 13: The POI could manage a logical screen allowing the customer to change the display language.
When the customer select a new language, outside the processing of a Sale system request, or during the processing a request from the Sale System, but the new language couldn't be reported in the response message, the POI must send an Event notification with:

- *EventToNotify* = "CustomerLanguage",
- The new language selected by the customer in *CustomerLanguage*.

Rule 14: The POI could manage specific buttons dedicated to some function (e.g. a button to call an operator on an unattended POI). The customer can press spontaneously on these buttons at any time, whatever the processing in progress.

When the customer presses the button, the POI must send an Event notification with:

- *EventToNotify* = "KeyPressed",
- The identification of the key pressed by the customer in *EventDetails*.

Rule 15: When the POI has performed, or want to perform an automatic administrative process, e.g. the reports at the end of day, the POI may send an Event notification with:

- *EventToNotify* = "SaleAdmin",
- The related service in *EventDetails*.

Name	Event Notification Outside Sessions
<i>Definition</i>	The POI Server and the POI Terminals may send Event Notification outside sessions.
<i>Usage</i>	Allows the Sale System to know activity and problem occurring in the POI System.
<i>Specification</i>	4.7.3 Event Notification Message

Configuration 12: Event Notification Outside Sessions

4.7.3.4 Example

MessageHeader*(message example 74)*

MessageClass	Event
MessageCategory	Event
MessageType	Notification
DeviceID	1328
SaleID	SaleTermA
POIID	POITerm1

EventNotification

TimeStamp	2009-12-13T23:11:16.4+01:00
EventToNotify	BeginMaintenance
EventDetails	POITerm1 Maintenance: Parameters Update

4.7.4 Diagnosis Messages

4.7.4.1 Presentation of the Messages

The Diagnosis request message body *DiagnosisRequest*, contains the following information:

1. The POI Terminal subject of the diagnosis: *POIID*, if the request message is sent by another Terminal.
2. A flag to request a diagnosis of the Acquirer hosts: *HostDiagnosisFlag*.

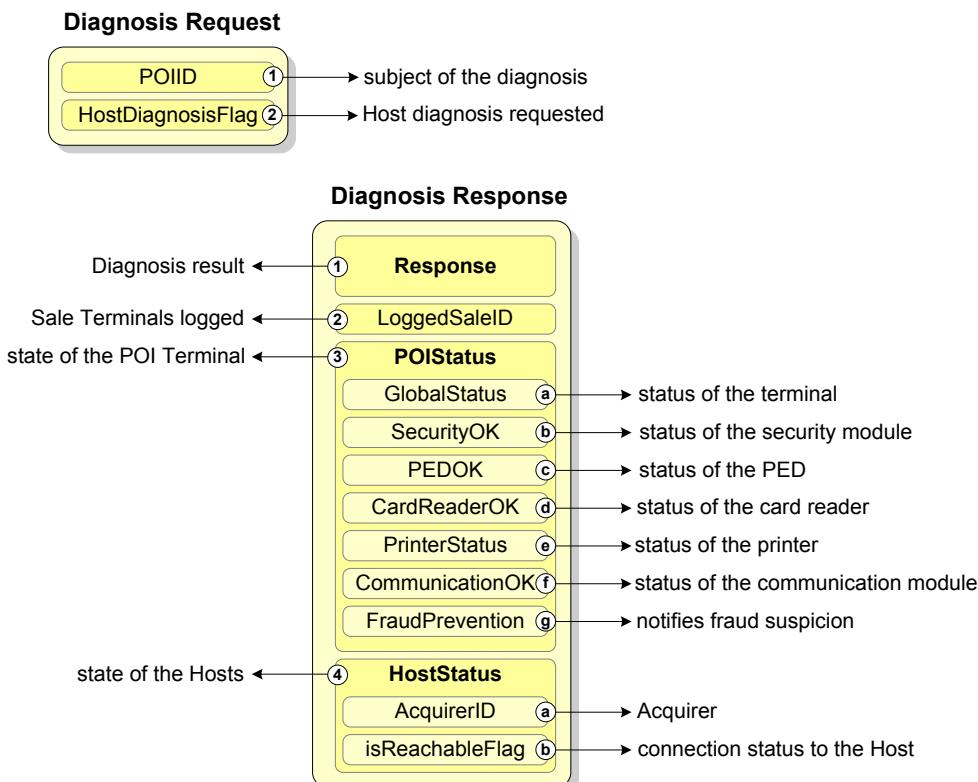


Figure 241: Diagnosis Request/Response Information

The Diagnosis response message body *DiagnosisResponse*, contains the following information:

1. The global result of the Diagnosis: *Response*.
2. The set of the Sale Terminals which are logged to this POI Terminal: *LoggedSaleID*.
3. The state of the various modules of the POI Terminal *POIStatus*:
 - a. The overall status of the POI Terminal: *GlobalStatus*.
 - b. The status of the security module: *SecurityOKFlag*.
 - c. The status of the PIN Entry Device: *PEDOKFlag*.
 - d. The status of the card reader: *CardReaderOKFlag*.
 - e. The status of the printer: *PrinterStatus*.
 - f. The status of the communication module: *CommunicationOKFlag*.
 - g. Suspicion of fraud detected by the POI System: *FraudPreventionFlag*.
4. If requested, the state of the Acquirer Hosts *HostStatus*:
 - a. The identification of the Acquirer: *AcquirerID*.

- b. The status of connectivity to the Host: *IsReachableFlag*.

4.7.4.2 Diagnosis Request Layout

<i>DiagnosisRequest Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
ProtocolVersion	[1..1]		
MessageClass	[1..1]		Service
MessageCategory	[1..1]		Diagnosis
MessageType	[1..1]		Request
ServiceID	[1..1]		
SaleID	[1..1]		
POIID	[1..1]		
DiagnosisRequest	[1..1]		
POIID	[0..1]		default MessageHeader.POID
HostDiagnosisFlag	[0..1]		default False
AcquirerID	[0..n]		Present if requesting the diagnosis of these hosts only.

4.7.4.3 Diagnosis Response Layout

<i>DiagnosisResponse Component</i>	<i>Mult.</i>	<i>Profile</i>	<i>Rule</i>
MessageHeader	[1..1]		
ProtocolVersion	[1..1]		
MessageClass	[1..1]		Copy
MessageCategory	[1..1]		Diagnosis
MessageType	[1..1]		Response
ServiceID	[1..1]		Copy
SaleID	[1..1]		Copy
POIID	[1..1]		Copy
DiagnosisResponse	[1..1]		
Response	[1..1]		
Result	[1..1]		
ErrorCondition	[0..1]		see <i>PaymentResponse</i>
AdditionalResponse	[0..1]		see <i>PaymentResponse</i>
LoggedSaleID	[0..n]		If Sale Terminal logged to this POI Terminal
POIStatus	[0..1]		if Response.Result is Success
GlobalStatus	[1..1]		
SecurityOKFlag	[0..1]		If security module present
PEDOKFlag	[0..1]		If PED present
CardReaderOKFlag	[0..1]		If card reader device present
PrinterStatus	[0..1]		If printer device present
CommunicationOKFlag	[0..1]		If communication infrastructure present
CashHandlingDevice	[0..n]		If cash handling devices present.
CashHandlingOKFlag	[1..1]		
Currency	[1..1]		
CoinsOrBills	[1..n]		
UnitValue	[1..1]		
Number	[1..1]		
FraudPreventionFlag	[0..1]		Default False
HostStatus	[0..n]		If request.HostDiagnosisFlag is True
AcquirerID	[1..1]		
IsReachableFlag	[0..1]		default True

4.7.4.4 Diagnosis Processing

- Rule 1: The Diagnosis message request could be sent to another POI Terminal than the subject of the diagnosis, or to the POI Server. If the POI Terminal receiving the request cannot make the diagnosis of the Terminal, the Diagnosis response is sent with *Result*="Failure" and *ErrorCondition*="UnavailableService" (see section 4.6.4.3.3 *Unavailable Administrative Service*)
- Rule 2: If the POI Terminal receiving the Diagnosis request is not available because another request is processed by the POI, a Diagnosis response is sent with *Result*="Failure" and *ErrorCondition*="Busy" (see section 4.6.5.1.2 *POI Busy*)
- Rule 3: If the POI Terminal subject of the Diagnosis request is not available, a Diagnosis response is sent with *Result*="Failure" and *ErrorCondition*="Busy" (see section 4.6.5.1.1 *Component Unavailable*)
- Rule 4: If the POI Terminal, subject of the Diagnosis, does not include a security module, a PIN Entry Device, a card reader, a printer, or a communication module, the data elements *SecurityOKFlag*, *PEDOKFlag*, *CardReaderOKFlag*, *PrinterStatus* and *CommunicationOKFlag* respectively are absent from the Diagnosis response.
- Rule 5: If the Diagnosis request contains *HostDiagnosisFlag* set to False or absent, the Diagnostic response does not contain any *HostStatus* data structure.
If the Diagnosis request contains *HostDiagnosisFlag* set to True, the Diagnostic response must contain a *HostStatus* data structure for each of the host connected to the POI.
If the Diagnosis request contains *HostDiagnosisFlag* set to True and at least one occurrence of *AcquirerID*, the Diagnostic response must contain a *HostStatus* data structure for each of the host identified by the *AcquirerID* values.
If the Diagnosis request contains *HostDiagnosisFlag* set to False or absent and at least one occurrence of *AcquirerID*, the Diagnostic response does not contain any *HostStatus* data structure.
- Rule 6: When the cash handling machine managed by the POI does not have any more coins or bills of a certain value, the Diagnosis response must contain the related occurrence of *CoinsOrBills* with *Number* equal to 0.

4.7.4.5 Error Cases

When the Diagnosis request is successfully processed, the Diagnosis response message gets the value “Success” in the data element *Response.Result*, and the value “Failure” in case of error. These errors are enumerated below, listed by value of the *ErrorCondition* data element.

MessageFormat

Standard errors are defined in section 4.6.2.1 *Message Format*. These are permanent errors, which have to be resolved without any other attempt.

LoggedOut

The Sale Terminal has never sent a Login message request since the last Logout message sending or the start-up of the POI Terminal. This is the typical error after a crash of the POI Terminal or the POI System.

UnavailableService

The Diagnosis administrative service is not available in the POI System (see section 4.6.4.3.3 *Unavailable Administrative Service*).

The POI cannot make the diagnosis of another POI Terminal.

Busy

The POI is not available to make the Diagnosis.

4.7.4.6 Example

The Sale Terminal sends a diagnosis to the POI Terminal without printer.

MessageHeader		(message example 75)
ProtocolVersion	3.0	
MessageClass	Service	
MessageCategory	Diagnosis	
MessageType	Request	
ServiceID	660	
SaleID	SaleTermA	
POIID	POITerm1	
DiagnosisRequest		
HostDiagnosisFlag	True	

MessageHeader		(message example 76)
ProtocolVersion	3.0	
MessageClass	Service	
MessageCategory	Diagnosis	
MessageType	Response	
ServiceID	660	
SaleID	SaleTermA	
POIID	POITerm1	
DiagnosisResponse		
Response		
Result	Success	
LoggedSaleID	SaleTermA	
POIStatus		
GlobalStatus	OK	
SecurityOKFlag	True	
PEDOKFlag	True	
CardReaderOKFlag	True	
PrinterStatus	OK	
CommunicationOKFlag	True	
HostStatus		
AcquirerID	64592	
IsReachableFlag	True	
HostStatus		
AcquirerID	50265	
IsReachableFlag	True	

4.7.5 Error Resolution and Error Situations

4.7.5.1 Error Resolutions Specifications on the POI System

The following types of error situations may occur on the POI Terminal:

1. *Service Request Processing*: Errors occurring during the processing of a Service request sent by the Sale System.
2. *Device Request Sent to the Sale System*: Error occurring on a Device request sent to the Sale System inside the processing of a Service request.
3. *Outside a Service Request*: Error occurring on the POI outside the processing of any request sent by the Sale System.

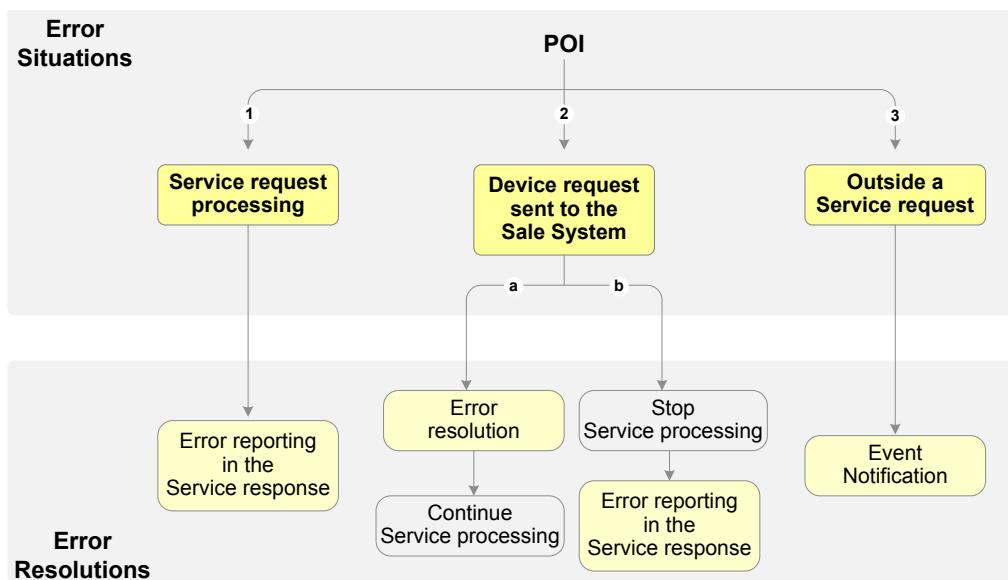


Figure 242: Error Occurring on a POI System

1. Service Request Processing Error Situation

In this case the error is reported to the Sale System inside the Service response, and the Sale System can take the appropriate action. The error reporting is detailed in the section *4.6 Error Reporting and Error Cases*, and every message specification.

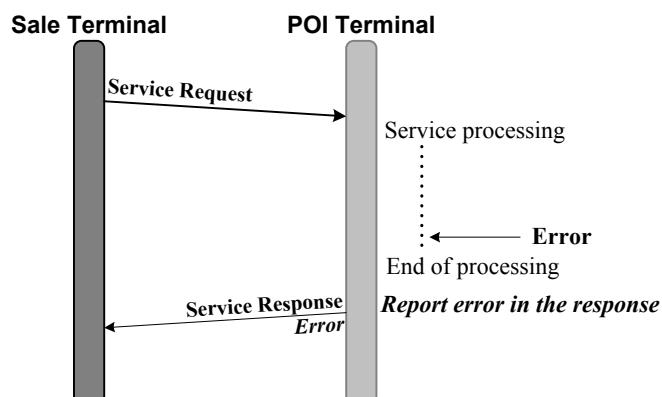


Figure 243: Service Request Processing Error Situation

2. Device Request Sent to the Sale System Error Situation

It could be considered as a sub-case of the previous one which occurs on the Sale to POI interface. The POI decides if the error has detrimental to the general processing of the Service request (e.g. this is not generally the case for a Display request), and either resolve the problem and continue the Service processing, or terminate the Service with an appropriate error case.

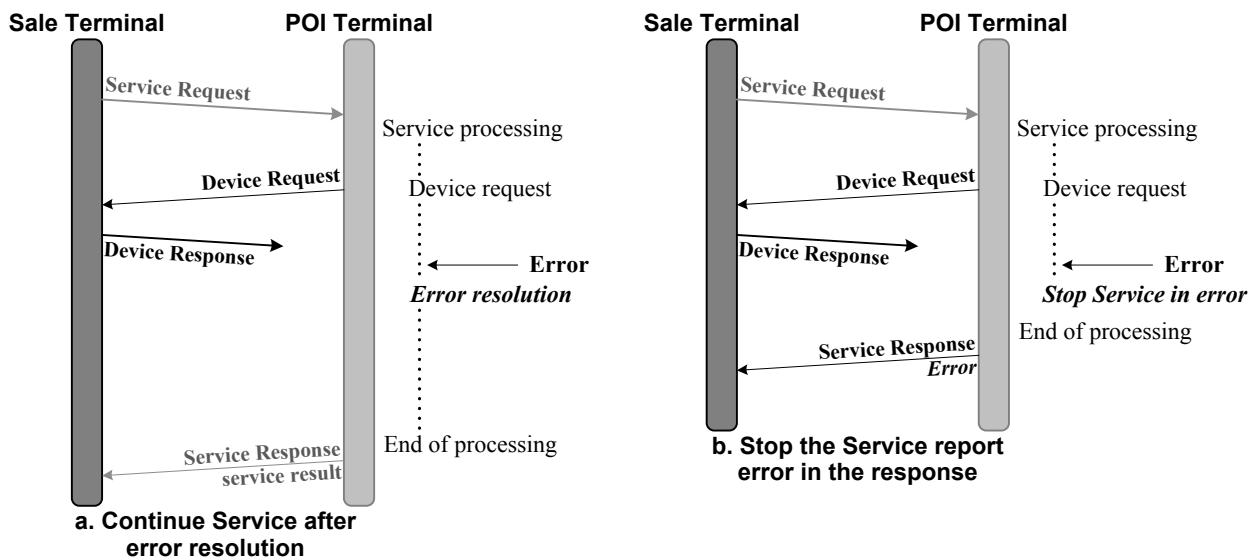


Figure 244: Device Request Sent to the Sale System Error Situations

3. Error Situation outside a Service Request

In this situation if a transport connection is available, the POI sends an Event notification to announce the error to the Sale System.

The error could be the reject of an understandable message, where a response cannot be built from the request.

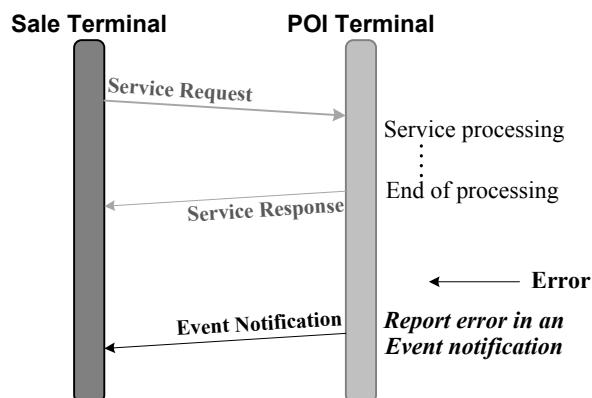


Figure 245: Error Situation outside a Service Request

4.7.5.2 Error Resolutions Specifications on the Sale System

We consider the following types of error situations occurring on the Sale Terminal:

1. *General Error Resolution*: This concerns messages requests which support the standard error resolution algorithm. They cover Payment, Loyalty, StoredValue, Reversal, CardAcquisition, Batch, and CardReaderAPDU messages.
2. *Dedicated Error Resolution*: This concerns messages requests errors which are resolved by an action specific to the involved request. They cover Login, Logout, EnableService, Admin, Reconciliation, Print, Input, CardReaderInit, and CardReaderPowerOff messages.
3. *Idempotent²⁷ Requests*: This concerns messages requests which could be repeated without consequences. They cover Diagnosis, BalanceInquiry, GetTotals, Abort, Batch, and Display messages.

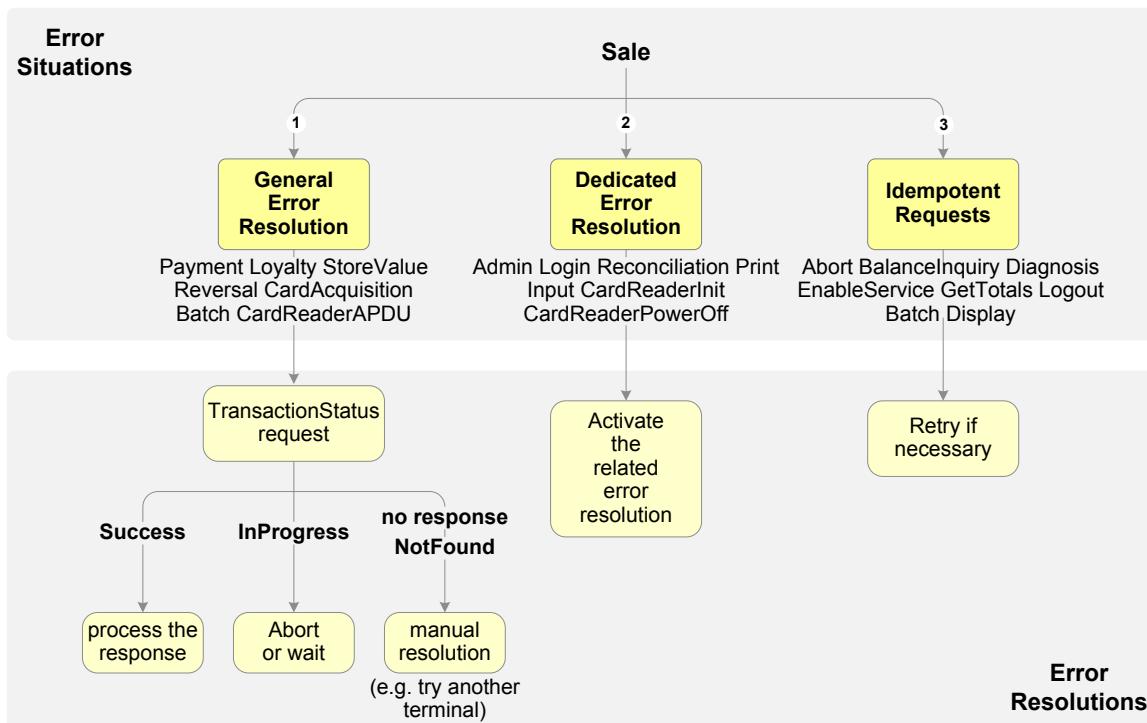


Figure 246: Error Occurring on a Sale System

1. General Error Resolution

When there is no acceptable response to the message request, the Sale System takes the following actions:

- The Sale System sends a TransactionStatus request to know the status of the transaction processing, possibly from a different Sale Terminal or to a different POI Terminal.
- If the Transaction Status is successful (i.e. *Result*="Success"), the response message contains the response message of the original transaction, which is processed.
- If the Transaction Status answers the original transaction is not completed (i.e. *Result*="Failure" and *ErrorCondition*="InProgress"), the Sale System chooses the appropriate resolution for the situation: send an Abort request or wait for the completion of the original transaction.

²⁷ The request could be safely called multiple times without consequence on the state of the receiver, and having the same result.

- If the Transaction Status is in error (e.g. no response in acceptable time, or *Result*="Failure" and *ErrorCondition*="NotFound"), only manual resolution could be undertaken, as try another Sale Terminal or POI Terminal for the exchange of the TransactionStatus messages.

2. Dedicated Error Resolution

For each of the messages in error an appropriate error resolution exists. For instance the Login request without response could be repeated a certain number of times, or a diagnosis could be sent to verify the status of the POI Terminal.

3. Idempotent Requests

The message could be repeated or abandoned according to the context and the environment of the processing.

4.7.5.3 Application Timers

When the application, on the Sale or the POI System, sends a request message and wait for a response to this request, the application must watch the response with a timer in order to:

- Limit the waiting time for the requestor during a real-time process,
- Take the appropriate action in absence of a response to the request,
- Not wait forever, leaving resources busy.

The difficulty for this timer is to fix its value:

- It is impossible to find a reasonable limit of time for the various messages processing, even per class of messages.
- The requirement of the sender of the message could differ, according to the situation, it is inappropriate to fix value in the specifications.
- The variance of the time processing for a message is large, and cannot be fixed by the sender of the message. In addition to the processing time to perform the service, there are many random time such as exchanges between software or hardware modules, request to external host, or user interactions.
- The receiver of the message does not necessary know this time at the beginning of the process, so even a timer negotiation is difficult to realise.

So the main function behind such timer is to avoid blocking forever the sender of the message. Additional strategy could be executed sometimes to refine the timer value. For instance, during the processing of service, the POI sends to the Sale System some device request, at least to show the main step of the processing (e.g. PIN verification, payment host request), the Sale System could then reararm the timer at the reception of these display statuses.

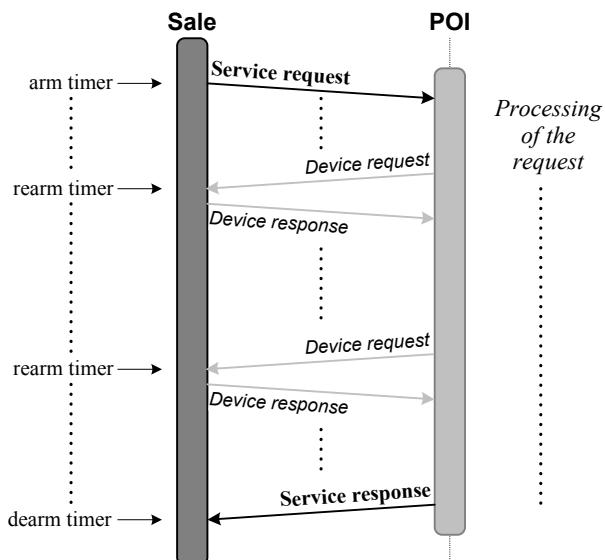


Figure 247: Rearming the Timer on Device requests

The approach of the Retailer protocol specification is to provide clear specifications of error resolution and error management, in order to allow flexible behaviour depending on the class of messages, the context of the error, and the requirements of the requestor.

Possible timer management by Sale System to watch the response to a Service request depends on the type of error resolution to which the request belongs (see section 4.7.5.2 *Error Resolutions Specifications on the Sale System*):

- If the "General Error Resolution" applies, after a timeout, a Transaction Status request has to be sent to the POI, and depending on the response, the Sale System might continue to wait for the completion of the request (see section 4.7.5.5 *Interrogation After Timeout*), or abort the Service request (see section 4.7.5.6 *Abort on Timeout*).
- If the "Dedicated Error Resolution" applies, the timer must be long enough to allow the POI to complete the processing, and avoid too frequent errors.
- If the "Idempotent Error Resolution" applies, the timer must be long enough to allow the POI to complete the processing, but error resolution on timeout is made easier by resubmission of the request.

Concerning the Device requests sent by the POI or the Sale System:

- The following commands include the maximum time to wait for a response, including waiting forever: Input, PIN, CardReaderInit, and CardReaderPowerOff. The requestor may manage closely the response time.
- The commands Display and CardReaderAPDU may require the use of a timer, only to avoid waiting forever.
- Depending on the print mode, Print command requires either no response, or an acknowledgement (belongs to the previous case), or waiting the end of the print with a timer long enough to allow processing completion, and avoid too frequent errors.

4.7.5.4 Loss of Payment Response

This is the typical error when the response to the Payment, Loyalty, StoredValue, Reversal, CardAcquisition, or CardReaderAPDU is lost. The general error resolution mechanism is activated:

1. The Sale Terminal sends a Payment request to the POI Terminal and wait for the Payment response.
2. The POI Terminal processes the transaction related to the Payment request.
3. At the end of the transaction, the POI sends the Payment response message, but this message is lost and not received by the Sale Terminal which continues to wait.
4. The timer monitoring the Payment response expires at the Sale Terminal. The Sale Terminal activates the general error resolution mechanism and send TransactionStatus request for the Payment request.
5. The POI Terminal retrieves the Payment response and sends back the TransactionStatus response contains this Payment response.
6. The Sale Terminal treats the Payment Response and continues the transaction.

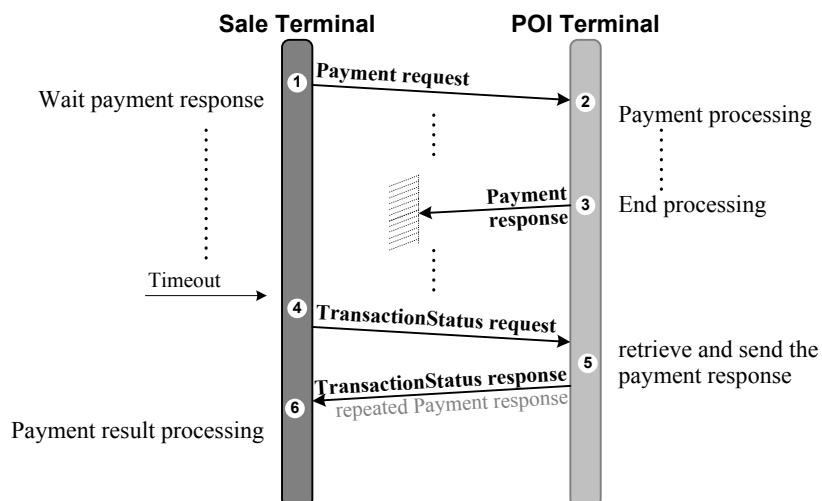


Figure 248: Loss of Payment Response

4.7.5.5 Interrogation After Timeout

This is the typical error when the processing of the Payment, Loyalty, StoredValue, Reversal, CardAcquisition, or CardReaderAPDU message is too long. The general error resolution mechanism is activated:

1. The Sale Terminal sends a Payment request to the POI Terminal and wait for the Payment response.
2. The POI Terminal processes the transaction related to the Payment request.
3. The timer monitoring the Payment response expires at the Sale Terminal. The Sale Terminal activates the general error resolution mechanism and sends a TransactionStatus request for the Payment request.
4. The Payment transaction related to the Payment request message identifier, is not finished and is still in progress. The POI Terminal sends back the TransactionStatus response containing *Result*=“Failure” and *ErrorCondition*=“InProgress”.
5. At the reception of the TransactionStatus, the Sale terminal waits to the Payment response message.
6. At the end of the transaction, the POI sends the Payment response message.
7. The Sale Terminal treats the Payment Response and continues the transaction.

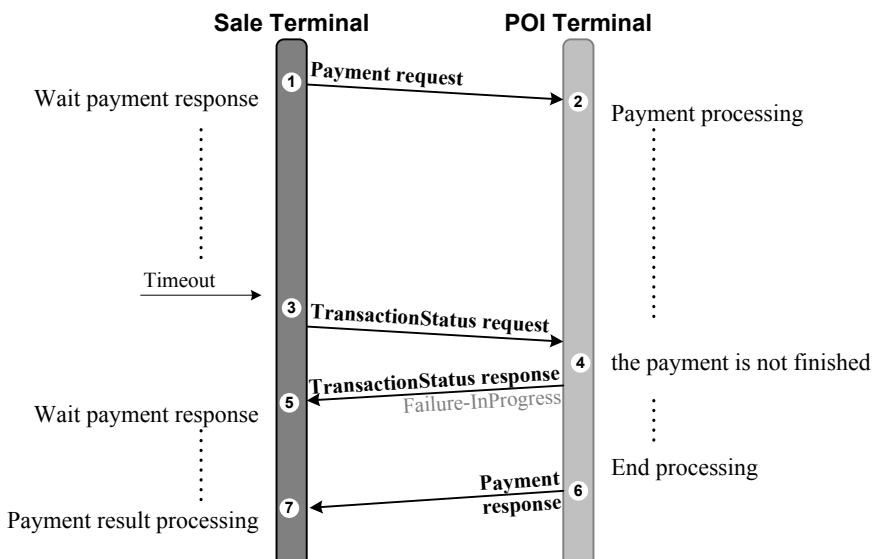


Figure 249: Timeout on Waiting Payment Response

Sometimes, it could be difficult to adjust the timeout, particularly for an external exchange and user action on the interface.

To attenuate these disparities, the Sale Terminal could re-arm the timer of the waiting for response each time the Sale Terminal receives a device request of the POI related to the service in progress.

It is also a good practice to send a display status in attended situation, particularly when there is an expected waiting time as PIN verification or online authorisation.

4.7.5.6 Abort on Timeout

Sometimes the Cashier or the Sale Terminal application decides to abort the payment after a timeout. The Abort request message is sent directly after the timeout or more often to be sure that the payment is not finished, after the exchange of Transaction Status messages:

1. The Sale Terminal sends a Payment request to the POI Terminal and waits for the Payment response.
2. The POI Terminal processes the transaction related to the Payment request.
3. The timer monitoring for the Payment response expires at the Sale Terminal. The Sale Terminal activates the general error resolution mechanism and send TransactionStatus request for the Payment request.
4. The Payment transaction related to the Payment request message identifiers, is not finished and is still in progress. The POI Terminal sends back the TransactionStatus response containing *Result*="Failure" and *ErrorCondition*="InProgress".
5. At the reception of the TransactionStatus, the Sale Terminal (and the Cashier) notices that the payment is in progress, and decides to abort the payment.
6. The Sale Terminal sends an Abort request message.
7. The POI receives the Abort request message, aborts the payment, and sends the Payment response message.
8. The Sale Terminal receives the Payment Response with the aborted result, and continues the transaction.

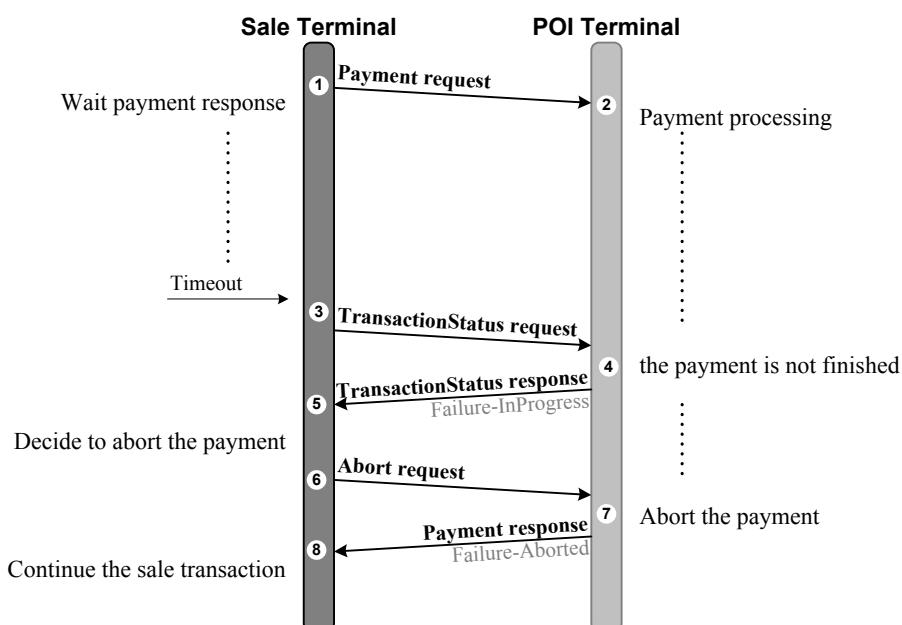


Figure 250: Abort after Payment Response Timeout

4.7.5.7 Transport Connection Broken

The response message to a Payment, Loyalty, StoredValue, Reversal, CardAcquisition, or CardReaderAPDU is sent in the transport connection on which the request message was received.

When a new transport connection is opened by the Sale terminal, there is no Login required by the protocol, and in some cases, the POI Terminal does not know the owner of the connection. The general error resolution mechanism is activated by the Sale System:

1. The Sale Terminal sends a Payment request to the POI Terminal on an open transport connection 1, and waits for the Payment response on the same connection 1.
2. The POI Terminal processes the transaction related to the Payment request.
3. The transport connection 1 is broken by the communication infrastructure. Sale Terminal and POI Terminal receive both a disconnection indication. The POI continues the payment processing (in some case, it might abort the payment)
4. The Sale Terminal opens a new transport connection 2.
5. The Sale Terminal wants to have the result of the payment transaction, and activates the general error resolution mechanism. It sends Transaction Status request for the Payment request.
6. The Payment transaction is not finished. The POI Terminal sends back the TransactionStatus response containing *Result*=“Failure” and *ErrorCondition*=“InProgress”.
7. At the reception of the TransactionStatus, the Sale terminal waits to the Payment response message.
8. At the end of the transaction, the POI sends the Payment response message.
9. The Sale Terminal treats the Payment Response and continues the transaction.

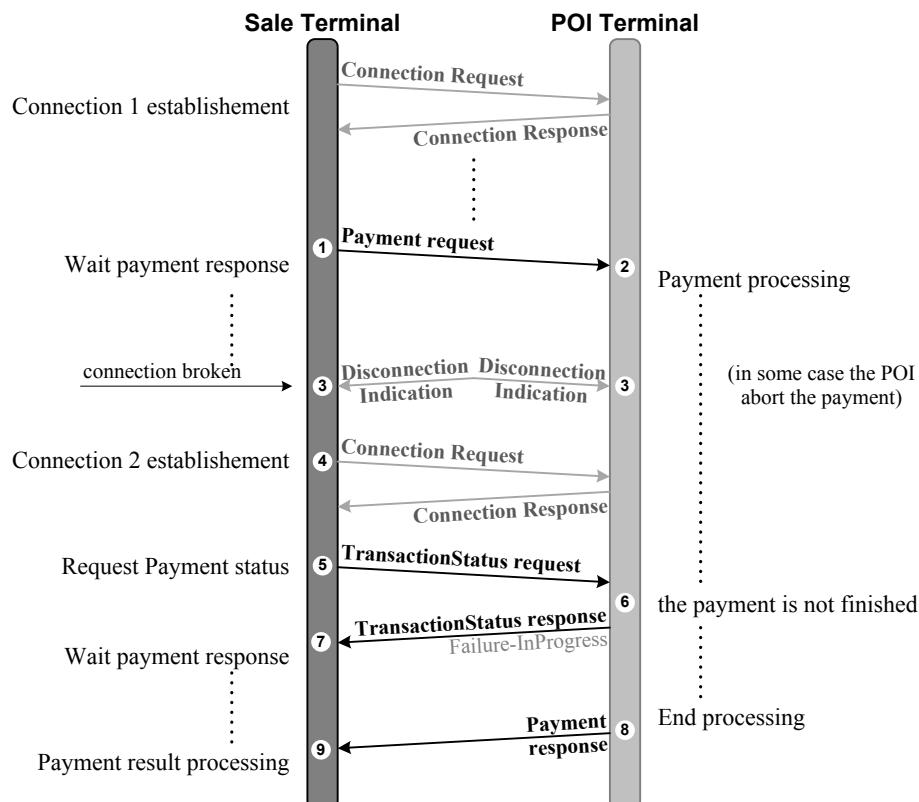


Figure 251: Transport Connection Broken before Payment Response

If the transport connection is broken before the response to a device request instead of a payment request:

- The Sale Terminal has the initiative to establish the new transport connection. If the transport connection is reopened, the POI Terminal sends the Device response on the new transport connection.
- If the POI Terminal has to send a Device response without an established transport connection, depending on the device request and on the implementation, it continues the process or terminates with an error (e.g. after a Display Status it continues the process, but after a receipt printing it could stop the payment with an error for some card brand).

4.7.5.8 POI Terminal Crash During a Session

The POI Terminal could crash inside a session.

- When the POI Terminal start-up, it has lost its login states.
- At the reception of the first Message Request from the Sale Terminal which was logged on this Terminal, the POI send a Message Response with *Result*="Failure" and *ErrorCondition*="LoggedOut" (see section 4.6.4.2 *LoggedOut Error*).
- The Sale System has to send a new Login Request to the POI Terminal in order to be able to continue to work with this Terminal and resubmit the refused Message Request.

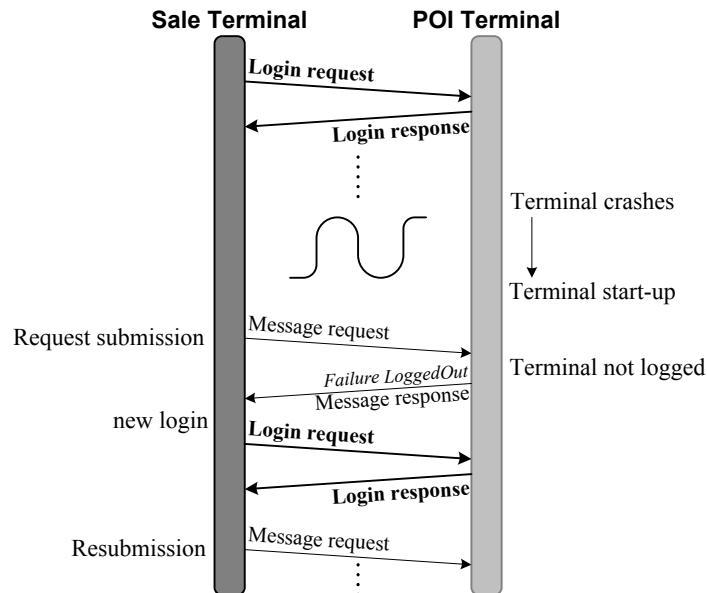


Figure 252: POI Terminal Crash During a Session

The Message Request sent after the POI Terminal crash could be the beginning of the error processing because of no response to a request sent before the crash.

4.7.5.9 User Cancellation During a Pending Device Request

During a Payment transaction, the Customer has the possibility to cancel the transaction, usually with a specific key on the POI Terminal keyboard.

If a Device request was in progress on the Sale Terminal requiring some processing or Cashier waiting time (e.g. Print or Input), the POI Terminal has to cancel the payment processing, and sends appropriate response to the Sale Terminal.

On the other side, the Sale Terminal has to monitor the uncompleted Device requests when it receives the response of the Payment and makes an internal abort:

1. The POI Terminal processes the transaction related to the Payment request.
2. During the payment the POI Terminal requests some information to the Cashier and sends to the Sale Terminal an Input request.
3. The Sale Terminal processes the Input request.
4. During the PIN verification, the Cardholder cancels the transaction. The POI Terminal aborts the payment and sends:
 - a. An event Notification with *EventToNotify*= "Abort", and
 - b. The Payment response with *Result*="Failure" and *ErrorCondition*="Cancel".
5. The Sale Terminal takes into account the payment response and aborts the Input process in progress.
6. The Sale Terminal sends the Input response with *Result*="Failure" and *ErrorCondition*="Cancel".
7. The POI Terminal which has completed the Payment transaction, ignores the Input response message.

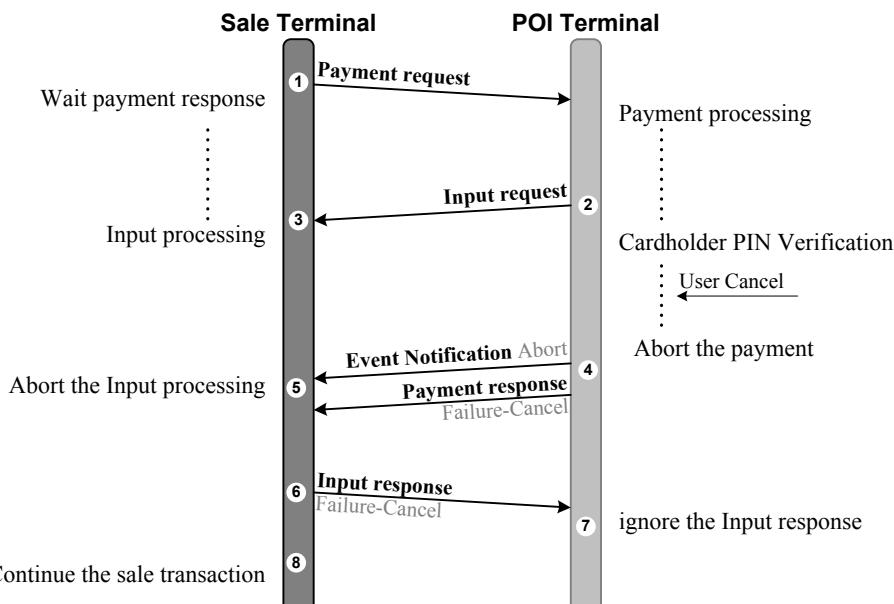


Figure 253: User Cancellation during a Pending Device Request

5 Data Definition

5.1 Messages Definition

This section presents the global format of the Sale to POI protocol messages.

There are two structures, one for the request or notification messages, another one for the response messages. These two data structures contain first the message header, common to all messages, followed by a body dedicated to the type of message.

For coding reasons, the body of the message is defined by a choice between the various alternatives: *RequestMessageBody* and *ResponseMessageBody*.

5.1.1 Request Message

Or	Component	Mult.	Profile	Rule
	SaleToPOIRequest	[1..1]		
	MessageHeader	[1..1]		
	RequestMessageBody	[1..1]		
{Or}	AbortRequest	[1..1]		
	BalanceInquiryRequest	[1..1]		
	BatchRequest	[1..1]		
	CardAcquisitionRequest	[1..1]		
	AdminRequest	[1..1]		
	DiagnosisRequest	[1..1]		
	DisplayRequest	[1..1]		
	EnableServiceRequest	[1..1]		
	EventNotification	[1..1]		
	GetTotalsRequest	[1..1]		
	InputRequest	[1..1]		
	InputUpdate	[1..1]		
	LoginRequest	[1..1]		
	LogoutRequest	[1..1]		
	LoyaltyRequest	[1..1]		
	PaymentRequest	[1..1]		
	PINRequest	[1..1]		
	PrintRequest	[1..1]		
	CardReaderAPDUREquest	[1..n]		
	CardReaderInitRequest	[1..1]		
	CardReaderPowerOffRequest	[1..1]		
	ReconciliationRequest	[1..1]		
	ReversalRequest	[1..1]		
	SoundRequest	[1..1]		
	StoredValueRequest	[1..1]		
	TransactionStatusRequest	[1..1]		
Or}	TransmitRequest	[1..1]		

Or	Component	Mult.	Profile	Rule
	SecurityTrailer	[0..1]		

5.1.2 Response Message

Or	Component	Mult.	Profile	Rule
	SaleToPOIResponse	[1..1]		
	MessageHeader	[1..1]		
	ResponseMessageBody			
{Or}	BalanceInquiryResponse	[1..1]		
	CardAcquisitionResponse	[1..1]		
	BatchResponse	[1..1]		
	AdminResponse	[1..1]		
	DiagnosisResponse	[1..1]		
	DisplayResponse	[1..1]		
	EnableServiceResponse	[1..1]		
	GetTotalsResponse	[1..1]		
	InputResponse	[1..1]		
	LoginResponse	[1..1]		
	LogoutResponse	[1..1]		
	LoyaltyResponse	[1..1]		
	PaymentResponse	[1..1]		
	PINResponse	[1..1]		
	PrintResponse	[1..1]		
	CardReaderAPDUREsponse	[1..n]		
	CardReaderInitResponse	[1..1]		
	CardReaderPowerOffResponse	[1..1]		
	ReconciliationResponse	[1..1]		
	ReversalResponse	[1..1]		
	StoredValueResponse	[1..1]		
	SoundResponse	[1..1]		
	TransactionStatusResponse	[1..1]		
Or}	TransmitResponse	[1..1]		
	SecurityTrailer	[0..1]		

5.2 Data Dictionary

This section contains the data dictionary of the protocol, which provides exhaustive definitions of all the messages and their components. An introduction presents the organization of the data dictionary entries and the used notation.

All the entries are listed by their name in alphabetical order, and most of the references to the name of an entry contain a link to the corresponding entry in the data dictionary.

5.2.1 Introduction

This section contains the detailed definition of all data that could be present in the messages of the Acquirer protocol.

A data is either:

- A *data element*, a primitive data that does not contain other data.
- A *data structure*, a data that could contain other data elements or data structures.

The section 5.2.2 *Data Elements and Structures*, contains the definition of these data element and data structure in alphabetical order.

As we will see in following section, every data has a type. The section 5.2.3 *Data Types*, contains the definition of all the types used by several data elements or data structures.

Inside these two sections, data are defined with the attributes presented in the figure below, and explained afterwards.

Name	APDUData	Name of the data
Definition	Data field of the APDU command (Lc + Data)	
References	ISO 7816-4	
Usage	APDU request for card read device request. APDU response for card read device response.	
Type	ByteSequence	
Format	{1,255}	
CrossRef	ReadCardAPDUDRequest ReadCardAPDUDResponse	
XMLCoding		
ASN1Coding		

Figure 254: Data Definition Attributes

5.2.1.1 Data Names

The *name* is the identification of the data element or data structure in plain English.

In most of the cases, the name is unique among the protocol messages, and can identify unambiguously the data element or the data structure.

However, in some circumstances, the same data name is used in several locations, with a different meaning or content (e.g. MessageBody, Type, and Value). In this case, the identification needs to be completed by the context of the data, as in the section 5.2.2 *Data Elements and Structures*, with the message name or the name of the enclosing data structure.

The general naming convention rules of the ISO 20022 Repository are used to avoid further translations:

1. Use the British English vocabulary.
2. Observe the (character) restrictions that are typical for most syntaxes when naming elements:
 - All names must start with an alphabetic character.
 - All characters following the first characters must be alphabetic characters or numeric characters.
3. Apply camel case convention:
 - A name for elements and attributes may be made up of multiple words, each consisting of alphanumeric characters.
 - Each word starts with a capital letter.
 - All white spaces between words are removed.

5.2.1.2 Data Definition

It contains the semantic of the data which explains its meaning.

Definition is concise, and is independent from data syntax.

Definition does not express the way the data could be used, but what the data represents.

5.2.1.3 References

It indicates the origin(s) of the data definition, and the identification of the data inside the origin.

These references does not necessary correspond exactly to the data element defined in the Acquirer protocol.

There could be multiple references of the data element, the most general appears first.

Reference data can be altered or modified.

Reference origin must be publicly available.

Identification of the data may be a name and other identification as an XML tag name (e.g. for ISO 20022 tag), a BER-TLV tag (e.g. EMV tag), an ISO 8583 bit number.

5.2.1.4 Usage

The *Usage* entry describes for what purpose the data is used, in the various contexts of the messages where the data occurs

It could be seen as a more concrete definition of the data in the context of the protocol (i.e. a definition by extension)

Usage is based on the occurrences of the data in messages or data structures (cross reference entry).

The Usage description is limited to the most relevant information for the Retailer protocol, and never describes the process by actors behind a data element if the process has no impact on the protocol.

5.2.1.5 Type

The type defines the based syntax of the data.

Type of data elements might be a basic type or a built type. Basic type is one of the following predefined types:

- *Boolean*: a data element which could take the value of “true” or “false”.
- *TextString*: a sequence of printable characters (Unicode). The character sequence might have any length, including zero characters.
- *DigitString*: a sequence of decimal digit characters (value ‘0’ through ‘9’). The digit sequence might have any length, including zero characters.
- *Integer*: a signed integer. The value of the integer data is unbounded. Implementations must accept at least 32 bits signed integers.
- *Decimal*: a decimal number (i.e. value ‘0’ through ‘9’), before and after the decimal point (character ‘.’). The decimal point and the number after it are optional. It cannot be prefixed with either a plus sign or a minus sign. Exponential and scientific notation are not supported.
- *ByteSequence*: a sequence of binary values without any interpretation. The binary sequence might have any length, including zero characters.
- *Enumeration*: a set of alternative values where each discrete value has a specific meaning. These values (or meanings) are identified by a label, and the list of possible values defines the type.
For instance the data element PeriodUnit, which identifies the period between consecutive payments, has the basic type Enumeration. The set of alternative values defining the data type is: “Day”, “Month” and “Year”. So PeriodUnit data element might take one of these three values.
- *Cluster*: like an Enumeration type, a Cluster type is defined by a set of alternative values identified by labels. But the value of a Cluster data element is a subset or an unordered list of the values defining the Cluster type. The subset could have any number of elements, including zero elements or all the elements of the set defining the Cluster.
For instance the data element POIOnLineCapab, which gives the on-line / off-line capabilities of the POI, has the basic type Cluster. The set of alternative values defining the data type is: “OffLine”, “OnLine” and “SemiOffLine”. So POIOnLineCapab data element might be any list among these three values (e.g. “SemiOffLine”, “OnLine OffLine”, “...”).

A built type is type based on basic type with some constraints. For instance ISOLanguage2A, which is the identification of a language according to the ISO 639-1 codes using two alpha characters, is built on the basic type TextString, with a length of 2 characters according to ISO 639-1.

Type of data structure is either:

- An “anonymous type” defined by the list of data element the structure contains. These anonymous types are defined in the Data Dictionary using the name of the field (for instance

in the PaymentRequest structure, the sub-structure SaleData is defined in the Data Dictionary by SaleData).

- An explicitly and externally defined type, which can be shared by several data structure of the messages. In this case, the explicit data structure type is defined in the section 5.2.3 *Data Types*, and the data structure in the section 5.2.2 *Data Elements and Structures* has a link to this type.

5.2.1.6 Format

Format refines the type of data, with some limits on the format.

Format definition could be:

- Range of the data length (notation: {min,max}, max could be absent for unlimited length as {min,}).
- Range of values (notation: [min-max], min and max could be absent for unlimited values, but not at the same time).
- A default value (e.g. "default 2")²⁸,
- A regular expression conforming to the XML/Schema definition,
- A semantic constraint (e.g. the format of a date as MMYY).

5.2.1.7 CrossRef

Cross reference entry lists all the locations in the messages where the data could be present. Cross references are in alphabetic order.

5.2.1.8 XMLCoding

XMLCoding contains the information specific to the XML/Schema data coding. Two values could be found:

- *Attribute*: when the data element is an XML attribute of the enclosing structure, if and only if the multiplicity is [1..1] or [0..1], i.e. if the data element is not repeated.
- *Enclosing*: When the data element is the only XML element of the enclosing data structure, the value of the data element is encoded as the XML value of the enclosing data structure. For instance *TrackData* is composed of 2 XML attributes *TrackNumb* and *TrackFormat* and one XML element *TrackValue* which is declared as "XML Enclosing" in the data dictionary. *TrackData* is then optimised, *TrackValue* is no more an XML element but the XML content of *TrackData*.

<TrackData TrackNum="2">;4970100202013617=11059019580970000000?</TrackData>

5.2.1.9 ASN1Coding

ASN1Coding contains the information specific to ASN.1/DER data coding. It contains the value of the ASN.1 tag of the class APPLICATION.

²⁸ The meaning of a default value is the same as XML/Schema or ASN.1 default values. For instance "default 2" means that if the data element is absent, it has the value "2", and has the same result than send the data element with the value "2".

5.2.1.10 Enumeration or Cluster Labels

For Enumeration type or Cluster type, the set of alternatives values has to be listed to define completely the type Enumeration or Cluster.

Name	GenericProfile
Definition	Functional profile of the Sale to POI protocol.
References	
Usage	Sent in the Login Request to identify the messages that might be requested or received by the Sale Terminal during the session. Sent in the Login Response to identify the messages that might be processed or sent by the POI Terminal during the session.
Type	Enumeration
Format	
CrossRef	LoginRequest.SaleSystemData.SaleTerminalData.SaleProfile LoginResponse.POISystemData.POITerminalData.POIPProfile
XMLCoding	Attribute
ASN1Coding	103

Values of the enumeration (or cluster)

Label	Description	Code
Basic	Protocol services that needs to be implemented by all	
Standard	Protocol services involving interaction between Sale devices shared between the two Systems.	
Extended	Complete Protocol services	

Figure 255: Enumeration or Cluster Labels

As presented in the figure above, a table containing all the possible values follows the data element definition (for an anonymous type), or the type definition (for an explicit defined type). For each value of the set, this table contains:

- The Label of the value,
- The Description or meaning of the value,
- Possibly a numeric code associated with the value, for compatibility reasons with an existing coding of values.

5.2.1.11 Data Structure Definition

For the definition of a data structure, the list of data components has to be listed to completely define the data structure.

The diagram shows a table for defining a data structure named 'HostStatus'. Below the table, an arrow points from the text 'Components of the data structure' to the first column of the table.

Name	HostStatus
Definition	State of a Host.
References	
Usage	Indicate the reachability of the host by the POI Terminal.
Type	defined data structure
Format	
CrossRef	DiagnosisResponse
XMLCoding	
ASN1Coding	108

Component	Mult.	Constraint	Rule
AcquirerID	[1..1]		
isReachable	[0..1]	default True	

Figure 256: Data Structure Definition

As presented in the figure above, a table containing all the components follows the data structure definition (for an anonymous type), or the type definition (for an explicit defined type). For each component, this table contains:

- The Name of the data, with a link to the definition of this data which could be a data element or another data structure,
- The Multiplicity of the component inside the data structure, in the same the message layout did,
- The specific constraints to this data for this data structure,
- The condition of presence of the data component.

5.2.2 Data Elements and Structures

5.2.2.1 A

Name	AbortReason
Definition	Reason of aborting a transaction
References	
Usage	Free text to log for further examination.
Type	TextString
Format	
CrossRef	AbortRequest
XMLCoding	
ASN1Coding	12

Name	AbortRequest
Definition	Body of the Abort Request message.
References	
Usage	It conveys Information requested for identification of the message request carrying the transaction to abort. A message to display on the CustomerError Device could be sent by the Sale System (DisplayOutput).
Type	defined data structure
Format	
CrossRef	AbortRequest
XMLCoding	
ASN1Coding	13

Component	Mult.	Constraint	Rule
MessageReference	[1..1]		
AbortReason	[1..1]		
DisplayOutput	[0..1]		To display an abort message to the Customer

Name	AccessedBy
Definition	Identification of an entity accessing data to perform an operation.
References	
Usage	Allow the synchronisation of customer order processing, when several parts of the Sale System access a customer order at the same time. The Sale entity active on the customer order is identified by this data.
Type	TextString
Format	
CrossRef	CardAcquisitionResponse.CustomerOrder PaymentRequest.PaymentData.CustomerOrder PaymentResponse.PaymentResult.CustomerOrder ReversalResponse.CustomerOrder
XMLCoding	Attribute
ASN1Coding	421

Name	AccountNumber
Definition	Identification of the customer account.
References	
Usage	
Type	TextString
Format	
CrossRef	CardAcquisitionResponse.PaymentInstrumentData.CheckData PaymentRequest.PaymentData.PaymentInstrumentData.CheckData PaymentResponse.PaymentResult.PaymentInstrumentData.CheckData
XMLCoding	
ASN1Coding	338

Name	AccountType
Definition	Type of cardholder account used for the transaction
References	
Usage	Allows a cardholder to select the type of account used for the transaction
Type	Enumeration
Format	
CrossRef	BalanceInquiryRequest.PaymentData
XMLCoding	Attribute
ASN1Coding	14

Label	Description	Code
Default	Default account	
Savings	Savings account	
Checking	Checking account	
CreditCard	Credit card account	
Universal	Universal account	
Investment	Investment account	
CardTotals	Card totals	
EpurseCard	e-Purse card account	

Name	AcquirerID
Definition	Identification of the Acquirer
References	ISO 8583 - Elemt. 32 - acquiring institution identification code
Usage	Identifications of the Acquirer when the POI System is multi-acquirer.
Type	DigitString
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentAcquirerData DiagnosisRequest DiagnosisResponse.HostStatus GetTotalsResponse.Totals.TransactionTotals PaymentRequest.TransactionData.TransactionConditions PaymentResponse.PaymentResult.PaymentAcquirerData ReconciliationRequest ReconciliationResponse.Totals.TransactionTotals ReversalRequest.OriginalPOITransaction
XMLCoding	Attribute
ASN1Coding	17

Name	AcquirerPOIID
<i>Definition</i>	Identification of the POI for the payment Acquirer
<i>References</i>	
<i>Usage</i>	Identification of the POI System or POI Terminal for the Acquirer.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentAcquirerData PaymentResponse.PaymentResult.PaymentAcquirerData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	15

Name	AcquirerTransactionID
<i>Definition</i>	Identification of the Transaction for the Acquirer.
<i>References</i>	
<i>Usage</i>	To identify the payment transaction on the Sale System.
<i>Type</i>	TransactionIdentificationType
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentAcquirerData PaymentResponse.PaymentResult.PaymentAcquirerData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	16

Name	AdditionalInformation
Definition	Unqualified information.
References	
Usage	
Type	TextString
Format	
CrossRef	CardAcquisitionResponse.CustomerOrder PaymentRequest.PaymentData.CustomerOrder PaymentResponse.PaymentResult.CustomerOrder ReversalResponse.CustomerOrder
XMLCoding	
ASN1Coding	422

Name	AdditionalInput
Definition	Additional information required to verify the PIN like part of the PAN, or driver ID.
References	
Usage	e.g. the PAN for the PIN Block ISO 1
Type	TextString
Format	
CrossRef	PINRequest.CardholderPIN PINResponse.CardholderPIN
XMLCoding	Attribute
ASN1Coding	18

Name	AdditionalProductInfo
Definition	Additional information related to the line item.
References	
Usage	
Type	TextString
Format	
CrossRef	PaymentRequest.TransactionData.SaleItem LoyaltyRequest.TransactionData.SaleItem StoredValueRequest.StoredValueData.SaleItemRebate
XMLCoding	
ASN1Coding	19

Name	AdditionalResponse
<i>Definition</i>	Additional information related to processing status of a message request .
<i>References</i>	
<i>Usage</i>	For logging purpose, in order to allow further analysis, statistics and deferred processing on the success or the failure of the request processing. This data element could be send when the message request has been processed successfully but with some warning in order to accept the processing of the message on the field but revealing a strange behaviour during the system integration or testing phase. See the specifications to have some frames of AdditionalResponse contents depending on the value of ErrorCondition.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse CardAcquisitionResponse.Response DiagnosisResponse.Response DisplayResponse.OutputResult.Response EnableServiceResponse.Response GetTotalsResponse InputResponse.InputResult.Response InputResponse.OutputResult.Response LoginResponse.Response LoginResponse.Response LoginResponse.Response PaymentResponse.Response PINResponse.Response PrintResponse.OutputResult.Response ReconciliationResponse.Response CardReaderInitResponse.Response CardReaderAPDUResponse.Response CardReaderPowerOffResponse.Response ReversalResponse.Response SoundResponse.Response StoredValueResponse TransactionStatusResponse.Response TransmitResponse.Response
<i>XMLCoding</i>	
<i>ASN1Coding</i>	20

Name	AdminRequest
Definition	Content of the Custom Admin Request message.
References	
Usage	Empty
Type	defined data structure
Format	
CrossRef	AdminRequest
XMLCoding	
ASN1Coding	314

Component	Mult.	Constraint	Rule
Servicelidentification	[0..1]		

Name	AdminResponse
Definition	Content of the Custom Admin Response message.
References	
Usage	It conveys the result of the Custom Admin.
Type	defined data structure
Format	
CrossRef	AdminResponse
XMLCoding	
ASN1Coding	315

Component	Mult.	Constraint	Rule
Response	[1..1]		

Name	Alignment
<i>Definition</i>	Alignment of the text string on the display line or print line
<i>References</i>	
<i>Usage</i>	Absence of this data element means the characters have a normal alignment.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputElement.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	21

<i>Label</i>	<i>Description</i>	<i>Code</i>
Left		
Right		
Centred		
Justified		

Name	AllowedLoyaltyBrand
<i>Definition</i>	Loyalty brands or programs allowed by the Sale System for the loyalty transaction.
<i>References</i>	
<i>Usage</i>	Restrict the loyalty to some predefined loyalty programs.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionRequest.TransactionData PaymentRequest.TransactionData.TransactionConditions
<i>XMLCoding</i>	
<i>ASN1Coding</i>	22

Name	AllowedPaymentBrand
<i>Definition</i>	Card payment brands allowed by the Sale System for the payment transaction.
<i>References</i>	
<i>Usage</i>	Restrict the payment to some predefined payment cards.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionRequest.TransactionData PaymentRequest.TransactionData.TransactionConditions
<i>XMLCoding</i>	
<i>ASN1Coding</i>	23

Name	AllowedProduct
Definition	Product that are payable by the payment card.
References	
Usage	Restriction of product payable by a card.
Type	defined data structure
Format	
CrossRef	CardAcquisitionResponse.PaymentInstrumentData.CardData PaymentResponseResponse.PaymentResult.PaymentInstrumentData.CardData
XMLCoding	
ASN1Coding	394

Component	Mult.	Constraint	Rule
ProductCode	[1..1]		
EanUpc	[0..1]		
ProductLabel	[0..1]		
AdditionalProductInfo	[0..1]		

Name	AllowedProductCode
Definition	Product codes that are payable by the payment card.
References	
Usage	If not all the products are accepted for the payment card proposed by the Customer. In this case, Result is failure, ErrorCondition is Payment Restriction. For One Time Reservation, the POI can send product codes payable by the payment, even if the POI did not send SaleItem in the request. If at least one product sent in the request is accepted, the Result is Success.
Type	DigitString
Format	{1,20}
CrossRef	CardAcquisitionResponse.PaymentInstrumentData.CardData PaymentResponseResponse.PaymentResult.PaymentInstrumentData.CardData
XMLCoding	
ASN1Coding	24

Name	AmountsReq
Definition	Various amounts related to the payment and loyalty request from the Sale System.
References	
Usage	Amounts requested by the Sale System for the payment and loyalty transaction, containing: The currency which is the same for all these amounts The requested amount to pay The cash back part of the requested amount for a payment with cash back The already amount paid for the Sale transaction. This is a split payment, where the Sale Items (the basket) if present is not split. The tip part of the requested amount for a payment with tip
Type	defined data structure
Format	
CrossRef	PaymentRequest.TransactionData
XMLCoding	
ASN1Coding	26

Component	Mult.	Constraint	Rule
Currency	[1..1]		
RequestedAmount	[0..1]		Absent if the maximum amount is unknown for a OneTimeReservation The value has to be greater than 0.
CashBackAmount	[0..1]		if payment with cash back. This data has to be performed by the POI. If cash back is not allowed, the payment is refused Failure-PaymentRestriction.
TipAmount	[0..1]		If payment with tip requested by the Sale System.
PaidAmount	[0..1]		If SplitPaymentFlag is True (split amount of a split payment).
MinimumAmountToDeliver	[0..1]		if unknown maximum amount for a OneTimeReservation or minimum amount requested by the Sale System
MaximumCashBackAmount	[0..1]		Maximum amount which could be requested for cash-back to the Sale System.
MinimumSplitAmount	[0..1]		Minimum amount of a split, which could be requested.

Name	AmountsResp
<i>Definition</i>	Various amounts related to the payment response from the POI System.
<i>References</i>	
<i>Usage</i>	Amounts approved by the POI and the Acquirer for the payment and loyalty transaction, containing: The authorised amount to be paid The amount of the rebates The amount of financial fees The cash back part of the requested amount for a payment with cash back The tip part of the requested amount for a payment with tip
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult
<i>XMLCoding</i>	
<i>ASN1Coding</i>	27

Component	Mult.	Constraint	Rule
Currency	[0..1]		Mandatory for currency conversion.
AuthorizedAmount	[1..1]		
TotalRebatesAmount	[0..1]		
TotalFeesAmount	[0..1]		
CashBackAmount	[0..1]		if payment with cash back
TipAmount	[0..1]		If payment with tip requested by the Sale System.

Name	AmountValue
<i>Definition</i>	Value of an amount.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.LoyaltyData.LoyaltyAmount PaymentResponse.LoyaltyResult.LoyaltyAmount LoyaltyRequest.LoyaltyData.LoyaltyAmount LoyaltyResponse.LoyaltyResult.LoyaltyAmount ReversalRequest.OriginalPOITransaction
<i>XMLCoding</i>	Enclosing
<i>ASN1Coding</i>	28

Name	APDUClass
Definition	Class field of the APDU command (CLA)
References	ISO 7816-4
Usage	APDU request for Card Reader device request. For specific card like synchronous card, a private value should be used in accordance to ISO 7816-4 (private range D0-FE)
Type	ByteSequence
Format	{1,1}
CrossRef	CardReaderAPDUREquest
XMLCoding	Attribute
ASN1Coding	29

Name	APDUData
Definition	Data field of the APDU command (Lc + Data)
References	ISO 7816-4
Usage	APDU request for Card Reader device request. APDU response for Card Reader device response.
Type	ByteSequence
Format	
CrossRef	CardReaderAPDUREquest CardReaderAPDUREsponse
XMLCoding	
ASN1Coding	30

Name	APDUExpectedLength
Definition	Expected length of the data field of the APDU response to the command (Le)
References	ISO 7816-4
Usage	APDU request for Card Reader device request.
Type	ByteSequence
Format	{1,1}
CrossRef	CardReaderAPDUREquest
XMLCoding	Attribute
ASN1Coding	31

Name	APDUIInstruction
Definition	Instruction field of the APDU command (INS)
References	ISO 7816-4
Usage	APDU request for Card Reader device request.
Type	ByteSequence
Format	{1,1}
CrossRef	CardReaderAPDUREquest
XMLCoding	Attribute
ASN1Coding	32

Name	APDUPar1
Definition	Parameter 1 field of the APDU command (P1)
References	ISO 7816-4
Usage	APDU request for Card Reader device request.
Type	ByteSequence
Format	{1,1}
CrossRef	CardReaderAPDUREquest
XMLCoding	Attribute
ASN1Coding	33

Name	APDUPar2
Definition	Parameter 2 field of the APDU command(P2)
References	ISO 7816-4
Usage	APDU request for Card Reader device request.
Type	ByteSequence
Format	{1,1}
CrossRef	CardReaderAPDUREquest
XMLCoding	Attribute
ASN1Coding	34

Name	ApplicationName
Definition	Name of the software product.
References	
Usage	Sent in the Login Request (resp. Response) to identify the Sale System (resp. POI System) product name during the session.
Type	TextString
Format	
CrossRef	LoginRequest.SaleSoftware LoginResponse.POISystemData.POISoftware
XMLCoding	Attribute
ASN1Coding	219

Name	ApprovalCode
<i>Definition</i>	Code assigned to a transaction approval by the Acquirer.
<i>References</i>	ISO 8583 - Elemt. 38 - approval code
<i>Usage</i>	Could be an identifier of the approved transaction for the Acquirer. This data element is conditional for the Loyalty Acquirers. Used in the PaymentRequest request for a referral
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentAcquirerData LoyaltyResponse.LoyaltyResult.LoyaltyAcquirerData PaymentResponse.LoyaltyResult.LoyaltyAcquirerData PaymentResponse.PaymentResult.PaymentAcquirerData PaymentRequest.PaymentTransaction.OriginalPOITransaction ReversalRequest.OriginalPOITransaction
<i>XMLCoding</i>	
<i>ASN1Coding</i>	35

Name	AreaSize
<i>Definition</i>	Size of an area
<i>References</i>	
<i>Usage</i>	Contain the size of the pad area where the signature is written, given with the maximum abissa and ordinate values (X and Y). The maximum value is "FFFF".
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult.CapturedSignature
<i>XMLCoding</i>	
<i>ASN1Coding</i>	343

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
X	[1..1]		
Y	[1..1]		

Name	ATRValue
<i>Definition</i>	Value of the Answer To Reset of a chip card
<i>References</i>	ISO 7816-3
<i>Usage</i>	Card reader device response
<i>Type</i>	ByteSequence
<i>Format</i>	{1,100}
<i>CrossRef</i>	CardReaderInitResponse.ICCResetData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	36

Name	AuthenticationMethod
<i>Definition</i>	Method for customer authentication.
<i>References</i>	
<i>Usage</i>	Allows the Sale System informed about customer authentication for the payment transaction.
<i>Type</i>	Cluster
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	347

<i>Label</i>	<i>Description</i>	<i>Code</i>
Bypass	Authentication bypassed by the merchant.	
ManualVerification	Manual verification, for example passport or drivers license.	
MerchantAuthentication	Merchant-related authentication.	
OfflinePIN	Off-line PIN authentication (Personal Identification Number).	
OnLinePIN	On-line PIN authentication (Personal Identification Number).	
PaperSignature	Handwritten paper signature.	
SecuredChannel	Channel-encrypted transaction.	
SecureCertificate	Secure electronic transaction with cardholder X.509 certificate.	
SecureNoCertificate	Secure electronic transaction without cardholder certificate.	
SignatureCapture	Electronic signature capture (handwritten signature).	
UnknownMethod	Authentication method is performed unknown.	

Name	AuthorizedAmount
<i>Definition</i>	The amount authorized by the Acquirer for the payment transaction.
<i>References</i>	
<i>Usage</i>	AuthorizedAmount include CashBackAmount na d TipAmount. Payment response from the payment Acquirer. AuthorizedAmount = request.RequestedAmount - TotalRebatesAmount + TotalFeesAmount - (PartialDeclinedAmount)
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult.AmountsReq
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	37

5.2.2.2 B-C

Name	BalanceInquiryRequest
Definition	Content of the Balance Inquiry Request message.
References	
Usage	It conveys Information related to the account for which a Balance Inquiry is requested
Type	defined data structure
Format	
CrossRef	BalanceInquiryRequest
XMLCoding	
ASN1Coding	39

Component	Mult.	Constraint	Rule
PaymentAccountReq	[0..1]		
LoyaltyAccountReq	[0..1]		

Name	BalanceInquiryResponse
Definition	Content of the Balance Inquiry Response message.
References	
Usage	It conveys the balance and the identification of the associated payment, loyalty or stored value account.
Type	defined data structure
Format	
CrossRef	BalanceInquiryResponse
XMLCoding	
ASN1Coding	40

Component	Mult.	Constraint	Rule
Response	[1..1]		
PaymentAccountStatus	[0..1]		If BalanceInquiryRequest. PaymentAccount present
LoyaltyAccountStatus	[0..1]		If BalanceInquiryRequest. LoyaltyData present

Name	BankID
<i>Definition</i>	Identification of the bank.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.PaymentInstrumentData.CheckData PaymentRequest.PaymentData.PaymentInstrumentData.CheckData PaymentResponse.PaymentResult.PaymentInstrumentData.CheckData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	337

Name	BarcodeType
<i>Definition</i>	Type of BarCode coding.
<i>References</i>	
<i>Usage</i>	Qualification of the barcode value to display or print
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.OutputBarCode EnableServiceRequest.Output.OutputElement.OutputBarCode InputRequest.Output.OutputElement.OutputBarCode PrintRequest.Output.OutputElement.OutputBarCode CardReaderInitRequest.Output.OutputElement.OutputBarCode CardReaderAPDUREquest.Output.OutputElement.OutputBarCode CardReaderPowerOffRequest.Output.OutputElement.OutputBarCode
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	41

<i>Label</i>	<i>Description</i>	<i>Code</i>
EAN8		
EAN13		
UPCA		
Code25		
Code128		
PDF417		
QRCode	2D Barcode. Only Numeric and Alphanumeric supported. Version, Encoding mode and Error correction mode are selected by the printer driver.	

Name	BarcodeValue
Definition	Value with a BarCode coding.
References	
Usage	The barcode value to display or print
Type	TextString
Format	
CrossRef	DisplayRequest.Output.OutputElement.OutputBarCode EnableServiceRequest.Output.OutputElement.OutputBarCode InputRequest.Output.OutputElement.OutputBarCode PrintRequest.Output.OutputElement.OutputBarCode CardReaderInitRequest.Output.OutputElement.OutputBarCode CardReaderAPDUREquest.Output.OutputElement.OutputBarCode CardReaderPowerOffRequest.Output.OutputElement.OutputBarCode
XMLCoding	Enclosing
ASN1Coding	42

Name	BatchRequest
<i>Definition</i>	Content of the Batch Request message.
<i>References</i>	
<i>Usage</i>	Message to send Payment, Loyalty and Reversal transactions to be performed without the Sale System, or to request the Payment, Loyalty and Reversal transactions that has been performed without the Sale System.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BatchRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	378

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
RemoveAllFlag	[0..1]	default False	
TransactionToPerform	[0..n]		

Name	BatchResponse
<i>Definition</i>	Content of the Batch Response message.
<i>References</i>	
<i>Usage</i>	It conveys either the response of transactions to perform sent in the batch request, or a collection of results of performed Payment, Loyalty and Reversal transactions in a batch message or file.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BatchResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	380

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
PerformedTransaction	[0..n]		

Name	BeepKeyFlag
Definition	Indicates, when the user press a key, if a beep has to be generated (value True).
References	
Usage	During the processing of an Input command.
Type	Boolean
Format	
CrossRef	InputRequest.InpuData PINRequest
XMLCoding	Attribute
ASN1Coding	389

Name	CapturedSignature
Definition	Numeric value of a handwritten signature.
References	
Usage	Contain the value of a handwritten signature, e.g. the signature of a cardholder on the merchant payment receipt. Only one format of the signature is allowed, it contains: The size of the pad area where the signature is written, given with the maximum abscissa and ordinate values. The sequence of coordinates where the pen changes direction or lift.
Type	defined data structure
Format	
CrossRef	PaymentResponse.PaymentResult
XMLCoding	
ASN1Coding	342

Component	Mult.	Constraint	Rule
AreaSize	[0..1]		
SignaturePoint	[1..n]		

Name	CardAcquisitionReference
<i>Definition</i>	Reference to the last CardAcquisition, to use the same card.
<i>References</i>	
<i>Usage</i>	Avoid re-entering the same card.
<i>Type</i>	TransactionIdentificationType
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest.LoyaltyData BalanceInquiryRequest.PaymentData LoyaltyRequest.LoyaltyData PaymentRequest.PaymentData PaymentRequest.LoyaltyData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	44

Name	CardAcquisitionRequest
<i>Definition</i>	Content of the Card Acquisition Request message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the payment and loyalty cards to read and analyse. This message pair is usually followed by a message pair (e.g. payment or loyalty) which refers to this Card Acquisition message pair. The Card Acquisition message pair groups all the cards needed by the message pair to follow. In case of EMV, the transaction does not start (GPO) in the Card Acquisition message pair.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	45

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
SaleData	[1..1]		
CardAcquisitionTransaction	[1..1]		

Name	CardAcquisitionResponse
<i>Definition</i>	Content of the Card Acquisition Response message.
<i>References</i>	
<i>Usage</i>	<p>It conveys Information related to the payment and loyalty cards read and processed by the POI System and entered by the Customer:</p> <ul style="list-style-type: none"> As for the Payment request, the result of the CardAcquisition and the identification of the transaction. One payment card, the one chosen by the Customer, with one payment brand, or several, if the card includes several brands, and the customer has not chosen one. As usual, there is no payment brand if the result is not successful. Zero, one or more loyalty brands The card entry modes of these card entered by the Customer in the order of the PaymentBrand and LoyaltyBrand The card data of the payment card, if this is required for this type of payment card
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	46

Component	Mult.	Constraint	Rule
Response	[1..1]		
SaleData	[1..1]		
POIData	[1..1]		
PaymentBrand	[0..n]		Brands available for payment by the card and not chosen by the Customer
PaymentInstrumentData	[0..1]		If this type of payment card is configured to send information if the CardAcquisition response
LoyaltyAccount	[0..n]		If loyalty card selected by the customer
CustomerOrder	[0..n]		If a customer orders has to be created.

Name	CardAcquisitionTransaction
Definition	Data related to the payment and loyalty card acquisition.
References	
Usage	Elements requested by the Sale System that are global to the payment or loyalty transaction.
Type	defined data structure
Format	
CrossRef	CardAcquisitionRequest
XMLCoding	
ASN1Coding	47

Component	Mult.	Constraint	Rule
AllowedPaymentBrand	[0..n]		
AllowedLoyaltyBrand	[0..n]		
LoyaltyHandling	[0..1]	default Allowed	
CustomerLanguage	[0..1]		If the language is selected by the Sale System before the request to the POI.
ForceEntryMode	[0..n]		
ForceCustomerSelectionFlag	[0..1]	default False	
TotalAmount	[0..1]		Mandatory for contactless card, otherwise absent
PaymentType	[0..1]		Mandatory for contactless card, otherwise absent
CashBackFlag	[0..1]		For contactless, True if cash back has been requested, default False. Otherwise absent.

Name	CardBrand
Definition	Type of payment or loyalty card
References	
Usage	Indicates the type of card in the ReconciliationResponse. See <i>PaymentBrand</i> and <i>LoyaltyBrand</i> .
Type	TextString
Format	
CrossRef	GetTotalsResponse.Totals.TransactionTotals ReconciliationResponse.Totals.TransactionTotals
XMLCoding	Attribute
ASN1Coding	48

Name	CardCountryCode
Definition	Country Code attached to the card (3 numerics).
References	ISO 3166-1
Usage	Determine local or international transaction. This data is present if provided by the card used for the transaction.
Type	DigitString
Format	{3,3}
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData BalanceInquiryRequest.PaymentData .PaymentInstrumentData.CardData CardAcquisitionResponse.PaymentInstrumentData.CardData PaymentRequest.PaymentData .PaymentInstrumentData.CardData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData
XMLCoding	Attribute
ASN1Coding	319

Name	CardData
Definition	Information related to the payment card used for the transaction.
References	
Usage	<p>Allows acquisition of the card data by the Sale System before the Payment, CardAcquisition or BalanceInquiry request to the POI.</p> <p>It could also be sent in the CardAcquisition response, to be processed by the Sale System. In this case, the card or type of card has to be configured to have this behaviour. It is then expected that for this card the information sent in response to the payment are the same for the CardAcquisition response.</p> <p>Data that could be protected in the response are grouped in the data structure SensitiveCardData. In BalanceInquiryResponse and CardAcquisitionResponse, AllowedProduct may be provided for card with restrictions on products.</p>
Type	defined data structure
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData BalanceInquiryRequest.PaymentData.PaymentInstrumentData CardAcquisitionResponse.PaymentInstrumentData PaymentRequest.PaymentData.PaymentInstrumentData PaymentResponse.PaymentResult.PaymentInstrumentData
XMLCoding	
ASN1Coding	49

Component	Mult.	Constraint	Rule
PaymentBrand	[0..1]		If card PAN is readable
MaskedPAN	[0..1]		If required for the card, instead of clear PAN
PaymentAccountRef	[0..1]		Mandatory if available.
EntryMode	[0..1]		Mandatory in the request
CardCountryCode	[0..1]		If available in the card
ProtectedCardData	[0..1]		SensitiveCardData protected by CMS EnvelopedData
SensitiveCardData	[0..1]		If structure non empty and unprotected
AllowedProductCode	[0..n]		If ErrorCondition is "PaymentRestriction", some products are not payable by the payment card (payment response).
AllowedProduct	[0..n]		If the card has restrictions on product that can be purchased (card acquisition or balance inquiry response).
PaymentToken	[0..1]		Present in If requested in CardAcquisitionResponse or PaymentResponse if requested in the request or in the Login request for the session. Otherwise absent.
CustomerOrder	[0..n]		If the list of customer orders has been requested.

Name	CardholderPIN
Definition	Encrypted PIN and related information
References	
Usage	To request PIN Verify to the POI, or to get the encrypted PIN block.
Type	defined data structure
Format	
CrossRef	PINRequest PINResponse
XMLCoding	
ASN1Coding	51

Component	Mult.	Constraint	Rule
EncrPINBlock	[1..1]		
PINFormat	[1..1]		
AdditionalInput	[0..1]		

Name	CardReaderAPDUREquest
<i>Definition</i>	Content of the Card Reader APDU Request message.
<i>References</i>	
<i>Usage</i>	It contains the APDU request to send to the chip of the card, and a possible invitation message to display on the CashierInterface or the CustomerInterface.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardReaderAPDUREquest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	53

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
APDUClass	[1..1]		
APDUIInstruction	[1..1]		
APDUPar1	[1..1]		
APDUPar2	[1..1]		
APDUData	[0..1]		
APDUExpectedLength	[0..1]		

Name	CardReaderAPDUREsponse
<i>Definition</i>	Content of the Card Reader APDU Response message.
<i>References</i>	
<i>Usage</i>	It contains the result of the requested service, APDU response sent by the chip of the card in response to the APDU request.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardReaderAPDUREsponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	54

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
APDUData	[0..1]		
CardStatusWords	[1..1]		

Name	CardReaderInitRequest
<i>Definition</i>	Content of the Card Reader Init Request message.
<i>References</i>	
<i>Usage</i>	It contains possible restrictions for the type of card reader to use, and a possible invitation message to display on the CashierInterface or the CustomerInterface.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardReaderInitRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	55

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
WarmResetFlag	[0..1]		
ForceEntryMode	[0..n]		
LeaveCardFlag	[0..1]	default True	
MaxWaitingTime	[0..1]		
DisplayOutput	[0..1]		

Name	CardReaderInitResponse
<i>Definition</i>	Content of the Card Reader Init Response message.
<i>References</i>	
<i>Usage</i>	It contains the result of the Card Reader by the POI Card Reader (magnetic stripe content, or chip initialisation data).
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardReaderInitResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	56

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
EntryMode	[0..1]		RFID, MagStripe, ICC, EMVContactless or SynchronousICC
TrackData	[0..4]		if EntryMode is RFID or MagStripe
ICCResetData	[0..1]		if EntryMode is ICC, EMVContactless or SynchronousICC

Name	CardReaderOKFlag
<i>Definition</i>	Indicates if the card readers are working and usable.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.POIStatus LoginResponse.POIStatus
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	57

Name	CardReaderPowerOffRequest
<i>Definition</i>	Content of the Card Reader Power-Off Request message.
<i>References</i>	
<i>Usage</i>	It contains a possible invitation message to display on the CashierInterface or the CustomerInterface, for removing the card.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardReaderPowerOffRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	58

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
MaxWaitingTime	[0..1]		
DisplayOutput	[0..1]		

Name	CardReaderPowerOffResponse
<i>Definition</i>	Content of the Card Reader Power-Off Response message.
<i>References</i>	
<i>Usage</i>	It contains the result of the processed command.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardReaderPowerOffResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	59

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		

Name	CardSeqNumb
<i>Definition</i>	Card Sequence Number
<i>References</i>	EMV - Tag 5F34 - Application Primary Account Number (PAN) Sequence Number
<i>Usage</i>	Identify a card inside a set of cards with the same PAN. This data is present if provided by the card used for the transaction.
<i>Type</i>	DigitString
<i>Format</i>	{2,3}
<i>CrossRef</i>	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData CardAcquisitionResponse.PaymentInstrumentData.CardData.SensitiveCardData PaymentRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	60

Name	CardStatusWords
<i>Definition</i>	Status of a smartcard response to a command (SW1-SW2)
<i>References</i>	ISO 7816-4
<i>Usage</i>	Card reader device response. For a synchronous card, there are only two values: 9000 in case of success and 6F00 in case of failure.
<i>Type</i>	ByteSequence
<i>Format</i>	{2,2}
<i>CrossRef</i>	CardReaderInitResponse.ICCResetData CardReaderAPDUResponse
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	61

Name	CashBackAmount
<i>Definition</i>	The cash-back part of the amount requested by the Sale for the payment.
<i>References</i>	
<i>Usage</i>	Payment with cash-back (forbidden for reservations). If present in the Payment request, this data has to be performed by the POI in coordination with the Customer. If present in the Payment response, a cashback has been performed.
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.Amounts PaymentResponse.PaymentResult.Amounts
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	62

Name	CashBackFlag
<i>Definition</i>	Cash back has been requested with the payment transaction.
<i>References</i>	
<i>Usage</i>	Allows choice of the Customer language when the POI displays messages or print text to Merchant interface.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionRequest.TransactionData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	411

Name	CashHandlingDevice
<i>Definition</i>	Status of cash handling device.
<i>References</i>	
<i>Usage</i>	Indicate the status and the remaining coins and bill in a cash handling device.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.POISatus LoginResponse.POISatus
<i>XMLCoding</i>	
<i>ASN1Coding</i>	362

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
CashHandlingOKFlag	[1..1]		
Currency	[1..1]		
CoinsOrBills	[1..n]		

Name	CashHandlingOKFlag
<i>Definition</i>	Indicates if the cash handling device is working and usable.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.POISatus.CashHandlingDevice LoginResponse.POISatus.CashHandlingDevice
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	363

Name	CertificationCode
Definition	Certification code of the software which manages the Sale to POI protocol.
References	
Usage	Sent in the Login Request (resp. Response) to get the certification code of the Sale System (resp. POI System) product software. This code could be a software checksum or any number associated to the software.
Type	TextString
Format	
CrossRef	LoginRequest.SaleSoftware LoginResponse.POISystemData.POISoftware
XMLCoding	Attribute
ASN1Coding	65

Name	CharacterHeight
Definition	Character height of the text string to display or print.
References	
Usage	Absence of this data element means the characters have a normal height.
Type	Enumeration
Format	
CrossRef	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputElement.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
XMLCoding	Attribute
ASN1Coding	66

Label	Description	Code
SingleHeight		
DoubleHeight		
HalfHeight		

Name	CharacterSet
<i>Definition</i>	The character encoding of the text string.
<i>References</i>	IANA character encoding
<i>Usage</i>	Used for ASN.1 message encoding only. For XML message encoding, character encoding of the XML document is used. The device (Sale or POI Terminal) provide several character sets which have to be selected dynamically.
<i>Type</i>	Integer
<i>Format</i>	[3-2000]
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	67

Name	CharacterStyle
<i>Definition</i>	Typographic style of the sequence of characters to display or print.
<i>References</i>	
<i>Usage</i>	Absence of this data element means the characters have a normal style.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputElement.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	68

Label	Description	Code
Normal		
Bold		
Italic		
Underlined		

Name	CharacterWidth
Definition	Character width of the text string to display or print.
References	
Usage	Absence of this data element means the characters have a normal width.
Type	Enumeration
Format	
CrossRef	DiplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputElement.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
XMLCoding	Attribute
ASN1Coding	69

Label	Description	Code
SingleWidth		
DoubleWidth		

Name	Charges
Definition	Charges related to a transaction.
References	
Usage	Charge related to the payment instalments.
Type	SimpleAmountType
Format	
CrossRef	PaymentRequest.PaymentData.Instalment
XMLCoding	Attribute
ASN1Coding	377

Name	CheckCardNumber
<i>Definition</i>	Check guarantee card number.
<i>References</i>	
<i>Usage</i>	The human readable number from the Check Guarantee Card that is presented during the check tendering process.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.PaymentInstrumentData.CheckData PaymentRequest.PaymentData.PaymentInstrumentData.CheckData PaymentResponse.PaymentResult.PaymentInstrumentData.CheckData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	340

Name	CheckCountry
<i>Definition</i>	Country of the bank check.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	ISOCountry3A
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.PaymentInstrumentData.CheckData PaymentRequest.PaymentData.PaymentInstrumentData.CheckData PaymentResponse.PaymentResult.PaymentInstrumentData.CheckData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	341

Name	CheckData
<i>Definition</i>	Information related to the paper check used for the transaction.
<i>References</i>	
<i>Usage</i>	Allows the check information to be provided by the Sale System before requesting the payment, or stored by the Sale System after processing of the payment.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.PaymentInstrumentData PaymentRequest.PaymentData.PaymentInstrumentData PaymentResponse.PaymentResult.PaymentInstrumentData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	70

Component	Mult.	Constraint	Rule
BankID	[0..1]		Mandatory if TrackData absent
AccountNumber	[0..1]		Mandatory if TrackData absent
CheckNumber	[0..1]		Mandatory if TrackData absent
TrackData	[0..1]		Mandatory if CheckNumber absent
CheckCardNumber	[0..1]		If provided by the customer
TypeCode	[0..1]	default Personal	
Country	[0..1]		Absent if country of the Sale system

Name	CheckNumber
<i>Definition</i>	Identification of the bank check.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.PaymentInstrumentData.CheckData PaymentRequest.PaymentData.PaymentInstrumentData.CheckData PaymentResponse.PaymentResult.PaymentInstrumentData.CheckData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	339

Name	CheckTypeCode
<i>Definition</i>	Type of bank check.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.PaymentInstrumentData.CheckData PaymentRequest.PaymentData.PaymentInstrumentData.CheckData PaymentResponse.PaymentResult.PaymentInstrumentData.CheckData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	336

<i>Label</i>	<i>Description</i>	<i>Code</i>
Personal		
Company		

Name	CoinsOrBills
<i>Definition</i>	Number of coins or bills of a given value.
<i>References</i>	
<i>Usage</i>	Indicates the remaining number of coins or bills of a given value in a cash handling device. When the cash handling machine does not have any more coins or bills of a certain value, the number must be equal to 0.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.POIStatus.CashHandlingDevice LoginResponse.POIStatus.CashHandlingDevice
<i>XMLCoding</i>	
<i>ASN1Coding</i>	346

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
UnitValue	[1..1]		
Number	[1..1]		

Name	Color
<i>Definition</i>	Color of the text string to display or print.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputElement.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	71

Label	Description	Code
White		
Black		
Red		
Green		
Blue		
Yellow		
Magenta		
Cyan		

Name	Commission
<i>Definition</i>	Commission for a service.
<i>References</i>	
<i>Usage</i>	Commission for a currency conversion.
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult.CurrencyConversion
<i>XMLCoding</i>	
<i>ASN1Coding</i>	414

Name	CommunicationOKFlag
<i>Definition</i>	Indicates if the communication infrastructure is working and usable.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.POIStatus LoginResponse.POIStatus
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	72

Name	ConfirmedFlag
<i>Definition</i>	Confirmation or not of what has been requested to the user in a GetConfirmation input command.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	InputResponse.InputResult.Input
<i>XMLCoding</i>	
<i>ASN1Coding</i>	73

Name	ConvertedAmount
<i>Definition</i>	Amount after a currency conversion.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	AmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult.CurrencyConversion
<i>XMLCoding</i>	
<i>ASN1Coding</i>	334

Name	CumulativeAmount
<i>Definition</i>	Sum of a collection of amounts.
<i>References</i>	
<i>Usage</i>	Total amount of the payment instalments.
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.PaymentData.Instalment
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	375

Name	Currency
Definition	Currency of a monetary amount.
References	
Usage	
Type	ISOCurrency3A
Format	
CrossRef	BalanceInquiryResponse.PaymentResult BalanceInquiryResponse.LoyaltyResult CardAcquisitionResponse.CustomerOrder DiagnosisResponse.POISatus.CashHandlingDevice GetTotalsResponse.Totals.TransactionTotals.PaymentTotals GetTotalsResponse.Totals.TransactionTotals.LoyaltyTotals PaymentRequest.PaymentData.CustomerOrder PaymentRequest.TransactionData.Amounts PaymentRequest.LoyaltyData.LoyaltyAmount PaymentResponse.LoyaltyResult.LoyaltyAmount PaymentResponse.PaymentResult.PaymentInstrumentData.CardData.CustomerOrder LoginResponse.POISatus.CashHandlingDevice LoyaltyRequest.LoyaltyData.LoyaltyAmount LoyaltyRequest.TransactionData LoyaltyResponse.LoyaltyResult.LoyaltyAmount ReconciliationResponse.Totals.TransactionTotals.PaymentTotals ReconciliationResponse.Totals.TransactionTotals.LoyaltyTotals ReversalResponse.CustomerOrder
XMLCoding	Attribute
ASN1Coding	74

Name	CurrencyConversion
Definition	Information related to a currency conversion
References	
Usage	A currency conversion occurred in the payment, and the merchant needs to know information related to this conversion (e.g. to print on the sale receipt)
Type	defined data structure
Format	
CrossRef	PaymentResponse.PaymentResult
XMLCoding	
ASN1Coding	333

Component	Mult.	Constraint	Rule
CustomerApprovedFlag	[0..1]	Default True	
ConvertedAmount	[1..1]		
Rate	[0..1]		Conversion rate of the target currency against the source currency.
Markup	[0..1]		Markup of the conversion.
Commission	[0..1]		Commission of the conversion.
Declaration	[0..1]		If a declaration has to be presented to the customer

Name	CurrentAmount
<i>Definition</i>	total amount of all completed transactions of a customer order.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.CustomerOrder PaymentRequest.PaymentData.CustomerOrder PaymentResponse.CustomerOrder ReversalRequest.CustomerOrder ReversalResponse.CustomerOrder
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	420

Name	CurrentBalance
<i>Definition</i>	Balance of an account.
<i>References</i>	
<i>Usage</i>	Account balance after processing of the transaction
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult BalanceInquiryResponse.LoyaltyResult BalanceInquiryResponse.StoredValueResult LoyaltyResponse.LoyaltyResult.LoyaltyAmount PaymentResponse.LoyaltyResult.LoyaltyAmount StoredValueResponse.StoredValueResult
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	38

Name	CustomerApprovedFlag
<i>Definition</i>	Notify if the customer has approved something.
<i>References</i>	
<i>Usage</i>	Indicates if the customer has accepted a currency conversion.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult.CurrencyConversion
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	412

Name	CustomerLanguage
<i>Definition</i>	Language of the Customer
<i>References</i>	
<i>Usage</i>	Allows choice of the Customer language when the POI displays messages or print text to Merchant interface.

Usually this is the role of the POI System to select the Customer language, but the Customer language could be selected by the Sale System before the POI intervention for some context (e.g. ticketing).

When a language is selected by the customer outside the processing of a request, or when the request cannot be aborted, e.g. during an Input processing, an event may sent.

<i>Type</i>	ISOLanguage2A
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionRequest.TransactionData EventNotification LoyaltyRequest.TransactionData PaymentRequest.TransactionData.TransactionConditions PaymentResponse.PaymentResult ReversalRequest.OriginalPOITransaction StoredValueRequest.TransactionData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	75

Name	CustomerOrder
<i>Definition</i>	Customer order attached to a customer, recorded in the POI system.
<i>References</i>	
<i>Usage</i>	Allows the management of customer orders by the POI, for instance in a multi-channel or a click and collect sale transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse PaymentRequest.PaymentData PaymentResponse ReversalRequest ReversalResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	415

Component	Mult.	Constraint	Rule
CustomerOrderID	[0..1]		
SaleReferenceId	[1..1]		
OpenOrderState	[0..1]	default True	
StartDate	[1..1]		
EndDate	[0..1]		If OpenOrderState = "False"
ForecastedAmount	[1..1]		
CurrentAmount	[1..1]		
Currency	[0..1]		
AccessedBy	[0..1]		If multiple currencies are allowed.
AdditionalInformation	[0..1]		If order process in progress.

Name	CustomerOrderID
<i>Definition</i>	Additional and optional identification of a customer order.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.CustomerOrder PaymentRequest.PaymentData.CustomerOrder PaymentResponse.PaymentResult.PaymentInstrumentData.CardData.CustomerOrder ReversalRequest.CustomerOrder ReversalResponse.CustomerOrder
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	416

Name	CustomerOrderReq
Definition	List of customer orders must be sent in response message.
References	
Usage	List of customer order open, closed or both to be sent in the response messages.
Type	Cluster
Format	
CrossRef	CardAcquisitionRequest.SaleData LoginRequest PaymentRequest.SaleData Reversal.SaleData
XMLCoding	Attribute
ASN1Coding	423

Label	Description	Code
Open	Customer order not completed.	
Closed	Completed customer orders.	
Both	All type of CustomerOrder should be listed	

5.2.2.3 D

Name	DateTime
Definition	Date and Time
References	
Usage	In the Login request message, the Sale System gives its date and time to the POI System. In the Login response, the POI System gives its date and time to the Sale System.
Type	ISODateTime
Format	
CrossRef	LoginRequest LoginResponse.POISystemData
XMLCoding	
ASN1Coding	76

Name	DebitPreferredFlag
Definition	The preferred type of payment is a debit transaction rather a credit transaction.
References	
Usage	Allows the Sale system to request a debit payment transaction rather a credit payment transaction if the selected card payment can accept a debit transaction.
Type	Boolean
Format	
CrossRef	PaymentRequest.TransactionData.TransactionConditions
XMLCoding	Attribute
ASN1Coding	348

Name	Declaration
<i>Definition</i>	Declaration to present to the customer or the cashier for validation.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult.CurrencyConversion
<i>XMLCoding</i>	
<i>ASN1Coding</i>	335

Name	DefaultInputString
<i>Definition</i>	Default string value for an input command.
<i>References</i>	
<i>Usage</i>	On the TextString, DigitString and DecimalString input commands: default string displayed on the input field before entering the string. On GetConfirmation input command: "Y" for yes, "N" for no.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	InputRequest.InpuData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	77

Name	DestinationAddress
Definition	Transport address containing the IP address or the DNS (Domain Name Server) address, followed by the character ':' and the port number if the default port is not used.
References	
Usage	Processing of a Transmit request message.
Type	TextString
Format	
CrossRef	TransmitRequest
XMLCoding	Attribute
ASN1Coding	311

Name	DefaultSelectedFlag
Definition	Selection of a menu entry to be displayed.
References	
Usage	In Input request message, it allows to select one or sevral menu entries before any user action.
Type	Boolean
Format	
CrossRef	InputRequest.Output.MenuEntry
XMLCoding	Attribute
ASN1Coding	390

Name	Device
<i>Definition</i>	Logical device located on a Sale Terminal or a POI Terminal, in term of class of information to output (display, print or store), or input (keyboard) for the Cashier or the Customer.
<i>References</i>	
<i>Usage</i>	<p>Define the logical device on which an operation required on the message request has to be processed.</p> <p>The physical device depends on the architecture and the implementation of the System providing output or input on the device.</p> <p>The name of the logical device is not attached a particular side of the Sale to POI interface, but has different meaning depending on the side of the interface.</p> <p>For instance, "CustomerDisplay" has different meaning on the Sale System, this is part of the Sale Terminal display that the Customer could see, and on the POI System, this is the interface with the Cardholder to enter the PIN and to display some information. Related to the Sale to POI interface, this is the POI</p>
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output DisplayResponse.OutputResult EnableServiceRequest.Output EventNotification.Output PrintRequest.Output PrintResponse.OutputResult InputRequest.Output InputRequest.InpuData InputResponse.OutputResult CardReaderInitRequest.Output CardReaderAPDUREquest.Output CardReaderPowerOffRequest.Output
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	78

<i>Label</i>	<i>Description</i>	<i>Code</i>
CashierDisplay	Used by the POI System (or the Sale System when the device is managed by the POI Terminal), to display some information to the Cashier.	
CustomerDisplay	Used by the Sale System (or the POI System when the device is managed by the Sale Terminal), to display some information to the Customer.	
CashierInput	Any kind of keyboard allowing all or part of the commands of the Input message request from the Sale System to the POI System (InputCommand data element). The output device attached to this input device is the CashierDisplay device.	
CustomerInput	Any kind of keyboard allowing all or part of the commands of the Input message request from the POI System to the Sale System (InputCommand data element). The output device attached to this input device is the CustomerDisplay device.	

Name	DeviceID
Definition	Identification of a device message pair
References	
Usage	<p>It allows a unique identification of a message pair, between a Sale System/Terminal and a POI System/Terminal during period of time, typically one day.</p> <p>This identifier is mandatory for messages of the "Device" MessageClass. For the Device Dialogue (i.e. device request coming from the sale outside a service request/response), DeviceID identifies alone the device message pair. For the Device Dialogue (i.e. device request sent inside a service request/response), DeviceID identifies the device message pair inside the "Service" message pair identified by ServiceID.</p> <p>This identifier is absent for messages of "Service", and "Event" MessageClass.</p> <p>The ServiceID of the message response is always the same value than the ServiceID of the message request.</p> <p>This identifier allows the recognition of duplicate message and association of a message response to its message request.</p> <p>Value of DeviceID could be a structured string of alpha characters, for instance with a prefix to easily distinguish a particular sequence of messages.</p>
Type	TextString
Format	{1,10}
CrossRef	MessageHeader InputUpdate.MessageReference
XMLCoding	Attribute
ASN1Coding	79

Name	DiagnosisRequest
Definition	Content of the Diagnosis Request message.
References	
Usage	It conveys Information related to the target POI for which the diagnosis is requested
Type	defined data structure
Format	
CrossRef	DiagnosisRequest
XMLCoding	
ASN1Coding	80

Component	Mult.	Constraint	Rule
POIID	[0..1]		default MessageHeader.POID
HostDiagnosisFlag	[0..1]	default False	
AcquirerID	[0..n]		Present if requesting the diagnosis of these hosts only.

Name	DiagnosisResponse
<i>Definition</i>	Content of the Diagnosis Response message.
<i>References</i>	
<i>Usage</i>	It conveys the result of the requested diagnosis and a possible message to display on a logical device.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	81

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
LoggedSaleID	[0..n]		If Sale Terminal logged to this POI Terminal
POIStatus	[0..1]		if Response.Result is Success
HostStatus	[0..n]		

Name	DigitInput
<i>Definition</i>	The digits which are typed by the Customer on the POI or the Cashier on the Sale Terminal.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	DigitString
<i>Format</i>	
<i>CrossRef</i>	InputResponse.InputResult.Input
<i>XMLCoding</i>	
<i>ASN1Coding</i>	82

Name	DisableCancelFlag
<i>Definition</i>	Indicates if the Cancel function key has to be desactivated (value True).
<i>References</i>	
<i>Usage</i>	During the processing of an Input command GetConfirmation, SiteManager, or GetMenuEntry.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	InputRequest.InpuData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	383

Name	DisableCorrectFlag
<i>Definition</i>	Indicates if the Correct function key has to be desactived (value True).
<i>References</i>	
<i>Usage</i>	During the processing of an Input command GetConfirmation, SiteManager, or GetMenuEntry.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	InputRequest.InpuData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	384

Name	DisableValidFlag
<i>Definition</i>	Indicates if the Valid function key has to be desactived (value True).
<i>References</i>	
<i>Usage</i>	During the processing of an Input command GetConfirmation, SiteManager, or GetMenuEntry.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	InputRequest.InpuData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	385

Name	DisplayOutput
Definition	Information to display and the way to process the display.
References	
Usage	It contains a complete display operation for a Display or an Input Device type. For the Input Devices, Diagnosis and EnableService, ResponseRequiredFlag and MinimumDisplayTime shall be absent.
Type	defined data structure
Format	
CrossRef	AbortRequest DisplayRequest EnableServiceRequest EventNotification InputRequest CardReaderInitRequest CardReaderAPDUREquest CardReaderPowerOffRequest
XMLCoding	
ASN1Coding	83

Component	Mult.	Constraint	Rule
ResponseRequiredFlag	[0..1]	default True	
MinimumDisplayTime	[0..1]	default 0	
Device	[1..1]		CashierDisplay, CustomerDisplay
InfoQualify	[1..1]		Status, Error, Display, POIReplication
OutputContent	[1..1]		
MenuEntry	[0..n]		One instance of MenuEntry per item to display in the menu for the get menu input command.
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display or print.

Name	DisplayRequest
Definition	Content of the Display Request message.
References	
Usage	It conveys the data to display and the way to process the display. It contains the complete content to display. It might contain an operation (the DisplayOutput element) per Display Device type.
Type	defined data structure
Format	
CrossRef	DisplayRequest
XMLCoding	
ASN1Coding	84

Component	Mult.	Constraint	Rule
DisplayOutput	[1..n]		Complete display content for output devices. At most one DisplayOutput per Device/InfoQualify pair

Name	DisplayResponse
Definition	Content of the Display Response message.
References	
Usage	It conveys the result of the display, parallel to the message request, except if response not required and absent.
Type	defined data structure
Format	
CrossRef	DisplayResponse
XMLCoding	
ASN1Coding	85

Component	Mult.	Constraint	Rule
OutputResult	[1..n]		One per DisplayOutput item of the request, and in the same order.

Name	DocumentQualifier
<i>Definition</i>	Qualification of the document to print to the Cashier or the Customer.
<i>References</i>	
<i>Usage</i>	Allow the manager of the printer, Sale or POI Terminal, to send the information to a particular physical printer or to use the paper type accordingly. Allow the choice for the duplication of the payment receipt.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentReceipt PrintRequest.Output TransactionStatusRequest
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	86

Label	Description	Code
SaleReceipt	Where the POI system print the Sale receipt when requested by the Sale Terminal.	
CashierReceipt	Where the Sale system print the Cashier copy of the Payment receipt when requested by the POI Terminal.	
CustomerReceipt	Where you print the Customer Payment receipt that could be located on the Sale Terminal or in some cases a restricted Customer Sale ticket on the POI Terminal.	
Document	When the POI System wants to print specific document (check, dynamic currency conversion ...). Used by the Sale System when the printer is not located on the Sale System.	
Voucher	Coupons, voucher or special ticket generated by the POI or the Sale System and to be printed.	
Journal	When the POI or the Sale System wants to store a message on the journal printer or electronic journal of the Sale Terminal (it is sometimes a Sale Logging/Journal Printer).	

5.2.2.4 E-F

Name	EanUpc
<i>Definition</i>	Standard product code of item purchased with the transaction.
<i>References</i>	
<i>Usage</i>	Must be present in a StoredValueData to identify the stored value product.
<i>Type</i>	DigitString
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.SaleItem LoyaltyRequest.TransactionData.SaleItem StoredValueRequest.StoredValueData.SaleItemRebate
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	87

Name	EnableServiceRequest
<i>Definition</i>	Content of the Enable Service Request message.
<i>References</i>	
<i>Usage</i>	It conveys the services that will be enabled for the POI Terminal without the request of the Sale System, and a possible invitation for the Customer to start the services.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	EnableServiceRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	88

Component	Mult.	Constraint	Rule
TransactionAction	[1..1]		
ServicesEnabled	[0..1]		Mandatory if TransactionAction is "StartTransaction", absent if not.
DisplayOutput	[0..1]		

Name	EnableServiceResponse
<i>Definition</i>	Content of the Enable Service Response message.
<i>References</i>	
<i>Usage</i>	It conveys the result of the Enable Service processing.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	EnableServiceResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	89

Component	Mult.	Constraint	Rule
Response	[1..1]		

Name	EndDate
<i>Definition</i>	Date time of the end of an operation.
<i>References</i>	
<i>Usage</i>	Identifies the last time an operation occurred on a customer order
<i>Type</i>	ISODateTime
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.CustomerOrder PaymentRequest.PaymentData.CustomerOrder PaymentResponseCustomerOrder ReversalResponse.CustomerOrder
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	419

Name	EncrPINBlock
Definition	Encrypted PIN
References	
Usage	This data structure is the PIN block encrypted by the application with CMS encryption data structure (EnvelopedData).
Type	ContentInformationType
Format	{8,}
CrossRef	PINRequest.CardholderPIN PINResponse.CardholderPIN
XMLCoding	
ASN1Coding	90

Name	EndOfLineFlag
Definition	Text is at the end of a line.
References	
Usage	Allow the display or the print of a "new line" and a "carriage return" characters after the formatted text.
Type	Boolean
Format	
CrossRef	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputElement.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
XMLCoding	Attribute
ASN1Coding	91

Name	EntryMode
Definition	Entry mode of the payment instrument information
References	EPAS Acquirer Protocol: POICardReadCapabilities
Usage	In the Payment, Loyalty or StoredValue Request messages, it informs the POI System the entry mode of the payment instrument information when read by the Sale Terminal. (e.g. because the payment instrument information are a bar-code read by the Cashier on a scanner device). In the Payment, Loyalty or StoredValue Response messages, it informs the Sale System the entry mode of the payment instrument.
Type	Cluster
Format	
CrossRef	BalanceInquiryRequest.LoyaltyData.LoyaltyAccountID BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData BalanceInquiryResponse.LoyaltyResult.LoyaltyAccountID BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID CardAcquisitionResponse.LoyaltyAccount.LoyaltyAccountID CardAcquisitionResponse.PaymentInstrumentData.StoredValueAccountID CardReaderInitResponse PaymentRequest.PaymentData.PaymentInstrumentData.CardData PaymentRequest.LoyaltyData.LoyaltyAccountID PaymentRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID PaymentResponse.PaymentResult.PaymentInstrumentData.CardData PaymentResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID PaymentResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID LoyaltyRequest.LoyaltyData.LoyaltyAccountID LoyaltyResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID StoredValueRequest.StoredValueData.StoredValueAccountID StoredValueResponse.StoredValueResult.StoredValueAccountID
XMLCoding	Attribute
ASN1Coding	50

Label	Description	Code
RFID	Payment instrument information are taken from RFID	
Keyed	Manual key entry	
Manual	Reading of embossing or OCR of printed data either at time of transaction or after the event.	
File	Account data on file	
Scanned	Scanned by a bar code reader.	
MagStripe	Magnetic stripe card reader.	
ICC	Contact ICC (asynchronous)	
SynchronousICC	Contact ICC (synchronous)	
Tapped	Contactless card reader Magnetic Stripe	
Contactless	Contactless card reader conform to ISO 14443	
Mobile	Mobile phone.	

Name	ErrorCondition
Definition	Condition that has produced an error on the processing of a message request
References	LoginResponse.Result
Usage	Allow refinement on processing an error inside a message response, and a specific behaviour of the software, depending on its implementation. The number of enumerated values is limited to avoid complexity of both protocol and software design. This component could also be present to declare a warning to the requestor if the error is not fatal for the processing of the request, in order to react during the testing phase or correct the problem in the field.
Type	Enumeration
Format	
CrossRef	BalanceInquiryResponse CardAcquisitionResponse.Response DiagnosisResponse.Response DisplayResponse.OutputResult.Response EnableServiceResponse.Response GetTotalsResponse GetTotalsResponse.Totals.TransactionTotals InputResponse.InputResult.Response InputResponse.OutputResult.Response LoginResponse.Response LoginResponse.Response LoginResponse.Response PaymentResponse.Response PINResponse.Response PrintResponse.OutputResult.Response ReconciliationResponse.Response CardReaderInitResponse.Response CardReaderAPDURESPONSE.Response CardReaderPowerOffResponse.Response ReconciliationResponse.Totals.TransactionTotals ReversalResponse.Response SoundResponse.Response StoredValueResponse TransactionStatusResponse.Response TransmitResponse.Response
XMLCoding	Attribute
ASN1Coding	95

Label	Description	Code
Aborted	The Initiator of the request has sent an Abort message request, which was accepted and processed.	
Busy	The system is busy, try later	
Cancel	The user has aborted the transaction on the PED keyboard, for instance during PIN entering.	
DeviceOut	Device out of order	
InsertedCard	If the Input Device request a NotifyCardInputFlag and the Customer enters a card in the card reader without answers to the Input command, the POI abort the Input command processing, and answer a dedicated ErrorCondition value in the Input response message.	
InProgress	The transaction is still in progress and then the command cannot be processed	
LoggedOut	Not logged in	
MessageFormat	Error on the format of the message, AdditionalResponse shall contain the identification of the data, and the reason in clear text.	
NotAllowed	A service request is sent during a Service dialogue. A combination of services not possible to provide. During the CardReaderInit message processing, the user has entered a card which has to be protected by the POI, and cannot be processed with this device request from the external, and then the Sale System.	

NotFound	The transaction is not found (e.g. for a reversal or a repeat)	
PaymentRestriction	Some sale items are not payable by the card proposed by the Customer.	
Refusal	The transaction is refused by the host or the rules associated to the card, and cannot be repeated.	
UnavailableDevice	The hardware is not available (absent, not configured...)	
UnavailableService	The service is not available (not implemented, not configured, protocol version too old...)	
InvalidCard	The card entered by the Customer cannot be processed by the POI because this card is not configured in the system	
UnreachableHost	Acquirer or any host is unreachable or has not answered to an online request, so is considered as temporary unavailable. Depending on the Sale context, the request could be repeated (to be compared with "Refusal").	
WrongPIN	The user has entered the PIN on the PED keyboard and the verification fails.	

Name	EventDetails
Definition	Information about the event the POI notifies to the Sale System.
References	
Usage	Free text to log.
Type	TextString
Format	
CrossRef	EventNotification
XMLCoding	
ASN1Coding	92

Name	EventNotification
Definition	Content of the EventNotification message.
References	
Usage	It conveys Information related to the event, and possible action (maintenance, message to display).
Type	defined data structure
Format	
CrossRef	EventNotification
XMLCoding	
ASN1Coding	93

Component	Mult.	Constraint	Rule
TimeStamp	[1..1]		
EventToNotify	[1..1]		
EventDetails	[0..1]		Mandatory for transaction identification when <i>EventToNotify</i> = "SaleWakeUp", "FunctionKeyPressed" or "SaleAdmin". Otherwise if present, the Sale logs it for further examination
RejectedMessage	[0..1]		Mandatory if <i>EventToNotify</i> is "Reject", absent in other cases
MaintenanceRequiredFlag	[0..1]	default False	
CustomerLanguage	[0..1]		Mandatory when <i>EventToNotify</i> = "CustomerLanguage", otherwise absent.
DisplayOutput	[0..1]		To display an event message

Name	EventToNotify
<i>Definition</i>	Event the POI notifies to the Sale System.
<i>References</i>	
<i>Usage</i>	Any event which occurs outside a transaction requested by the Sale System. The pair of data elements (EventToNotify, EventDetails) follows the same formatting rules than the pair of data elements (ErrorCondition, AdditionalResponse) inside the Response data structure.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	EventNotification
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	94

<i>Label</i>	<i>Description</i>	<i>Code</i>
BeginMaintenance	Begin of POI Maintenance	
EndMaintenance	End of POI Maintenance	
Shutdown	The POI Terminal or the POI System is shutting down	
Initialised	The POI Terminal or the POI System is now ready to work	
OutOfOrder	The POI Terminal or the POI System cannot work	
Completed	An Abort request has been sent to abort a message which is already completed.	
Abort	One or several device request has been sent by the POI during the processing of a service requested by the Sale System. The processing is cancelled by the Customer or stopped by the POI. If the device response is not received by the POI, an event is sent to inform the Sale to abort internally these device requests.	
SaleWakeUp	A POI terminal requests the payment of the transaction identified by the content of EventDetails in the Event notification.	
SaleAdmin	The POI has performed, or want to perform an automatic administrative process, e.g. the reports at the end of day.	
CustomerLanguage	The customer has selected a different language on the POI.	
KeyPressed	The customer has pressed a specific key on the POI.	
SecurityAlarm	Problem of security	
StopAssistance	When the Customer assistance is stopped, because the Customer has completed its input.	
CardInserted	A card is inserted in the card reader (see Input request and NotifyCardInputFlag)	
CardRemoved	A card is removed from the card reader.	
Reject	A message request is rejected. An error explanation and the message in error have to be put in the EventDetails data element.	

Name	ExpiryDate
Definition	Date after which the card cannot be used.
References	EMV - Tag 5F24 - Application Expiration Date
Usage	If EMV expiry date is present, it overrides Track2 information. Format is MMYY.
Type	DigitString
Format	{4,4}
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData BalanceInquiryResponse.StoredValueResult.StoredValueAccountID CardAcquisitionResponse.PaymentInstrumentData.CardData.SensitiveCardData CardAcquisitionResponse.PaymentInstrumentData.StoredValueAccountID PaymentRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData PaymentRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID PaymentResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID StoredValueRequest.StoredValueData.StoredValueAccountID StoredValueResponse.StoredValueResult.StoredValueAccountID
XMLCoding	Attribute
ASN1Coding	96

Name	ExpiryDateTime
Definition	Expiry date and time.
References	
Usage	Limit the validity of a payment token.
Type	ISODateTime
Format	
CrossRef	CardAcquisitionResponse.PaymentInstrumentData.CardData.PaymentToken PaymentResponse.PaymentResult.PaymentInstrumentData.CardData.PaymentToken
XMLCoding	Attribute
ASN1Coding	410

Name	FirstAmount
<i>Definition</i>	First amount of a payment.
<i>References</i>	
<i>Usage</i>	First amount of the payment instalments.
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.PaymentData.Instalment
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	376

Name	FirstPaymentDate
<i>Definition</i>	First date of a payment.
<i>References</i>	
<i>Usage</i>	For instalment, the date of the first payments, if not immediate.
<i>Type</i>	ISODate
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.PaymentData.Instalment
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	373

Name	Font
<i>Definition</i>	Name of the font.
<i>References</i>	
<i>Usage</i>	Used to change character font of the output, as agreed between the POI and Sale Systems.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	98

Name	ForceCustomerSelectionFlag
<i>Definition</i>	Indicates if the Customer realises the selection of the card application.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionRequest.TransactionData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	100

Name	ForceEntryMode
<i>Definition</i>	Payment instrument entry mode requested by the Sale System.
<i>References</i>	
<i>Usage</i>	Avoid retry on an out of order card reading device, when the sale system knows that some card entry modes on the POI do not work. Usage of the CardReader entry modes The order of entry modes, give a priority to the various card readings.
<i>Type</i>	Cluster
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionRequest.TransactionData PaymentRequest.TransactionData.TransactionConditions CardReaderInitRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	99

<i>Label</i>	<i>Description</i>	<i>Code</i>
RFID	Payment instrument information are taken from RFID	
Keyed	Manual key entry	
Manual	Reading of embossing or OCR of printed data either at time of transaction or after the event.	
File	Account data on file	
Scanned	Scanned by a bar code reader.	
MagStripe	Magnetic stripe	
ICC	Contact ICC (asynchronous)	
SynchronousICC	Contact ICC (synchronous)	
Tapped	Contactless card reader Magnetic Stripe	
Contactless	Contactless card reader conform to ISO 14443	
CheckReader	Check Reader	

Name	ForceOnlineFlag
<i>Definition</i>	Indicates if the Cashier requires POI forces online access to the Acquirer.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.TransactionConditions
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	101

Name	ForecastedAmount
<i>Definition</i>	Depending on the choice of the sale system it could equal the initial amount of the order or the global amount of the order.
<i>References</i>	
<i>Usage</i>	If ForecastedAmount equals the initial amount of the order it will allow the system to follow any additional payment. In case of equality with the global amount of the order, the system will then be able to follow remaining amounts.
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.CustomerOrder PaymentRequest.PaymentData.CustomerOrder PaymentResponse.CustomerOrder ReversalRequest.CustomerOrder ReversalResponse.CustomerOrder
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	429

Name	FraudPreventionFlag
<i>Definition</i>	Indicate a suspicion of fraud by the POI System.
<i>References</i>	
<i>Usage</i>	Could be sent to True by the POI system to notify to the Sale system and the Cashier, that a suspicion of fraud had been detected on the POI as an unexpected reboot of the POI.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.POIStatus LoginResponse.POIStatus
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	316

Name	FromRightToLeftFlag
Definition	Indicate if the entered character has to be displayed from the right to the left of the display field.
References	
Usage	For Digit, Text or DecimalString Input command, for some type of value as an amount.
Type	Boolean
Format	
CrossRef	InputRequest.InpuData
XMLCoding	Attribute
ASN1Coding	102

Name	FunctionKey
Definition	The number of the function key which is typed by the Customer on the POI or the Cashier on the Sale Terminal.
References	
Usage	
Type	DigitString
Format	
CrossRef	InputResponse.InputResult.Input
XMLCoding	
ASN1Coding	310

5.2.2.5 G-I

Name	GenericProfile
Definition	Functional profile of the Sale to POI protocol.
References	
Usage	Sent in the Login Request to identify the messages that might be requested or received by the Sale Terminal during the session. Sent in the Login Response to identify the messages that might be processed or sent by the POI Terminal during the session.
Type	Enumeration
Format	
CrossRef	LoginRequest.SaleSystemData.SaleTerminalData.SaleProfile LoginResponse.POISystemData.POITerminalData.POIPProfile
XMLCoding	Attribute
ASN1Coding	103

Label	Description	Code
Basic	Protocol services that needs to be implemented by all the Sale and POI	
Standard	Protocol services involving interaction between Sale System and POI System as devices shared between the two Systems.	
Extended	Complete Protocol services	

Name	GeographicCoordinates
Definition	Location on the Earth specified by two numbers representing vertical and horizontal position.
References	
Usage	Identifies the geographic location of a mobile phone.
Type	defined data structure
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation CardAcquisitionResponse.PaymentInstrumentData.MobileData.Geolocation PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation
XMLCoding	
ASN1Coding	356

Component	Mult.	Constraint	Rule
Latitude	[1..1]		
Longitude	[1..1]		

Name	Geolocation
Definition	Geographic location specified by geographic or UTM coordinates.
References	
Usage	Identifies the geographic location of a mobile phone.
Type	defined data structure
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData CardAcquisitionResponse.PaymentInstrumentData.MobileData PaymentRequest.PaymentData.PaymentInstrumentData.MobileData PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData
XMLCoding	
ASN1Coding	326

Component	Mult.	Constraint	Rule
GeographicCoordinates	[0..1]		
UTMCoordinates	[0..1]		

Name	GetTotalsRequest
<i>Definition</i>	Content of the Get Totals Request message.
<i>References</i>	
<i>Usage</i>	It conveys information from the Sale System related to the scope and the format of the totals to be computed by the POI System.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	GetTotalsRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	104

Component	Mult.	Constraint	Rule
TotalDetails	[0..1]		Require to present totals per value of element included in this cluster (POI Terminal, Sale Terminal, Cashier, Shift, TotalsGroupID)
TotalFilter	[0..1]		If structure is not empty

Name	GetTotalsResponse
<i>Definition</i>	Content of the Reconciliation Response message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Reconciliation transaction processed by the POI System
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	GetTotalsResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	105

Component	Mult.	Constraint	Rule
Response	[1..1]		
POIReconciliationID	[1..1]		
TransactionTotals	[0..n]		if Response.Result is Success

Name	GlobalCorrectionFlag
<i>Definition</i>	Indicates, when the user press the Correct function key in an input entry, if all the entered characters are removed (value True) or only the last entered character if any (value False).
<i>References</i>	
<i>Usage</i>	During the processing of an Input command TString, DigitString or DecimalString.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	InputRequest.InpuData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	404

Name	GlobalStatus
Definition	Global status of a POI Server or POI Terminal.
References	
Usage	
Type	Enumeration
Format	
CrossRef	DiagnosisResponse.POISatus LoginResponse.POISatus
XMLCoding	Attribute
ASN1Coding	106

Label	Description	Code
OK	The POI is ready to receive and process a request	
Busy	The POI Terminal cannot process a request because another processing is in progress.	
Maintenance	The POI is in maintenance processing	
Unreachable	The POI is unreachable or not responding	

Name	HostReconciliationID
Definition	Identifier of a reconciliation period with a payment or loyalty host.
References	
Usage	Allow the assignment of a transaction to the Acquirer reconciliation (or batch).
Type	TextString
Format	
CrossRef	GetTotalsResponse.Totals.TransactionTotals LoyaltyResponse.LoyaltyResult.LoyaltyAcquirerData ReconciliationResponse.Totals.TransactionTotals
XMLCoding	Attribute
ASN1Coding	300

Name	HostDiagnosisFlag
Definition	Indicates if Host Diagnosis are required
References	
Usage	Allows the Sale to request the POI to send a diagnosis to every Acquirer, Intermediary Agent, or other Hosts involved in the processing of the services to the Sale System
Type	Boolean
Format	
CrossRef	DiagnosisRequest
XMLCoding	Attribute
ASN1Coding	107

Name	HostStatus
Definition	State of a Host.
References	
Usage	Indicate the reachability of the host by the POI Terminal.
Type	defined data structure
Format	
CrossRef	DiagnosisResponse
XMLCoding	
ASN1Coding	108

Component	Mult.	Constraint	Rule
AcquirerID	[1..1]		
IsReachableFlag	[0..1]	default True	

Name	HostTransactionID
Definition	Identification of the transaction by the host in charge of the stored value transaction
References	
Usage	
Type	TransactionIdentificationType
Format	
CrossRef	StoredValueRequest.StoredValueData.OriginalPOITransaction StoredValueResponse.StoredValueResult
XMLCoding	
ASN1Coding	109

Name	ICCResetData
<i>Definition</i>	Data of a Chip Card related to the reset of the chip.
<i>References</i>	
<i>Usage</i>	Card reader device request
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardReaderInitResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	110

Component	Mult.	Constraint	Rule
ATRValue	[0..1]		if available
CardStatusWords	[0..1]		if available

Name	IdentificationSupport
<i>Definition</i>	Support of the loyalty account identification
<i>References</i>	
<i>Usage</i>	Allows knowing where and how you have found the loyalty account identification.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest.LoyaltyData.LoyaltyAccountID BalanceInquiryResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID CardAcquisitionResponse.LoyaltyAccount.LoyaltyAccountID LoyaltyResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID PaymentResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	114

Label	Description	Code
NoCard	The identification is not found on a card	
LoyaltyCard	The identification is on a card dedicated to this loyalty brand.	
HybridCard	The identification is on a card which might be used both for the loyalty and the payment.	
LinkedCard	The loyalty account is implicitly attached to the payment card. This is usually detected by the loyalty Acquirer.	

Name	IdentificationType
<i>Definition</i>	Type of account identification
<i>References</i>	
<i>Usage</i>	In a request message, it informs the POI System the type of the account or card identification, when provided by the Sale Terminal. (e.g. because the card information are a bar-code read by the Cashier on a scanner device). In a response message, it informs the Sale System the type of the account or card identification.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest.LoyaltyData.LoyaltyAccountID BalanceInquiryRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID BalanceInquiryResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID CardAcquisitionResponse.LoyaltyAccount.LoyaltyAccountID CardAcquisitionResponse.PaymentInstrumentData.StoredValueAccountID LoyaltyRequest.LoyaltyData.LoyaltyAccountID LoyaltyResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID PaymentRequest.LoyaltyData.LoyaltyAccountID PaymentRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID PaymentResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID PaymentResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID StoredValueRequest.StoredValueData.StoredValueAccountID StoredValueResponse.StoredValueResult.StoredValueAccountID
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	115

Label	Description	Code
PAN	Standard card identification (card number)	
ISOTrack2	ISO Track 2 including identification.	
BarCode	Bar-code with a specific form of identification	
AccountNumber	Account number	
PhoneNumber	A phone number identifies the account on which the phone card is assigned.	

Name	IMEI
<i>Definition</i>	International Mobile Equipment Identity.
<i>References</i>	ITU-T E.212
<i>Usage</i>	Unique number associated with the mobile phone device.
<i>Type</i>	DigitString
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.SensitiveMobileData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.SensitiveMobileData CardAcquisitionResponse.PaymentInstrumentData.MobileData.SensitiveMobileData PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.SensitiveMobileData PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.SensitiveMobileData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	331

Name	ImmediateResponseFlag
Definition	Request Immediate response to the message without waiting for the completion of the command.
References	
Usage	During the processing of an Input Device request containing the GetAnyKey command, indicate to wait for the confirmation of the message or not.
Type	Boolean
Format	
CrossRef	InputRequest.InpuData
XMLCoding	Attribute
ASN1Coding	111

Name	IMSI
Definition	International Mobile Subscriber Identity.
References	ITU-T E.212
Usage	Unique number associated with the mobile phone user, containing the Mobile Country Code (MCC), the Mobile Network Code (MNC), and the Mobile Identification Number (MSIN)
Type	DigitString
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.SensitiveMobileData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.SensitiveMobileData CardAcquisitionResponse.PaymentInstrumentData.MobileData.SensitiveMobileData PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.SensitiveMobileData PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.SensitiveMobileData
XMLCoding	Attribute
ASN1Coding	330

Name	InfoQualify
<i>Definition</i>	Qualification of the information to sent to an output logical device, to display or print to the Cashier or the Customer.
<i>References</i>	
<i>Usage</i>	Allow the manager of the device, Sale or POI Terminal, to send the information to a particular physical device or to present the information accordingly.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output DisplayResponse.OutputResult EnableServiceRequest.Output EventNotification.Output PrintRequest.Output PrintResponse.OutputResult InputRequest.Output InputRequest.InpuData InputResponse.OutputResult CardReaderInitRequest.Output CardReaderAPDUREquest.Output CardReaderPowerOffRequest.Output
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	112

Label	Description	Code
Status	The information is a new state on which the message sender is entering. For instance, during a payment, the POI could display to the Cashier that POI request an authorisation to the host acquirer.	
Error	The information is related to an error situation occurring on the message sender.	
Display	Standard display interface.	
Sound	Standard sound interface.	
Input	Answer to a question or information to be entered by the Cashier or the Customer, at the request of the POI Terminal or the Sale Terminal.	
POIReplication	Information displayed on the Cardholder POI interface, replicated on the Cashier interface.	
CustomerAssistance	Input of the Cardholder POI interface which can be entered by the Cashier to assist the Customer.	
Receipt	Where you print the Payment receipt that could be located on the Sale Terminal or in some cases a restricted Sale ticket on the POI Terminal.	
Document	When the POI System wants to print specific document (check, dynamic currency conversion ...). Used by the Sale System when the printer is not located on the Sale System.	
Voucher	Coupons, voucher or special ticket generated by the POI or the Sale System and to be printed.	

Name	Input
Definition	Data entered by the user, related to the input command.
References	
Usage	Choice of a data which contains data entered by the user on the requested device, depending on the requested InputCommand: <ul style="list-style-type: none"> ▪ GetConfirmation: the input is in ConfirmedFlag. ▪ GetAnyKey: there is no input. ▪ GetFunctionKey: the input is in FunctionKey.
Type	defined data structure
Format	
CrossRef	InputResponse.InputResult
XMLCoding	
ASN1Coding	116

Component	Mult.	Constraint	Rule
InputCommand	[1..1]		Copy
ConfirmedFlag	[0..1]		Mandatory, if InputCommand is GetConfirmation or SiteManager Not allowed, otherwise
FunctionKey	[0..1]		Mandatory, if InputCommand is GetFunctionKey Not allowed, otherwise
TextInput	[0..1]		Mandatory, if InputCommand is TextString Not allowed, otherwise
DigitInput	[0..1]		Mandatory, if InputCommand is DigitString Not allowed, otherwise
Password	[0..1]		Mandatory, if InputCommand is Password Not allowed, otherwise
MenuEntryNumber	[0..1]		Mandatory, if InputCommand is GetMenuEntry Not allowed, otherwise

Name	InputCommand
Definition	Type of requested input
References	
Usage	Allow medium level input commands
Type	Enumeration
Format	
CrossRef	InputRequest.InpuData
XMLCoding	Attribute
ASN1Coding	117

Label	Description	Code
GetAnyKey	Wait for a key pressed on the Terminal, to be able to read the message displayed on the Terminal.	
GetConfirmation	Wait for a confirmation Yes (Y) or No (N) on the Sale Terminal. Wait for a confirmation (Valid or Cancel button) on the POI Terminal. The result of the command is a Boolean: True or False.	
SiteManager	Wait for a confirmation Yes (Y) or No (N) of the Site Manager on the Sale Terminal.	
TextString	Wait for a string of alphanumeric characters, the length range could be specified.	
DigitString	Wait for a string of digit characters, the length range could be specified.	
DecimalString	Wait for a string of digit characters with a decimal point, the length range could be specified.	
GetFunctionKey	Wait for a function key pressed on the Terminal: From POI, Valid, Clear, Correct, Generic Function key number. From Sale, Generic Function key.	
GetMenuEntry	To choose an entry among a list of entrys (all of them are not necessary selectable). The OutputFormat has to be MenuEntry.	
Password	Request to enter a password with masked characters while typing the password.	

Name	InputData
Definition	Information related to an Input request.
References	
Usage	<p>It conveys the target input logical device, the type of input command, and possible minimum and maximum length of the input.</p> <p>In addition, if the requestor might require to receive an Event Notification if a card is inserted in a card reader, with the NotifyCardInputFlag.</p>
Type	defined data structure
Format	
CrossRef	InputRequest
XMLCoding	
ASN1Coding	118

Component	Mult.	Constraint	Rule
Device	[1..1]		CashierInput , CustomerInput, CustomerAssistance
InfoQualify	[1..1]		Input, CustomerAssistance
InputCommand	[1..1]		
NotifyCardInputFlag	[0..1]	default False	
MaxInputTime	[0..1]		If time limit for responding
ImmediateResponseFlag	[0..1]	default False	Optional if InputCommand is "GetAnyKey".
MinLength	[0..1]		Not allowed if InputCommand is not TextString or DigitString
MaxLength	[0..1]		Not allowed if InputCommand is not TextString or DigitString
MaxDecimalLength	[0..1]		Not allowed if InputCommand is not DecimalString Greater than <i>MinLength</i> , lower than <i>MaxLength</i> .
WaitUserValidationFlag	[0..1]	default True	Optional if MaxLength or MaxDecimalLength present
DefaultInputString	[0..1]		Not allowed if InputCommand is not TextString, DigitString or DecimalString
StringMask	[0..1]		Not allowed if InputCommand is not TextString, DigitString or DecimalString
FromRightToLeftFlag	[0..1]	default False	Not allowed if InputCommand is not TextString, DigitString or DecimalString
MaskCharactersFlag	[0..1]	default False	Not allowed if InputCommand is not TextString, DigitString or Password
BeepKeyFlag	[0..1]	default False	
GlobalCorrectionFlag	[0..1]	default False	Not allowed if InputCommand is not TextString, DigitString, Password or DecimalString
DisableCancelFlag	[0..1]	default False	Not allowed if InputCommand is not GetConfirmation, SiteManager, or GetMenuEntry
DisableCorrectFlag	[0..1]	default False	Not allowed if InputCommand is not GetConfirmation, SiteManager, or GetMenuEntry
DisableValidFlag	[0..1]	default False	Not allowed if InputCommand is not GetConfirmation, SiteManager, or GetMenuEntry
MenuBackFlag	[0..1]	default False	Allowed for the GetMenuEntry value of InputCommand.

Name	InputRequest
<i>Definition</i>	Content of the Input Request message.
<i>References</i>	
<i>Usage</i>	<p>It conveys data to display and the way to process the display, and contains the complete content to display.</p> <p>In addition to the display on the Input Device, it might contain an operation (the <i>DisplayOutput</i> element) per Display Device type.</p>
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	InputRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	119

Component	Mult.	Constraint	Rule
DisplayOutput	[0..1]		Mandatory if the display device is managed by the receiver.
InputData	[1..1]		

Name	InputResponse
<i>Definition</i>	Content of the Input Response message.
<i>References</i>	
<i>Usage</i>	<p>It conveys:</p> <ul style="list-style-type: none"> ▪ The result of the outputs, parallel to the message request, except if response not required and absent. ▪ The result of the input
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	InputResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	120

Component	Mult.	Constraint	Rule
OutputResult	[0..1]		If <i>DisplayOutput</i> present in the request.
InputResult	[1..1]		

Name	InputResult
Definition	Information related to the result the input.
References	
Usage	In the message response, it contains the result and the content of the input.
Type	defined data structure
Format	
CrossRef	InputResponse
XMLCoding	
ASN1Coding	121

Component	Mult.	Constraint	Rule
Device	[1..1]		Copy
InfoQualify	[1..1]		Copy
Response	[1..1]		
Input	[0..1]		If Response.Result is Success

Name	InputUpdate
<i>Definition</i>	Content of the Input Update message.
<i>References</i>	
<i>Usage</i>	It conveys update of the display of an Input request in progress.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	InputUpdate
<i>XMLCoding</i>	
<i>ASN1Coding</i>	386

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
MessageReference	[1..1]		
OutputContent	[1..1]		
MenuEntry	[0..n]		
OutputSignature	[0..1]		
MinLength	[0..1]		If present in the Input to update.
MaxLength	[0..1]		If present in the Input to update.
MaxDecimalLength	[0..1]		If present in the Input to update.

Name	Instalment
<i>Definition</i>	Information related an instalment transaction.
<i>References</i>	
<i>Usage</i>	To request an instalment to the issuer, or to make individual instalments of a payment transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.PaymentData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	367

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
InstalmentType	[0..1]		
SequenceNumber	[0..1]		
PlanID	[0..1]		
Period	[0..1]		
PeriodUnit	[0..1]		
FirstPaymentDate	[0..1]		
TotalNbOfPayments	[0..1]		
CumulativeAmount	[0..1]		
FirstAmount	[0..1]		
Charges	[0..1]		

Name	InstalmentType
Definition	Type of instalment transaction.
References	
Usage	For requesting an instalment payment transaction.
Type	Cluster
Format	
CrossRef	PaymentRequest.PaymentData.Instalment
XMLCoding	
ASN1Coding	368

Label	Description	Code
DeferredInstalments	The payment of the service or goods is deferred.	
EqualInstalments	The payment is split in several instalments of equal amounts.	
InequalInstalments	The payment is split in several instalments of different amounts.	

Name	IntegratedPrintFlag
<i>Definition</i>	Type of the print integrated to other prints.
<i>References</i>	
<i>Usage</i>	Allows a separated printing (paper cut if available), or integration to the sale receipt or other print If the printing is integrated, the response to the request is always immediate, even if the ResponseMode is set to PrintEnd.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentReceipt PrintRequest.Output
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	113

Name	IsReachableFlag
<i>Definition</i>	Indicate if a Host is reachable
<i>References</i>	
<i>Usage</i>	In the Response of a POI Diagnosis, if requested by the Sale.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.HostStatus
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	122

Name	ItemAmount
<i>Definition</i>	Total amount of the item line.
<i>References</i>	
<i>Usage</i>	In case of rebate (in the data structure Rebates), this is the total amount of the item line after rebate.
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.SaleItem PaymentResponse.LoyaltyResult.Rebates.SaleItemRebate LoyaltyRequest.TransactionData.SaleItem LoyaltyResponse.LoyaltyResult.Rebates.SaleItemRebate StoredValueRequest.StoredValueData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	123

Name	ItemID
<i>Definition</i>	Item identification inside a transaction (0 to n).
<i>References</i>	
<i>Usage</i>	Allow to have the identification and the offset of an item included in a transaction (request.SaleItem). Identification of new product (unique for all the new item in the response.SaleItemRebate) Link to the SaleItem entry for a stored value card product identifier StoredValueAccountID
<i>Type</i>	Integer
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest.StoredValueAccountID LoyaltyRequest.TransactionData.SaleItem LoyaltyResponse.LoyaltyResult.Rebates.SaleItemRebate PaymentRequest.TransactionData.SaleItem PaymentResponse.LoyaltyResult.Rebates.SaleItemRebate StoredValueRequest.StoredValueData.SaleItemRebate LoyaltyAccountID StoredValueRequest.StoredValueData.StoredValueAccountID StoredValueResponse.StoredValueResult.StoredValueAccountID
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	124

5.2.2.6 J-L

Name	KeyReference
<i>Definition</i>	Identify the key to use to encrypt the PIN block.
<i>References</i>	
<i>Usage</i>	When the encrypted PIN is provided as an input or an output of the PIN service. There are no predefined values for this identification.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	PINRequest.PINRequest
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	125

Name	Language
<i>Definition</i>	Identification of a language.
<i>References</i>	
<i>Usage</i>	Language of message to display (the Cardholder on the Sale or the Cashier on the POI)
<i>Type</i>	ISOLanguage2A
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.MessageReference EnableServiceRequest.Output.OutputElement.MessageReference PrintRequest.Output.OutputElement.MessageReference InputRequest.OutputElement.MessageReference InputRequest.MenuEntry.Predefined Content CardReaderInitRequest.Output.OutputElement.MessageReference CardReaderAPDUREquest.Output.OutputElement.MessageReference CardReaderPowerOffRequest.Output.OutputElement.MessageReference SoundRequest.SoundContent
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	126

Name	Latitude
<i>Definition</i>	Angular distance of a location on the earth south or north of the equator.
<i>References</i>	
<i>Usage</i>	The latitude is measured in degrees, minutes and seconds, following by 'N' for the north and 'S' for the south of the equator (e.g. 48°51'29" N the Eiffel Tower latitude)
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates CardAcquisitionResponse.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates
<i>XMLCoding</i>	
<i>ASN1Coding</i>	355

Name	LeaveCardFlag
Definition	Indicates if the POI has to keep the card in the reader for a smart card.
References	
Usage	In the Card Acquisition request, leave the card in the reader to continue the transaction after the request of the Sale System. In the CardReaderInit request, if the card contains both magstripe and a chip, add a flag in the request to keep the card to with the smartcard after the magstripe reading.
Type	Boolean
Format	
CrossRef	CardAcquisitionRequest.TransactionData CardReaderInit
XMLCoding	Attribute
ASN1Coding	127

Name	LoggedSaleID
Definition	Sale Terminal logged to.
References	
Usage	
Type	TextString
Format	
CrossRef	DiagnosisResponse
XMLCoding	
ASN1Coding	130

Name	LoginRequest
<i>Definition</i>	Content of the Login Request message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the session (period between a Login and the following Logout) to process
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	LoginRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	128

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
DateTime	[1..1]		
SaleSoftware	[1..1]		
SaleTerminalData	[0..1]		Present if the login involve a Sale Terminal
TrainingModeFlag	[0..1]	default False	The POI does not realise the transaction with the Acquirer
OperatorLanguage	[1..1]		Default value for Device type displays
OperatorID	[0..1]		4 conditions to send it: a) the Sale System wants the POI log it in the transaction log b) because of reconciliation with total on OperatorID c) because the POI needs it d) acquirer or issuer need it
ShiftNumber	[0..1]		Same as OperatorID
TokenRequestedType	[0..1]		If a token is requested during the session.
CustomerOrderReq	[0..1]		If customer orders must be listed in Card Acquisition and Payment response messages during the session.
POISerialNumber	[0..1]		If the login involve a POI Terminal and not the first Login to the POI System

Name	LoginResponse
<i>Definition</i>	Content of the Login Response message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Login to process
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	LoginResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	129

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
POISystemData	[0..1]		if Response.Result is Success

Name	LogoutRequest
Definition	Content of the Logout Request message.
References	
Usage	Empty
Type	defined data structure
Format	
CrossRef	LogoutRequest
XMLCoding	
ASN1Coding	131

Component	Mult.	Constraint	Rule
MaintenanceAllowed	[0..1]	default False	

Name	LogoutResponse
Definition	Content of the Logout Response message.
References	
Usage	It conveys the result of the Logout.
Type	defined data structure
Format	
CrossRef	LogoutResponse
XMLCoding	
ASN1Coding	132

Component	Mult.	Constraint	Rule
Response	[1..1]		

Name	Longitude
Definition	Angular measurement of the distance of a location on the earth east or west of the Greenwich observatory.
References	
Usage	The longitude is measured in degrees, minutes and seconds, following by 'E' for the east and 'W' for the west (e.g. 23°27'30" E)
Type	TextString
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates CardAcquisitionResponse.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. GeographicCoordinates
XML Coding	
ASN1 Coding	354

Name	LoyaltyAccount
<i>Definition</i>	Data related to a loyalty account processed in the transaction.
<i>References</i>	
<i>Usage</i>	This data structure conveys the identification of the account and the associated loyalty brand.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.LoyaltyResult CardAcquisitionResponse PaymentResponse.LoyaltyResult LoyaltyResponse.LoyaltyResult
<i>XMLCoding</i>	
<i>ASN1Coding</i>	133

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
LoyaltyAccountID	[1..1]		
LoyaltyBrand	[0..1]		If a card is analysed

Name	LoyaltyAccountID
<i>Definition</i>	Identification of a Loyalty account.
<i>References</i>	
<i>Usage</i>	In the Payment or the Loyalty Request message, it allows to identify the loyalty account by the Sale Terminal instead of the POI Terminal (e.g. because the account identification is a bar-code read by the Cashier on a scanner device). In the Payment or the Loyalty Response message, it contains either the same data than the request if the loyalty account is provided by the Sale Terminal, either the loyalty account identifier read by the POI Terminal. In addition, the response contains the IdentificationSupport.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest.LoyaltyData BalanceInquiryResponse.LoyaltyResult.LoyaltyAccount CardAcquisitionResponse.LoyaltyAccount LoyaltyRequest.LoyaltyData LoyaltyResponse.LoyaltyResult.LoyaltyAccount PaymentRequest.LoyaltyData PaymentResponse.LoyaltyResult.LoyaltyAccount
<i>XMLCoding</i>	
<i>ASN1Coding</i>	134

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
EntryMode	[1..1]		
IdentificationType	[1..1]		
IdentificationSupport	[0..1]		if PaymentResponse or LoyaltResponse or BalanceInquiryResponse
LoyaltyID	[1..1]		

Name	LoyaltyAccountReq
<i>Definition</i>	Data related to a requested Loyalty program or account.
<i>References</i>	
<i>Usage</i>	In the Balance Inquiry Request message, the Sale Terminal sends the identification of the loyalty account to request the balance.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	135

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
CardAcquisitionReference	[0..1]		If the loyalty account ID comes from a previous CardAcquisition
LoyaltyAccountID	[0..1]		If loyalty identification of the loyalty account is realised by the Sale System

Name	LoyaltyAccountStatus
<i>Definition</i>	Data related to the result of a loyalty Balance Inquiry.
<i>References</i>	
<i>Usage</i>	In the Message Response, the result of each loyalty brand transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	136

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
LoyaltyAccount	[1..1]		
CurrentBalance	[0..1]		if known (provided by the card or an external host)
LoyaltyUnit	[0..1]	default Point	
Currency	[0..1]		If Result is Success and If LoyaltyUnit is "Monetary"

Name	LoyaltyAcquirerData
Definition	Data related to the loyalty Acquirer during a loyalty transaction.
References	
Usage	Information allowing the Sale System to characterise the transaction on the loyalty Acquirer host.
Type	defined data structure
Format	
CrossRef	PaymentResponse.LoyaltyResult LoyaltyResponse.LoyaltyResult
XMLCoding	
ASN1Coding	137

Component	Mult.	Constraint	Rule
LoyaltyAcquirerID	[0..1]		If available
ApprovalCode	[0..1]		If provided by the Acquirer
LoyaltyTransactionID	[0..1]		If provided by the Acquirer
HostReconciliationID	[0..1]		If provided by the Acquirer

Name	LoyaltyAcquirerID
Definition	Identification of the loyalty Acquirer.
References	
Usage	In case of multi loyalty Acquirers.
Type	TextString
Format	
CrossRef	PaymentResponse.LoyaltyResult.LoyaltyAcquirerData LoyaltyResponse.LoyaltyResult.LoyaltyAcquirerData
XMLCoding	Attribute
ASN1Coding	138

Name	LoyaltyAmount
<i>Definition</i>	Amount of a loyalty account.
<i>References</i>	
<i>Usage</i>	An awarded amount or an amount to redeem to the loyalty account might be sent in the Payment request message. The amount to apply on the requested loyalty service, if not computed from the TotalAmount of the Loyalty request message. The Payment or the Loyalty Response shall contain the redeemed amount, the awarded amount if known. In addition, the loyalty account balance is provided in the message response if available.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.LoyaltyData PaymentResponse.LoyaltyResult LoyaltyRequest.LoyaltyData LoyaltyResponse.LoyaltyResult
<i>XMLCoding</i>	
<i>ASN1Coding</i>	140

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
LoyaltyUnit	[0..1]	default Point	
Currency	[0..1]		if LoyaltyUnit is Monetary
AmountValue	[1..1]		

Name	LoyaltyBrand
<i>Definition</i>	Identification of a Loyalty brand.
<i>References</i>	
<i>Usage</i>	This is the brand of the loyalty known by the Sale System, which allows the level of detail necessary to the Sale System.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.LoyaltyResult CardAcquisitionResponse.LoyaltyAccount PaymentResponse.LoyaltyResult.LoyaltyAccount LoyaltyResponse.LoyaltyResult.LoyaltyAccount
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	141

Name	LoyaltyCurrency
<i>Definition</i>	Currency of a monetary amount.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	ISOCurrency3A
<i>Format</i>	
<i>CrossRef</i>	GetTotalsResponse.Totals.TransactionTotals ReconciliationResponse.Totals.TransactionTotals
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	142

Name	LoyaltyData
<i>Definition</i>	Data related to a Loyalty program or account.
<i>References</i>	
<i>Usage</i>	In the Payment, Loyalty or Balance Inquiry Request message, it allows the Sale Terminal to send the identification of the loyalty account or an awarded amount or an amount to redeem to the loyalty account.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	LoyaltyRequest PaymentRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	143

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
CardAcquisitionReference	[0..1]		If the loyalty account ID comes from a previous CardAcquisition
LoyaltyAccountID	[0..1]		If loyalty identification of the loyalty account is realised by the Sale System
LoyaltyAmount	[0..1]		When the Sale System want to award the Loyalty account (not for BalanceInquiryRequest)

Name	LoyaltyHandling
<i>Definition</i>	Type of Loyalty processing requested by the Sale System.
<i>References</i>	
<i>Usage</i>	An explicit way to specify what the POI has to handle concerning the loyalty.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionRequest.TransactionData PaymentRequest.TransactionData.TransactionConditions
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	145

<i>Label</i>	<i>Description</i>	<i>Code</i>
Forbidden	No loyalty card to read and loyalty transaction to process. Any attempt to enter a pure loyalty card is rejected.	
Processed	The loyalty transaction is already processed, no loyalty card or loyalty transaction to process.	
Allowed	The loyalty is accepted, but the POI has not to require or ask a loyalty card. The loyalty is involved by the payment card (e.g. an hybrid or linked card).	
Proposed	The loyalty is accepted, and the POI has to ask a loyalty card. If the Customer does not enter a loyalty card, no loyalty transaction is realised.	
Required	The loyalty is required, and the POI refuses the processing of the message request if the cardholder does not entre a loyalty card	

Name	LoyaltyID
<i>Definition</i>	Loyalty account identification.
<i>References</i>	
<i>Usage</i>	The identification of the loyalty account conforming to the IdentificationType.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest.LoyaltyData.LoyaltyAccountID BalanceInquiryResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID CardAcquisitionResponse.LoyaltyAccount.LoyaltyAccountID LoyaltyRequest.LoyaltyData.LoyaltyAccountID LoyaltyResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID PaymentRequest.LoyaltyData.LoyaltyAccountID PaymentResponse.LoyaltyResult.LoyaltyAccount.LoyaltyAccountID
<i>XMLCoding</i>	Enclosing
<i>ASN1Coding</i>	144

Name	LoyaltyRequest
<i>Definition</i>	Content of the Loyalty Request message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Loyalty transaction to process
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BatchRequest.TransactionToPerform LoyaltyRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	146

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
SaleData	[1..1]		
LoyaltyTransaction	[1..1]		
LoyaltyData	[0..n]		

Name	LoyaltyResponse
<i>Definition</i>	Content of the Loyalty Response message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Loyalty transaction processed by the POI System.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	LoyaltyResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	147

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
SaleData	[1..1]		
POIData	[1..1]		
LoyaltyResult	[0..n]		if loyalty account identified
PaymentReceipt	[0..*]		If Basic profile implementation with no printer on the POI.

Name	LoyaltyResult
<i>Definition</i>	Data related to the result of a processed loyalty transaction.
<i>References</i>	
<i>Usage</i>	In the Message Response, the result of each loyalty brand transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse LoyaltyResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	148

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
LoyaltyAccount	[1..1]		
CurrentBalance	[0..1]		if known (provided by the card or an external host)
LoyaltyAmount	[0..1]		If awarded amount
LoyaltyAcquirerData	[0..1]		if content not empty
Rebates	[0..1]		if rebates awarded

Name	LoyaltyTotals
<i>Definition</i>	Totals of the loyalty transaction during the reconciliation period.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	GetTotalsResponse.Totals.TransactionTotals ReconciliationResponse.Totals.TransactionTotals
<i>XMLCoding</i>	
<i>ASN1Coding</i>	149

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
TransactionType	[1..1]		Award, ReverseAward, Redemption, ReverseRedemption, Rebate, ReverseRebate
TransactionCount	[1..1]		
TransactionAmount	[1..1]		

Name	LoyaltyTransactionID
<i>Definition</i>	Identification of the Transaction for the Loyalty Acquirer.
<i>References</i>	
<i>Usage</i>	To identify the loyalty transaction on the Sale System.
<i>Type</i>	TransactionIdentificationType
<i>Format</i>	
<i>CrossRef</i>	LoyaltyResponse.LoyaltyResult.LoyaltyAcquirerData PaymentResponse.LoyaltyResult.LoyaltyAcquirerData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	139

Name	LoyaltyTransactionType
<i>Definition</i>	Type of loyalty transaction.
<i>References</i>	
<i>Usage</i>	These types of loyalty transaction use the same message requests.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	LoyaltyRequest.TransactionData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	150

<i>Label</i>	<i>Description</i>	<i>Code</i>
Award	Direct or payment related award on a loyalty account. Award alone, award associated to a payment (may be with an additional award alone), award because of a payment resulting on rebates.	
Rebate	Rebate on a total amount, sale item amount, or sale items	
Redemption	Redemption on a loyalty account.	
AwardRefund	Refund of a loyalty award transaction.	
RebateRefund	Refund of a loyalty rebate transaction.	
RedemptionRefund	Refund of a loyalty redemption transaction.	

Name	LoyaltyTransaction
<i>Definition</i>	Data related to the loyalty transaction.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	LoyaltyRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	151

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
LoyaltyTransactionType	[1..1]		
Currency	[0..1]		
TotalAmount	[0..1]		
OriginalPOITransaction	[0..1]		if LoyaltyTransactionType is "AwardRefund", "RebateRefund" or "RedemptionRefund"
TransactionConditions	[0..1]		If one data element is present
SaleItem	[0..n]		

Name	LoyaltyUnit
<i>Definition</i>	Unit of a loyalty amount.
<i>References</i>	
<i>Usage</i>	The amount could be expressed in point or in a monetary value and a currency.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.LoyaltyResult GetTotalsResponse.Totals.TransactionTotals PaymentRequest.LoyaltyData.LoyaltyAmount PaymentResponse.LoyaltyResult.LoyaltyAmount LoyaltyRequest.LoyaltyData.LoyaltyAmount LoyaltyResponse.LoyaltyResult.LoyaltyAmount ReconciliationResponse.Totals.TransactionTotals
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	152

<i>Label</i>	<i>Description</i>	<i>Code</i>
Point	The amount is expressed in point.	
Monetary	The amount is expressed in a monetary value in a currency.	

5.2.2.7 M-N

Name	MaintenanceRequiredFlag
Definition	Indicates if the occurred event requires maintenance call or action.
References	
Usage	
Type	Boolean
Format	
CrossRef	EventNotification
XMLCoding	Attribute
ASN1Coding	153

Name	MaintenanceAllowed
Definition	Indicates that the POI terminal is able to (or have to) go to maintenance.
References	
Usage	Sent in the Logout Request to express that after closing the session, the POI may go to maintenance.
Type	Boolean
Format	
CrossRef	LogoutRequest
XMLCoding	Attribute
ASN1Coding	387

Name	ManufacturerID
Definition	Identification of the Manufacturer
References	
Usage	Sent in the Login Request (resp. Response) to identify the Sale System (resp. POI System) manufacturer during the session.
Type	TextString
Format	
CrossRef	LoginRequest.SaleSoftware LoginResponse.POISystemData.POISoftware
XMLCoding	Attribute
ASN1Coding	154

Name	Markup
Definition	Markup of an amount in percentage.
References	
Usage	Markup of a currency conversion.
Type	Decimal
Format	
CrossRef	PaymentResponse.PaymentResult.CurrencyConversion
XMLCoding	Attribute
ASN1Coding	413

Name	MaskCharactersFlag
Definition	Indicates to mask the characters entered by the user (i.e. replacing in the display of the input, the entered character by a standard character as '**').
References	
Usage	During the processing of an Input command.
Type	Boolean
Format	
CrossRef	InputRequest.InpuData
XMLCoding	Attribute
ASN1Coding	405

Name	MaskedMSISDN
Definition	Masked Mobile Subscriber Integrated Service Digital Network.
References	
Usage	Beginning (country and National Destination Code) and end of the MSISDN, separated by the character '**'.
Type	DigitString
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData CardAcquisitionResponse.PaymentInstrumentData.MobileData PaymentRequest.PaymentData.PaymentInstrumentData.MobileData PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData
XMLCoding	Attribute
ASN1Coding	325

Name	MaskedPAN
Definition	Masked Primary Account Number
References	
Usage	Part of the PAN is replaced by a string of '*' characters, to identify a customer account or relationship. Presence of this data element, which replace the PAN when SensitiveCardData is protected and replaced by ProtectedCardData. Alternatively the MaskedPAN can be used as a token to identify a customer.
Type	TextString
Format	
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData CardAcquisitionResponse.PaymentInstrumentData.CardData PaymentRequest.PaymentData.PaymentInstrumentData.CardData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData
XMLCoding	Attribute
ASN1Coding	155

Name	MaxDecimalLength
Definition	Maximum input length of the decimal part (without decimal point)
References	
Usage	
Type	Integer
Format	
CrossRef	InputRequest.InpuData InputUpdate
XMLCoding	Attribute
ASN1Coding	156

Name	MaximumCashBackAmount
<i>Definition</i>	Maximum amount which could be requested for cash-back to the Sale System.
<i>References</i>	
<i>Usage</i>	Allows the Cashier to limit the amount value of cash-back to deliver to the Customer.
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.Amounts
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	168

Name	MaximumTransmitTime
<i>Definition</i>	Maximum time in seconds of transmission.
<i>References</i>	
<i>Usage</i>	Processing of a Transmit request message.
<i>Type</i>	Integer
<i>Format</i>	
<i>CrossRef</i>	TransmitRequest
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	388

Name	MaxInputTime
<i>Definition</i>	Maximum input time in seconds
<i>References</i>	
<i>Usage</i>	Limit of time to answer to an Input request message.
<i>Type</i>	Integer
<i>Format</i>	
<i>CrossRef</i>	InputRequest.InpuData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	157

Name	MaxLength
Definition	Maximum input length
References	
Usage	Maximum length of an entered string, or maximum number of entries that can be selected in a menu.
Type	Integer
Format	
CrossRef	InputRequest.InpuData InputUpdate
XMLCoding	Attribute
ASN1Coding	158

Name	MaxWaitingTime
Definition	Maximum time to wait for the request processing in seconds.
References	
Usage	If present, the receiver of the request has to answer before this waiting time to the sender of the request. It limits the waiting time for external event, (e.g. card insertion or removing by the customer)
Type	Integer
Format	
CrossRef	CardReaderInitRequest CardReaderPowerOffRequest PINRequest
XMLCoding	Attribute
ASN1Coding	317

Name	MenuBackFlag
Definition	If it has the value True, it indicates that the "Back" function key (respectively "Home" function key) may be used to go back to the immediate upper level of the menu (resp. to the home of the menu). If it has the value False, it indicates that the current menu level has no parent menu.
References	
Usage	During the processing of an Input command GetMenuEntry.
Type	Boolean
Format	
CrossRef	InputRequest.InpuData
XMLCoding	Attribute
ASN1Coding	391

Name	MenuEntry
<i>Definition</i>	An entry of the menu to present to the Cashier
<i>References</i>	
<i>Usage</i>	It conveys message text and parameters of the menu entry. This output data could be only provided for an input command, in order to choose an entry of the menu.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	InputRequest.Output InputUpdate
<i>XMLCoding</i>	
<i>ASN1Coding</i>	159

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
MenuEntryTag	[0..1]	default Selectable	
OutputFormat	[1..1]		MessageRef, Text, XHTML
DefaultSelectedFlag	[0..1]	default False	
PredefinedContent	[0..1]		Mandatory, if OutputFormat is MessageRef, not allowed otherwise.
OutputText	[0..n]		Mandatory, if OutputFormat is Text, not allowed otherwise. One instance of OutputText per shared format
OutputXHTML	[0..1]		Mandatory, if OutputFormat is XHTML, not allowed otherwise.

Name	MenuEntryNumber
<i>Definition</i>	The index from 1 to n, of the menu item which is selected by the Cashier on the Sale Terminal. The value -1 indicates that the immediate upper level of the menu is requested. The value 0 indicates that the root of the menu is requested.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Integer
<i>Format</i>	[1-]
<i>CrossRef</i>	InputResponse.InputResult.Input
<i>XMLCoding</i>	
<i>ASN1Coding</i>	160

Name	MenuEntryTag
Definition	Characteristics related to the selection of a menu entry
References	
Usage	.
Type	Enumeration
Format	
CrossRef	InputRequest.Output.MenuEntry
XMLCoding	Attribute
ASN1Coding	161

Label	Description	Code
Selectable	The Cashier may select this entry of the menu	
NonSelectable	The Cashier cannot select this entry of the menu	
SubMenu	The selection of this entry produces the display of a sub-menu (by the sending of another Input Request message containing the entries of this sub-menu).	
NonSelectableSubMenu	The menu entry is a submenu, but cannot be selected.	

Name	MerchantCategoryCode
Definition	The code which identifies the category of the transaction (MCC).
References	ISO 18245
Usage	This code usually the ISO code assigned by the Acquirer, but could contain additional information. A particular Merchant can have several MCC related to the type of services or goods it provides.
Type	TextString
Format	{3,4}
CrossRef	PaymentRequest.TransactionData.TransactionConditions
XMLCoding	Attribute
ASN1Coding	162

Name	MerchantID
<i>Definition</i>	Identification of the Merchant for the Acquirer
<i>References</i>	ISO 8583 - Elem. 42 – card acceptor identification code
<i>Usage</i>	Identification of the merchant for the Acquirer.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentAcquirerData PaymentResponse.PaymentResult.PaymentAcquirerData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	364

Name	MerchantOverrideFlag
<i>Definition</i>	Indicate that the Merchant forced the result of the payment to successfull.
<i>References</i>	
<i>Usage</i>	Allows the Sale System to be sure that the payment has been forced.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	163

Name	Message
<i>Definition</i>	Content of a transmitted message.
<i>References</i>	
<i>Usage</i>	Message to send or receive in a Transmit request or response message.
<i>Type</i>	ByteSequence
<i>Format</i>	
<i>CrossRef</i>	TransmitRequest TransmitResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	392

Name	MessageCategory
Definition	Category of message.
References	
Usage	Allow the recognition by the receiver of the message of its type. In a Reversal, identify the type of transaction to reverse (PaymentRequest or LoyaltyRequest) In a TransactionStatus, identify the last transaction message.
Type	Enumeration
Format	
CrossRef	MessageHeader InputUpdate.MessageReference ReversalRequest.OriginalPOITransaction TransactionStatusRequest.MessageReference AbortRequest.MessageReference
XMLCoding	Attribute
ASN1Coding	52

Label	Description	Code
Abort	Abort message request	
Admin	Admin request or response	
BalanceInquiry	Balance Inquiry request or response	
Batch	Batch request or response	
CardAcquisition	Card Acquisition request or response	
CardReaderAPDU	Card Reader APDU request or response	
CardReaderInit	Card Reader Init request or response	
CardReaderPowerOff	Card Reader Power-Off request or response	
Diagnosis	Diagnosis request or response	
Display	Display message request or response	
EnableService	Enable Service message request or response	
Event	Event Notification message	
GetTotals	GetTotals message request or response	
Input	Input message request or response	
InputUpdate	Input Update message	
Login	Login message request or response	
Logout	Logout message request or response	
Loyalty	Loyalty message request or response	
Payment	Payment message request or response	
PIN	PIN message request or response	
Print	Print message request or response	
Reconciliation	Reconciliation message request or response	
Reversal	Reversal message request or response	
Sound	Sound message request or response	
StoredValue	Stored Value message request or response	
TransactionStatus	TransactionStatus message request or response	
Transmit	Transmit message request or response	

Name	MessageClass
<i>Definition</i>	Class of the message
<i>References</i>	
<i>Usage</i>	Inform the receiver of the class of message to allow: - Switching of the messages before their processing - Direction of the message (Sale to POI or POI to Sale) - Flow inside the message belongs to
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	MessageHeader
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	164

Label	Description	Code
Service	A transaction message pair initiated by the Sale System, and requested to the POI System.	
Device	A device message pair either: Inside a Service request and response. This device message pair is initiated by the POI System, and sent to Sale System, either Outside a Service request and response. This device message pair is initiated by the Sale System and sent to POI System.	
Event	An unsolicited event notification by the POI System to the Sale System.	

Name	MessageHeader
<i>Definition</i>	Message header of the Sale to POI protocol message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Sale to POI protocol management
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	all messages
<i>XMLCoding</i>	
<i>ASN1Coding</i>	165

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
ProtocolVersion	[0..1]		If MessageCategory is Login or Diagnosis
MessageClass	[1..1]		
MessageCategory	[1..1]		
MessageType	[1..1]		
ServiceID	[0..1]		If "Service" or "Event" MessageClass message. If "Device" MessageClass, and request from POI or response from Sale.
DeviceID	[0..1]		If "Device" MessageClass
SaleID	[1..1]		
POIID	[1..1]		

Name	MessageReference
<i>Definition</i>	Identification of a previous POI transaction.
<i>References</i>	
<i>Usage</i>	To abort a transaction in progress or to request the status of a transaction from which no response has been received. It identifies the message header of the message request to abort or request the status. The Abort or TransactionStatus could be sent from another Sale Terminal (SaleID) or to another POI Terminal (POIID) than the original message request.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	AbortRequest InputUpdate TransactionStatusRequest TransactionStatusResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	166

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
MessageCategory	[0..1]		Payment, Loyalty, StoredValue, CardAcquisition, Batch, Reconciliation, Display, Input, Print, CardReaderAPDU, CardReaderInit, CardReaderPowerOff
ServiceID	[0..1]		
DeviceID	[0..1]		
SaleID	[0..1]		default MessageHeader.SaleID
POIID	[0..1]		default MessageHeader.POID

Name	MessageType
Definition	Type of message of the Sale to POI protocol
References	
Usage	Inform the receiver of the category of message to allow: - Switching of the messages before their processing - Direction of the message (Sale to POI or POI to Sale) - Flow inside the message belongs to
Type	Enumeration
Format	
CrossRef	MessageHeader
XMLCoding	Attribute
ASN1Coding	167

Label	Description	Code
Request	Request message that requires a response, except if the request message mentions explicitly that a response message is not expected.	
Response	Response message, sent to answer to a request message.	
Notification	Unsolicited notification message that does not require an answer.	

Name	MinimumAmountToDeliver
<i>Definition</i>	Minimum amount the Sale System is allowed to deliver for this payment.
<i>References</i>	
<i>Usage</i>	For the OneTimeReservation, when the maximum amount is unknown, the Sale System indicates the minimum amount it allows.
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.Amounts
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	169

Name	MinimumDisplayTime
<i>Definition</i>	Number of seconds the message has to be displayed
<i>References</i>	
<i>Usage</i>	In Display request, to be sure that the Cashier or the Cardholder can see the displayed message. When the value is 0, a new message can be displayed immediately.
<i>Type</i>	Integer
<i>Format</i>	[0-999]
<i>CrossRef</i>	DisplayRequest.Output
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	170

Name	MinimumSplitAmount
Definition	Minimum amount of a split, which could be requested by a Customer.
References	
Usage	Allows the Merchant to limit the number of split requested by the Customer.
Type	SimpleAmountType
Format	
CrossRef	PaymentRequest.TransactionData.Amounts
XMLCoding	Attribute
ASN1Coding	171

Name	MinLength
Definition	Minimum input length
References	
Usage	Minimum length of an entered string, or minimum number of entries that can be selected in a menu.
Type	Integer
Format	
CrossRef	InputRequest.InpuData InputUpdate
XMLCoding	Attribute
ASN1Coding	172

Name	MobileCountryCode
Definition	Identifies the country of a mobile phone operator.
References	ITU-T E.212
Usage	
Type	DigitString
Format	{3,3}
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData CardAcquisitionResponse.PaymentInstrumentData.MobileData PaymentRequest.PaymentData.PaymentInstrumentData.MobileData PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData
XMLCoding	
ASN1Coding	323

Name	MobileData
<i>Definition</i>	Information related to the mobile for the payment transaction.
<i>References</i>	
<i>Usage</i>	Mobile phone is used as a payment instrument for the transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData BalanceInquiryRequest.PaymentData.PaymentInstrumentData CardAcquisitionResponse.PaymentInstrumentData PaymentRequest.PaymentData.PaymentInstrumentData PaymentResponse.PaymentResult.PaymentInstrumentData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	322

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
MobileCountryCode	[0..1]		If data available
MobileNetworkCode	[0..1]		If data available
MaskedMSISDN	[0..1]		If data available
Geolocation	[0..1]		If data available
ProtectedMobileData	[0..1]		SensitiveMobileData protected by CMS EnvelopedData
SensitiveMobileData	[0..1]		If unprotected mobile data

Name	MobileNetworkCode
<i>Definition</i>	Identifies the mobile phone operator inside a country.
<i>References</i>	ITU-T E.212
<i>Usage</i>	Used with a Mobile Country Code to uniquely identify a mobile phone operator.
<i>Type</i>	DigitString
<i>Format</i>	{2,3}
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData CardAcquisitionResponse.PaymentInstrumentData.MobileData PaymentRequest.PaymentData.PaymentInstrumentData.MobileData PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	324

Name	MSISDN
Definition	Mobile Subscriber Integrated Service Digital Network (i.e. mobile phone number of the SIM card).
References	
Usage	Country, National Destination Code, and Subscriber Number.
Type	DigitString
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.SensitiveMobileData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.SensitiveMobileData CardAcquisitionResponse.PaymentInstrumentData.MobileData.SensitiveMobileData PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.SensitiveMobileData PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.SensitiveMobileData
XMLCoding	Attribute
ASN1Coding	329

Name	NotifyCardInputFlag
Definition	Request Notification of card entered in the POI card reader.
References	
Usage	During the processing of an Input Device request, if the Customer enters a card in the card reader without answering to the Input command, the POI abort the Input command processing, and answer a dedicated ErrorCondition (Aborted) value in the Input response message.
Type	Boolean
Format	
CrossRef	InputRequest.InpuData
XMLCoding	Attribute
ASN1Coding	173

Name	Number
Definition	Number of elements
References	
Usage	Number of coins or bills of a certain value.
Type	Integer
Format	
CrossRef	DiagnosisResponse.POIStatus.CashHandlingDevice.CoinsOrBills LoginResponse.POIStatus.CashHandlingDevice.CoinsOrBills
XMLCoding	Attribute
ASN1Coding	365

5.2.2.8 O

Name	OnlineFlag
Definition	Indicate that the payment transaction processing has required the approval of a host.
References	
Usage	Allows the Sale System to know if the payment was online or offline.
Type	Boolean
Format	
CrossRef	PaymentResponse.PaymentResult
XMLCoding	Attribute
ASN1Coding	349

Name	OpenOrderState
Definition	Specifies if a customer order is currently Open. An open customer order is an order waiting for further operations.
References	
Usage	
Type	Boolean
Format	
CrossRef	CardAcquisitionResponse.CustomerOrder PaymentRequest.PaymentData.CustomerOrder PaymentResponse.CustomerOrder ReversalRequest.CustomerOrder ReversalResponse.CustomerOrder
XMLCoding	Attribute
ASN1Coding	417

Name	OperatorID
Definition	Identification of the Cashier or Operator.
References	
Usage	Sent in the Login Request for information only, to identify the Cashier that drives the Sale Terminal during the session. This identification could be sent in other message request if the Cashier identification has changed after the Login.
Type	TextString
Format	
CrossRef	CardAcquisitionRequest.SaleData GetTotalsRequest.TotalFilter GetTotalsResponse.Totals.TransactionTotals LoginRequest LoyaltyRequest.SaleData PaymentRequest.SaleData ReconciliationResponse.Totals.TransactionTotals StoredValueRequest.SaleData
XMLCoding	Attribute

Name	OperatorLanguage
Definition	Language of the Cashier or Operator.
References	
Usage	Allows choice of the Cashier language when the POI displays messages or print text to Merchant interface. The Login Request sends the default Cashier language, it could be sent in other message request if the Cashier language has changed after the Login.
Type	ISOLanguage2A
Format	
CrossRef	CardAcquisitionRequest.SaleData LoginRequest LoyaltyRequest.SaleData PaymentRequest.SaleData StoredValueRequest.SaleData
XMLCoding	Attribute
ASN1Coding	64

Name	OriginalPOITransaction
Definition	Identification of a previous POI transaction.
References	
Usage	<p>In the Payment or the Loyalty Request message, it allows using the card of a previous CardAcquisition or Payment/Loyalty request.</p> <p>To reverse a Payment or the Loyalty transaction.</p> <p>By default, the reversal is requested from the same Sale Terminal to the same POI Terminal than the original transaction.</p> <p>But the reversal could be requested by a different Sale Terminal (or the Sale System) to a different POI Terminal (or the POI System).</p>
Type	defined data structure
Format	
CrossRef	LoyaltyRequest.LoyaltyData PaymentRequest.PaymentTransaction ReversalRequest ReversalResponse StoredValueRequest.StoredValueData
XML Coding	
ASN1 Coding	174

Component	Mult.	Constraint	Rule
SaleID	[0..1]		
POIID	[0..1]		If original transaction is coming from another POI
POITransactionID	[0..1]		Absent if SaleReferenceID is sufficient to identify the transaction, or for some reversal cases.
ReuseCardDataFlag	[0..1]	default True	
ApprovalCode	[0..1]		If referral
CustomerLanguage	[0..1]		Optional for Reversal, otherwise absent.
AcquirerID	[0..1]		Optional for Reversal, otherwise absent.
AmountValue	[0..1]		Optional for Reversal, otherwise absent.
HostTransactionID	[0..1]		If POITransactionID not present

Name	OutputBarcode
<i>Definition</i>	BarCode content to display or print.
<i>References</i>	
<i>Usage</i>	Various usage of barcode
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement EnableServiceRequest.Output.OutputElement InputRequest.Output.OutputElement InputUpdate.Output PrintRequest.Output.OutputElement CardReaderInitRequest.Output.OutputElement CardReaderAPDUREquest.Output.OutputElement CardReaderPowerOffRequest.Output.OutputElement
<i>XMLCoding</i>	
<i>ASN1Coding</i>	175

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
BarcodeType	[0..1]	default EAN13	
BarcodeValue	[1..1]		

Name	OutputContent
Definition	Content to display or print.
References	
Usage	This is a sequence of elements if they have different formats.
Type	defined data structure
Format	
CrossRef	CardReaderInitRequest.Output CardReaderAPDUREquest.Output CardReaderPowerOffRequest.Output DisplayRequest.Output EnableServiceRequest.Output InputRequest.Output InputUpdate PaymentResponse.PaymentReceipt PrintRequest.Output
XMLCoding	
ASN1Coding	176

Component	Mult.	Constraint	Rule
OutputFormat	[1..1]		
PredefinedContent	[0..1]		Mandatory, if OutputFormat is MessageRef, not allowed otherwise.
OutputText	[0..n]		Mandatory, if OutputFormat is Text, not allowed otherwise. One instance of OutputText per shared format
OutputXHTML	[0..1]		Mandatory, if OutputFormat is XHTML, not allowed otherwise.
OutputBarcode	[0..1]		Mandatory, if OutputFormat is BarCode, not allowed otherwise.

Name	OutputFormat
Definition	Format of the content to display or print
References	
Usage	Display or print device function
Type	Enumeration
Format	
CrossRef	CardReaderInitRequest.Output.OutputElement CardReaderAPDUREquest.Output.OutputElement CardReaderPowerOffRequest.Output.OutputElement DisplayRequest.Output.OutputElement EnableServiceRequest.Output.OutputElement InputRequest.Output InputRequest.MenuEntry InputUpdate.Output PaymentResponse.PaymentReceipt.OutputElement PrintRequest.Output.OutputElement
XMLCoding	Attribute
ASN1Coding	177

Label	Description	Code
MessageRef	Predefined message (of any format) on the POI or the Sale. The output is then a PredefinedContent data structure.	
Text	Text message including control characters prefixed by an escape character. The DisplayOutput is then an OutputText data structure.	
XHTML	DisplayOutput uses the eXtensible HyperText Markup Language.	
BarCode	Barcode type to print The output is then a OutputBarCode data structure.	

Name	OutputResult
Definition	Information related to the result the output (display, print, input).
References	
Usage	In the message response, it contains the result of the output, if required in the message request.
Type	defined data structure
Format	
CrossRef	DisplayResponse PrintResponse InputResponse
XMLCoding	
ASN1Coding	179

Component	Mult.	Constraint	Rule
Device	[1..1]		Copy
InfoQualify	[1..1]		Copy
Response	[1..1]		

Name	OutputSignature
Definition	Vendor specific signature of text message to display or print.
References	
Usage	
Type	ByteSequence
Format	
CrossRef	CardReaderInitRequest.Output.OutputElement CardReaderAPDUREquest.Output.OutputElement CardReaderPowerOffRequest.Output.OutputElement DisplayRequest.Output.OutputElement EnableServiceRequest.Output.OutputElement InputRequest.Output.OutputElement InputUpdate PrintRequest.Output.OutputElement
XMLCoding	
ASN1Coding	180

Name	OutputText
<i>Definition</i>	Content of text message to display or print.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the content of the text message and its format. All the data elements related to the format of the text to display or print are parameters valid for the whole Text content.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardReaderInitRequest.Output.OutputElement CardReaderAPDUREquest.Output.OutputElement CardReaderPowerOffRequest.Output.OutputElement DisplayRequest.Output.OutputElement EnableServiceRequest.Output.OutputElement InputRequest.Output InputRequest.Output.MenuEntry.PredefinedContent InputUpdate.Output PaymentResponse.PaymentReceipt.OutputElement PrintRequest.Output.OutputElement
<i>XMLCoding</i>	
<i>ASN1Coding</i>	181

Component	Mult.	Constraint	Rule
Text	[1..1]		
CharacterSet	[0..1]		If not present, the settings of the target system or device are used.
Font	[0..1]		If not present, the settings of the target system or device are used.
StartRow	[0..1]		If not present, the settings of the target system or device are used (e.g. current row position).
StartColumn	[0..1]		If not present, the settings of the target system or device are used (e.g. current column position).
Color	[0..1]		If not present, default colour used
CharacterWidth	[0..1]		If not present, default width used
CharacterHeight	[0..1]		If not present, default height used
CharacterStyle	[0..1]		If not present, default style used
Alignment	[0..1]		If not present, default alignment used
EndOfLineFlag	[0..1]	default True	

Name	OutputXHTML
Definition	XHTML document body containing the message to display or print.
References	
Usage	<p>It conveys Information related to the content of the message and its format using the eXtensible HyperText Markup Language (XHTML).</p> <p>The document can contain the following elements:</p> <ul style="list-style-type: none"> - For the character encoding: the charset attribute in a meta data element for the http-equiv attribute equal to "Content-Type" - For the colour: the color namespace values of the style attribute. - For the character size: the big or small data elements. - For the character style: the b, i or tt data elements - For the text alignment: the text-align namespace values of the style attribute. - For the end of line: the p data element. <p>The JPEG image format could be displayed with this markup language.</p> <p>To avoid problem of XML/schema validation with the XHTML tags, the type of this data element is ByteSequence, involving a base 64 conversion of the XHTML text.</p>
Type	ByteSequence
Format	
CrossRef	DiplayRequest.Output.OutputElement EnableServiceRequest.Output.OutputElement InputRequest.Output InputRequest.Output.MenuEntry.PredefinedContent InputUpdate.Output PaymentResponse.PaymentReceipt.OutputElement PrintRequest.Output.OutputElement CardReaderInitRequest.Output.OutputElement CardReaderAPDUREquest.Output.OutputElement CardReaderPowerOffRequest.Output.OutputElement
XMLCoding	
ASN1Coding	178

Name	OwnerName
Definition	Owner name of an account
References	
Usage	Name of the owner of a stored value account.
Type	TextString
Format	
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID CardAcquisitionResponse.PaymentInstrumentData.StoredValueAccountID PaymentRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID PaymentResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID StoredValueRequest.StoredValueData.StoredValueAccountID StoredValueResponse.StoredValueResult.StoredValueAccountID
XMLCoding	Attribute
ASN1Coding	352

5.2.2.9 P

Name	PaidAmount
Definition	Amount already paid amount in case of split payment.
References	
Usage	Depending on the context, a split payment is either a split amount, either a split basket (required by some payment means as fleet cards). The PaidAmount is present when the split payment is a split of the amount. Split of the basket involves two Sale Transactions, and has not to be recognised by the POI. Split payment is forbidden for reservations.
Type	SimpleAmountType
Format	
CrossRef	PaymentRequest.TransactionData.Amounts
XMLCoding	Attribute
ASN1Coding	182

Name	PAN
Definition	Primary Account Number
References	ISO 8583 - Elem. 2 - primary account number
Usage	Identify a customer account or relationship
Type	DigitString
Format	{8,28}
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData CardAcquisitionResponse.PaymentInstrumentData.CardData.SensitiveCardData PaymentRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData
XMLCoding	Attribute
ASN1Coding	183

Name	Password
Definition	A text password which is typed by the Customer on the POI or the Cashier on the Sale Terminal.
References	
Usage	
Type	ContentInformationType
Format	
CrossRef	InputResponse.InputResult.Input
XMLCoding	
ASN1Coding	313

Name	PaymentAccountRef
<i>Definition</i>	Reference of the PAN, which identifies the PAN or the card uniquely, named also PAR (Payment Account Reference). This reference may be defined by the card issuer or by a token service provider under the control of the card issuer, and cannot be used for a payment transaction.
<i>References</i>	
<i>Usage</i>	It allows, for a merchant, to identify the card which is used for a payment, whatever the card or a token is used for the payment transaction.
<i>Type</i>	Text
<i>Format</i>	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData CardAcquisitionResponse.PaymentInstrumentData.CardData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData
<i>CrossRef</i>	InputResponse.InputResult.Input
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	406

Name	PaymentAccountReq
<i>Definition</i>	Data related to the account pointed by the payment card
<i>References</i>	
<i>Usage</i>	Information provided by the Sale System related to the payment account requesting a balance.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	184

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
AccountType	[0..1]	default Default	
CardAcquisitionReference	[0..1]		if the card data comes from a previous CardAcquisition
PaymentInstrumentData	[0..1]		

Name	PaymentAccountStatus
<i>Definition</i>	Data related to the result of a Balance Inquiry request.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	185

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
PaymentInstrumentData	[0..1]		If a payment instrument is analysed
Currency	[0..1]		If PaymentInstrumentData present and Result is Success
CurrentBalance	[0..1]		If PaymentInstrumentData present and Result is Success
PaymentAcquirerData	[0..1]		If a card is analysed
LoyaltyAccountStatus	[0..1]		If PaymentInstrumentData absent and Result is Success

Name	PaymentAcquirerData
<i>Definition</i>	Data related to the response from the payment Acquirer.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult PaymentResponse.PaymentResult
<i>XMLCoding</i>	
<i>ASN1Coding</i>	186

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
AcquirerID	[0..1]		If several Acquirers
MerchantID	[1..1]		
AcquirerPOIID	[1..1]		
AcquirerTransactionID	[0..1]		If provided by the Acquirer
ApprovalCode	[0..1]		If available

Name	PaymentBrand
Definition	Type of payment card
References	
Usage	Indicates the card used to pay in the PaymentResponse. Sent in the CardAcquisitionResponse, to leave the Cashier to choose between several applications in a smartcard, or several brand in a co-branded card. In this case, the CardAcquisitionRequest.ForceCustomerSelectionFlag must contain the value False. Brands are part of the POI and Sale Systems configurations.
Type	TextString
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData CardAcquisitionResponse PaymentResponse.PaymentResult.PaymentInstrumentData.CardData
XMLCoding	Attribute
ASN1Coding	187

Name	PaymentCurrency
Definition	Currency of a monetary amount.
References	
Usage	
Type	ISOCurrency3A
Format	
CrossRef	GetTotalsResponse.Totals.TransactionTotals ReconciliationResponse.Totals.TransactionTotals
XMLCoding	Attribute
ASN1Coding	188

Name	PaymentData
<i>Definition</i>	Data related to the payment transaction.
<i>References</i>	
<i>Usage</i>	Elements requested by the Sale System that are related to the payment only.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	189

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
PaymentType	[0..1]	default Normal	
SplitPaymentFlag	[0..1]	default False	
CardAcquisitionReference	[0..1]		if the card data comes from a previous CardAcquisition
RequestedValidityDate	[0..1]		If time period of the OneTimeReservation, FirstReservation or UpdateReservation is requested
Instalment	[0..1]		If PaymentType is "Instalment" or "IssuerInstalment"
CustomerOrder	[0..1]		If a customer orders has to be created.
PaymentInstrumentData	[0..1]		If payment instrument data are read by the Sale System

Name	PaymentInstrumentData
<i>Definition</i>	Data related to the instrument of payment for the transaction.
<i>References</i>	
<i>Usage</i>	<p>Sent in the result of the payment transaction. For a card, it could also be sent in the CardAcquisition response, to be processed by the Sale System. In this case, the card or type of card has to be configured to have this behaviour. It is then expected that for this card the information sent in response to the payment are the same for the CardAcquisition response.</p> <p>Data that could be protected in the response are grouped in a dedicated data structure. These data could be sent in the request, if the payment instrument data are read by the Sale system.</p>
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult BalanceInquiryRequest.PaymentData CardAcquisitionResponse PaymentRequest.PaymentData PaymentResponse.PaymentResult
<i>XMLCoding</i>	
<i>ASN1Coding</i>	320

Component	Mult.	Constraint	Rule
PaymentInstrumentType	[1..1]		
CardData	[0..1]		If PaymentInstrumentType is "Card"
CheckData	[0..1]		If PaymentInstrumentType is "Check"
MobileData	[0..1]		If PaymentInstrumentType is "Mobile"

Name	PaymentInstrumentType
<i>Definition</i>	Type of payment instrument.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData BalanceInquiryRequest.PaymentData.PaymentInstrumentData CardAcquisitionResponse.PaymentInstrumentData PaymentRequest.PaymentData.PaymentInstrumentData PaymentResponse.PaymentResult.PaymentInstrumentData ReconciliationResponse.Totals.TransactionTotals
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	321

Label	Description	Code
Card	Payment card (credit or debit).	
Check	Paper check.	
Mobile	Operator account accessed by a mobile phone.	
StoredValue	Account accessed by a stored value instrument such as a card or a certificate.	
Cash	Cash managed by a cash handling system.	

Name	PaymentReceipt
Definition	Customer or Merchant payment receipt.
References	
Usage	If the payment receipts are printed by the Sale system and the POI or the Sale does not implement the Print exchange (Basic profile).
Type	defined data structure
Format	
CrossRef	PaymentResponse
XMLCoding	
ASN1Coding	345

Component	Mult.	Constraint	Rule
DocumentQualifier	[1..1]		SaleReceipt or CashierReceipt
IntegratedPrintFlag	[0..1]	default False	
RequiredSignatureFlag	[0..1]	default False	
OutputContent	[1..1]		

Name	PaymentRequest
<i>Definition</i>	Content of the Payment Request message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Payment transaction to process
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BatchRequest.TransactionToPerform PaymentRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	190

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
SaleData	[1..1]		
PaymentTransaction	[1..1]		
PaymentData	[0..1]		If one data element is present
LoyaltyData	[0..n]		Loyalty cards used with the payment transaction and read by the Sale System

Name	PaymentResponse
<i>Definition</i>	Content of the Payment Response message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Payment transaction processed by the POI System
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	191

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
SaleData	[1..1]		Copy
POIData	[1..1]		
PaymentResult	[0..1]		If one data element is present
LoyaltyResult	[0..n]		Loyalty cards used with the payment transaction
PaymentReceipt	[0..n]		If Basic profile implementation with no printer on the POI.
CustomerOrder	[0..n]		If a customer orders was created.

Name	PaymentResult
<i>Definition</i>	Data related to the result of a processed payment transaction.
<i>References</i>	
<i>Usage</i>	In the Message Response, the result of card payment transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	192

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
PaymentType	[0..1]	default Normal	Copy
PaymentInstrumentData	[0..1]		If a payment instrument is analysed by the POI
AmountsResp	[0..1]		If Result is Success or Partial
Instalment	[0..1]		Absent if PaymentType is not "IssuerInstalment"
CurrencyConversion	[0..n]		If currency conversion the Sale needs to know
MerchantOverrideFlag	[0..1]	default False	If payment forced by the Cashier
CapturedSignature	[0..1]		If handwritten signature is captured on the POI by a signature capture device.
ProtectedSignature	[0..1]		Encrypted handwritten signature captured on the POI by a signature capture device.
CustomerLanguage	[0..1]		If the customer language is different from the default language or different from the CustomerLanguage of the PaymentRequest if any.
OnlineFlag	[0..1]	default True	"True" if the payment transaction processing has required the approval of a host.
AuthenticationMethod	[0..1]		Method for customer authentication.
ValidityDate	[0..1]		if OneTimeReservation, FirstReservation or UpdateReservation
PaymentAcquirerData	[0..1]		If a card is analysed and data available

Name	PaymentToken
<i>Definition</i>	Surrogate of the PAN (Primary Account Number) of the payment card to identify the payment mean of the customer.
<i>References</i>	
<i>Usage</i>	It allows, for a merchant, to identify the customer.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.PaymentInstrumentData.CardData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	407

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
TokenRequestedType	[1..1]		
TokenValue	[1..1]		
ExpiryDateTime	[0..1]		

Name	PaymentTotals
<i>Definition</i>	Totals of the payment transaction during the reconciliation period.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	GetTotalsResponse.Totals.TransactionTotals ReconciliationResponse.Totals.TransactionTotals
<i>XMLCoding</i>	
<i>ASN1Coding</i>	193

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
TransactionType	[1..1]		Debit, Credit, ReverseDebit, ReverseCredit, OneTimeReservation, CompletedDeffered, FirstReservation, UpdateReservation, CompletedReservation, CashAdvance
TransactionCount	[1..1]		
TransactionAmount	[1..1]		

Name	PaymentTransaction
<i>Definition</i>	Data related to the payment and loyalty transaction.
<i>References</i>	
<i>Usage</i>	Elements requested by the Sale System that are global to the payment or loyalty transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	194

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
AmountsReq	[1..1]		
OriginalPOITransaction	[0..1]		if UpdateReservation, Completion or Refund
TransactionConditions	[0..1]		If one data element is present
SaleItem	[0..n]		If purchased products are required for the payment

Name	PaymentType
<i>Definition</i>	Type of payment transaction.
<i>References</i>	
<i>Usage</i>	Elements requested by the Sale System that are related to the payment only.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.PaymentData PaymentResponse.PaymentResult
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	195

<i>Label</i>	<i>Description</i>	<i>Code</i>
Normal	Normal Payment	
Refund	Payment refund	
OneTimeReservation	One time reservation to be just followed by a completion when the service or good is delivered. This service is sometimes called "Deferred Sale".	
FirstReservation	First reservation for an amount and period of time. This service is sometimes called "Pre-Authorisation".	
UpdateReservation	Adjustment of the amount or period of time of a reservation.	
Completion	End of the reservation transaction.	
CashAdvance	Cash advance at the POI System.	
CashDeposit	Cash deposit at the POI System, to credit an account.	
Recurring	Recurring payment.	
Instalment	Instalments of payment performed on behalf of the merchant.	
IssuerInstalment	Instalments of payment performed by the card issuer.	
PaidOut	Give money to in return for goods or services rendered to the merchant.	

Name	PEDOKFlag
<i>Definition</i>	Indicates if the PED is working and usable.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.POIStatus LoginResponse.POIStatus
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	196

Name	PerformedTransaction
<i>Definition</i>	Result of performed transactions.
<i>References</i>	
<i>Usage</i>	Contains result of transaction performed without the Sale system.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BatchResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	381

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
SaleData	[0..1]		If a request has been generated by the Sale system.
POIData	[1..1]		
PaymentResult	[0..1]		If a Payment transaction has been performed and one (or several) data element is present
LoyaltyResult	[0..n]		If a Loyalty transaction has been performed alone or with the Payment transaction.
ReversedAmount	[0..1]		If a transaction Reversal has been performed.

Name	Period
Definition	Period of time with defined unit of time.
References	
Usage	A period between 2 payment instalments.
Type	Integer
Format	
CrossRef	PaymentRequest.PaymentData.Instalment
XMLCoding	Attribute
ASN1Coding	371

Name	PeriodUnit
Definition	Type of instalment transaction.
References	
Usage	
Type	Enumeration
Format	
CrossRef	PaymentRequest.PaymentData.Instalment
XMLCoding	Attribute
ASN1Coding	372

Label	Description	Code
Daily	The day is the unit of the period.	
Weekly	The week is the unit of the period.	
Monthly	The month is the unit of the period.	
Annual	The year is the unit of the period.	

Name	PINEncAlgorithm
Definition	Identify the encrypted PIN block algorithm.
References	
Usage	When the encrypted PIN is provided as an input or an output of the PIN service. There are no predefined values for this identification.
Type	DigitString
Format	
CrossRef	PINRequest.PINRequest PINRequest.CardholderPIN PINResponse.CardholderPIN
XMLCoding	Attribute
ASN1Coding	197

Name	PINFormat
Definition	Identify the format of the PIN before encryption.
References	
Usage	When the encrypted PIN is provided as an input or an output of the PIN service.
Type	Enumeration
Format	
CrossRef	PINRequest.PINRequest PINRequest.CardholderPIN PINResponse.CardholderPIN
XMLCoding	Attribute
ASN1Coding	198

Label	Description	Code
ISO0	ISO 0	
ISO1	ISO 1	
ISO2	ISO 2	
ISO3	ISO 3	

Name	PINRequest
<i>Definition</i>	Content of the PIN Request message.
<i>References</i>	
<i>Usage</i>	It contains the type of request related to the PIN, and the associated parameters.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PINRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	199

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
PINRequestType	[1..1]		
PINVerifyMethod	[0..1]		Optional if PINRequestType is PINVerify or PINVerifyOnly, absent otherwise
AdditionalInput	[0..1]		Optional if PINRequestType is PINEnter or PINVerify, absent otherwise
PINEncAlgorithm	[0..1]		Optional if PINRequestType is PINEnter, absent otherwise
PINFormat	[0..1]		Optional if PINRequestType is PINEnter, absent otherwise
KeyReference	[0..1]		Optional if PINRequestType is PINEnter, absent otherwise
MaxWaitingTime	[0..1]		Absent if PINRequestType is PINVerify or PINVerifyOnly, optional if PINRequestType is PINEnter
CardholderPIN	[0..1]		Optional if PINRequestType is PINVerifyOnly, absent otherwise <i>Conformed to EPAS Acquirer protocol.</i>

Name	PINRequestType
<i>Definition</i>	Type of PIN Service.
<i>References</i>	
<i>Usage</i>	To request PIN services to the POI.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	PINRequest.PINRequest
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	200

<i>Label</i>	<i>Description</i>	<i>Code</i>
PINVerify	The PIN Entering and Verify.	
PINVerifyOnly	The PIN Verify only, the PIN is entered before and the PIN Block (encrypted PIN) is provided.	
PINEnter	The PIN is entered by the Cardholder, encrypted by the POI, and provided as a result.	

Name	PINResponse
Definition	Content of the PIN Response message.
References	
Usage	It contains the result of the requested service, with possibly the encrypted PIN.
Type	defined data structure
Format	
CrossRef	PINResponse
XMLCoding	
ASN1Coding	201

Component	Mult.	Constraint	Rule
Response	[1..1]		
CardholderPIN	[0..1]		

Name	PINVerifMethod
Definition	Identify the PIN verification method and keys.
References	
Usage	To verify the PIN in the PIN request. There are no predefined values for this identification.
Type	TextString
Format	
CrossRef	PINRequest.PINRequest
XMLCoding	Attribute
ASN1Coding	202

Name	PlanID
Definition	Identification of an instalment plan.
References	
Usage	
Type	TextString
Format	
CrossRef	PaymentRequest.PaymentData.Instalment
XMLCoding	Attribute
ASN1Coding	370

Name	POICapabilities
<i>Definition</i>	Hardware capabilities of the POI Terminal.
<i>References</i>	
<i>Usage</i>	<p>They are the POI devices the POI System allows the Sale System uses. These POI devices include the logical devices enumerated in the Devices data element (most of these devices could be on the POI part), plus card reader devices. Sent in the Login Response to identify the devices of the POI Terminal which can be used by the Sale System during the session.</p>
<i>Type</i>	Cluster
<i>Format</i>	
<i>CrossRef</i>	LoginResponse.POISystemData.POITerminalData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	203

<i>Label</i>	<i>Description</i>	<i>Code</i>
CashierDisplay	Used by the Sale System when the device is managed by the POI Terminal, to ask question or display some information to the Cashier.	
CashierError	To display to the Cashier information is related to an error situation occurring on the Sale Terminal when the device is managed by the POI Terminal.	
CashierInput	Any kind of keyboard allowing all or part of the commands of the Input message request from the POI System to the Sale System (InputCommand data element). The output device attached to this input device is the CashierDisplay device.	
CustomerDisplay	Standard Customer display interface used by the Sale System to ask question, or to show information to the Customer outside a Service dialogue.	
CustomerError	To display to the Customer information is related to an error situation occurring on the Sale Terminal during a Sale transaction.	
CustomerInput	Any kind of keyboard allowing all or part of the commands of the Input message request from the POI System to the Sale System (InputCommand data element). The output device attached to this input device is the CashierDisplay device.	
PrinterReceipt	Printer for the Sale receipt.	
PrinterDocument	When the Sale System wants to print specific document (check, dynamic currency conversion ...).	
PrinterVoucher	Coupons, voucher or special ticket generated by the Sale System and to be printed.	
MagStripe	Magnetic stripe card reader	
ICC	Contact ICC card reader	
EMVContactless	Contactless card reader with EMV applications	
CashHandling	Device which performs cash change, cash dispense or cash acceptance.	

Name	POIData
Definition	Data related to the POI System.
References	
Usage	In the Message Response, identification of the POI transaction.
Type	defined data structure
Format	
CrossRef	CardAcquisitionResponse LoyaltyResponse PaymentResponse ReversalResponse StoredValueResponse
XMLCoding	
ASN1Coding	204

Component	Mult.	Constraint	Rule
POITransactionID	[1..1]		
POIReconciliationID	[0..1]		If Result is Success

Name	POIID
Definition	Identification of a POI System or a POI Terminal for the Sale to POI protocol
References	
Usage	This is the hierarchical identification of the POI System and (possibly) a POI Terminal if necessary for the related message. Its scope is limited on the Sale to POI protocol only, and has a meaning for the Sale System only. In a Reversal, identify the POI Terminal which has initiated the transaction to reverse. In a TransactionStatus, identify the POI Terminal which has initiated the transaction to enquire. In the body of the Diagnosis request, it allows a POI to request the status of another POI. In the Diagnosis request, the POIID data element identifies the target of the diagnosis. It could point not only a POI Terminal, receiver or not of the Diagnosis request, but also the POI Server.
Type	TextString
Format	
CrossRef	AbortRequest.MessageReference DiagnosisRequest GetTotalsRequest.TotalFilter GetTotalsResponse.Totals.TransactionTotals InputUpdate.MessageReference MessageHeader ReversalRequest.OriginalPOITransaction ReversalResponse.OriginalPOITransaction ReconciliationResponse.Totals.TransactionTotals TransactionStatusRequest.MessageReference
XMLCoding	Attribute
ASN1Coding	205

Name	POIPProfile
Definition	Functional profile of the POI Terminal.
References	
Usage	Sent in the Login Response to identify the profiles the POI Terminal support during the session. The value of this data element contains: One of the generic profile: Basic, Standard or Extended, A list (possibly empty) of services related profiles: Loyalty, Reservation...
Type	defined data structure
Format	
CrossRef	LoginResponse.POISystemData.POITerminalData
XMLCoding	
ASN1Coding	207

Component	Mult.	Constraint	Rule
GenericProfile	[0..1]	default Standard	
ServiceProfiles	[0..1]		If a service profile could be requested

Name	POIReconciliationID
Definition	Identification of the reconciliation period between Sale and POI.
References	
Usage	Identification of the reconciliation period between Sale and POI, to provide the transaction totals during this period. Allows counting of transactions by both parties in the Sale to POI reconciliation. In the Reconciliation request, this field allows to request the reconciliation result of a previous period of transaction.
Type	DigitString
Format	
CrossRef	GetTotalsResponse.Totals LoyaltyResponse.POIData PaymentResponse.POIData ReversalResponse.POIData ReconciliationResponse.Totals StoredValueResponse.POIData
XMLCoding	Attribute
ASN1Coding	43

Name	POISerialNumber
Definition	Serial number of a POI Terminal
References	
Usage	Sent in the Login Request by the Sale System to inform the POI System what was the last known POI Terminal hardware. The POI sends its serial number in the Login Response to be identified by the Sale System.
Type	TextString
Format	
CrossRef	LoginRequest LoginResponse.POISystemData.POITerminalData
XMLCoding	Attribute
ASN1Coding	208

Name	POISoftware
Definition	Information related to the software of the POI System which manages the Sale to POI protocol.
References	
Usage	Allows in a session to identify the product features of a POI System.
Type	defined data structure
Format	
CrossRef	LoginResponse.POISystemData
XMLCoding	
ASN1Coding	206

Component	Mult.	Constraint	Rule
ManufacturerID	[1..1]		
ApplicationName	[1..1]		
SoftwareVersion	[1..1]		
CertificationCode	[1..1]		

Name	POIStatus
Definition	State of a POI Terminal.
References	
Usage	Indicate the availability of the POI Terminal components. The data element is absent if the component is not part of the POI Terminal.
Type	defined data structure
Format	
CrossRef	DiagnosisResponse LoginResponse
XMLCoding	
ASN1Coding	209

Component	Mult.	Constraint	Rule
GlobalStatus	[1..1]		
SecurityOKFlag	[0..1]		If security module present
PEDOKFlag	[0..1]		If PED present
CardReaderOKFlag	[0..1]		If card reader device present
PrinterStatus	[0..1]		If printer device present
CommunicationOKFlag	[0..1]		If communication infrastructure present
CashHandlingDevice	[0..n]		If cash handling devices present.
FraudPreventionFlag	[0..1]		default False

Name	POISystemData
<i>Definition</i>	Information related to the POI System
<i>References</i>	
<i>Usage</i>	In the Login message response, the data structure contains information related to the POI System fixed for the session or defined as default value.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	LoginResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	210

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
DateTime	[1..1]		
POISoftware	[1..1]		
POITerminalData	[0..1]		Present if the login involve a POI Terminal
POIStatus	[0..1]		if Response.Result is Success

Name	POITerminalData
<i>Definition</i>	Information related to the software and hardware feature of the POI Terminal
<i>References</i>	
<i>Usage</i>	Allows in a session to identify the features of the POI Terminal attached to a Sale Terminal per a Login Request message.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	LoginResponse.POISystemData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	211

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
TerminalEnvironment	[1..1]		
POICapabilities	[1..1]		
POIProfile	[0..1]		If at least one element is present. The Sale System decides if it can continue or not.
POISerialNumber	[1..1]		

Name	POITransactionID
<i>Definition</i>	Unique identification of a POI transaction for a POI
<i>References</i>	
<i>Usage</i>	To identify the transaction on the POI System.
<i>Type</i>	TransactionIdentificationType
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionResponse.POIData LoyaltyRequest.LoyaltyData.OriginalPOITransaction LoyaltyResponse.POIData PaymentRequest.PaymentData.OriginalPOITransaction PaymentResponse.POIData ReversalRequest.OriginalPOITransaction ReversalResponse.OriginalPOITransaction ReversalResponse.POIData StoredValueResponse.POIData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	212

Name	PredefinedContent
<i>Definition</i>	Reference of a predefined message to display or print.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the predefined message
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement EnableServiceRequest.Output.OutputElement PrintRequest.Output InputRequest.Output InputRequest.MenuEntry InputUpdate.Output CardReaderInitRequest.Output.OutputElement CardReaderAPDUREquest.Output.OutputElement CardReaderPowerOffRequest.Output.OutputElement
<i>XMLCoding</i>	
<i>ASN1Coding</i>	213

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
ReferenceID	[1..1]		
Language	[0..1]		

Name	PrinterStatus
<i>Definition</i>	Indicates if the printer is working and usable.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.POISatus LoginResponse.POISatus
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	214

<i>Label</i>	<i>Description</i>	<i>Code</i>
OK	The printer is operational.	
PaperLow	The printer is operational but paper roll is almost empty.	
NoPaper	Paper roll is empty, an operator must insert a new paper roll.	
PaperJam	An operator must remove the paper jam manually.	
OutOfOrder	The printer is out of order.	

Name	PrintOutput
<i>Definition</i>	Information to print and the way to process the print.
<i>References</i>	
<i>Usage</i>	It contains a complete print operation for a Print Device type.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PrintRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	215

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
DocumentQualifier	[1..1]		
ResponseMode	[1..1]		
IntegratedPrintFlag	[0..1]	default False	
RequiredSignatureFlag	[0..1]	default False	
OutputContent	[1..1]		
OutputSignature	[0..1]		If protection has to be provided to the vendor on the text to display or print.

Name	PrintRequest
Definition	Content of the Print Request message.
References	
Usage	It conveys the data to print and the way to process the print. It contains the complete content to print.
Type	defined data structure
Format	
CrossRef	PrintRequest
XMLCoding	
ASN1Coding	216

Component	Mult.	Constraint	Rule
PrintOutput	[1..1]		

Name	PrintResponse
Definition	Content of the Print Response message.
References	
Usage	It conveys the result of the print, parallel to the message request, except if response not required and absent.
Type	defined data structure
Format	
CrossRef	PrintResponse
XMLCoding	
ASN1Coding	217

Component	Mult.	Constraint	Rule
DocumentQualifier	[1..1]		Copy
Response	[1..1]		

Name	ProductCode
Definition	Product code of item purchased with the transaction.
References	
Usage	
Type	DigitString
Format	{1,20}
CrossRef	PaymentRequest.TransactionData.SaleItem PaymentResponse.LoyaltyResult.Rebates.SaleItemRebate LoyaltyRequest.TransactionData.SaleItem LoyaltyResponse.LoyaltyResult.Rebates.SaleItemRebate StoredValueRequest.StoredValueData.SaleItemRebate
XMLCoding	Attribute
ASN1Coding	218

Name	ProductLabel
Definition	Product name of an item purchased with the transaction.
References	
Usage	
Type	TextString
Format	
CrossRef	PaymentRequest.TransactionData.SaleItem LoyaltyRequest.TransactionData.SaleItem
XMLCoding	
ASN1Coding	393

Name	ProtectedCardData
Definition	Sensitive information related to the payment card, protected by CMS.
References	
Usage	This data structure is the data structure <i>SensitiveCardData</i> , which is CMS protected by encryption (EnvelopedData).
Type	ContentInformationType
Format	
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData CardAcquisitionResponse.PaymentInstrumentData.CardData PaymentRequest.PaymentData.PaymentInstrumentData.CardData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData
XMLCoding	
ASN1Coding	220

Name	ProtectedMobileData
Definition	Sensitive information related to the mobile phone, protected by CMS.
References	
Usage	This data structure is the data structure <i>SensitiveMobileData</i> , which is CMS protected by encryption (EnvelopedData).
Type	ContentInformationType
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData CardAcquisitionResponse.PaymentInstrumentData.MobileData PaymentRequest.PaymentData.PaymentInstrumentData.MobileData PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData
XMLCoding	
ASN1Coding	327

Name	ProtectedSignature
Definition	Numeric value of a handwritten signature.
References	
Usage	<p>Contain the value of a handwritten signature, e.g. the signature of a cardholder on the merchant payment receipt.</p> <p>The format before encryption is the encoded data structure CapturedSignature. The data structure before encryption includes the start and end tags for an XML encoding, the identifier and length bytes for an ASN.1 encoding, and the complete member ProtectedSignature for a JSON encoding.</p>
Type	ContentInformationType
Format	
CrossRef	PaymentResponse.PaymentResult
XMLCoding	
ASN1Coding	361

Name	ProtocolVersion
Definition	Version of the Sale to POI protocol specifications. For this version of specification, the value is "3.0"
References	
Usage	<p>Used for messages send outside a session, i.e. which are not between a Login and a Logout.</p> <p>In the Login message request, the Sale System sends the highest version of the Sale to POI protocol it can manage.</p> <p>In the Login message response, the POI System answers with the version of the Sale to POI protocol it will use.</p> <p>The version of the protocol sent in the Login request and response are valid for all the messages inside the session.</p> <p>Format of the protocol version is 'MM.mm', where <i>MM</i> is the major version number and <i>mm</i> the minor version number</p>
Type	TextString
Format	pattern MM.mm
CrossRef	LoginRequest.MessageHeader LoginResponse.MessageHeader
XMLCoding	Attribute
ASN1Coding	221

5.2.2.10 Q-R

Name	Quantity
<i>Definition</i>	Product quantity
<i>References</i>	
<i>Usage</i>	Sale item or rebate as additional units.
<i>Type</i>	Decimal
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.SaleItem PaymentResponse.LoyaltyResult.Rebates.SaleItemRebate LoyaltyRequest.TransactionData.SaleItem LoyaltyResponse.LoyaltyResult.Rebates.SaleItemRebate StoredValueRequest.StoredValueData.SaleItemRebate
<i>XMLCoding</i>	
<i>ASN1Coding</i>	222

Name	Rate
<i>Definition</i>	Rate of currency conversion.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Decimal
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult.CurrencyConversion
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	332

Name	Rebates
<i>Definition</i>	Rebate form to an award;
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.LoyaltyResult LoyaltyResponse.LoyaltyResult
<i>XMLCoding</i>	
<i>ASN1Coding</i>	223

Component	Mult.	Constraint	Rule
TotalRebate	[0..1]		If rebate on the total amount for this loyalty program
RebateLabel	[0..1]		If provided by the Acquirer
SaleItemRebate	[0..n]		only items with rebate (identified by ItemID)

Name	RebateLabel
Definition	Short text to qualify a rebate on an line item.
References	
Usage	Text to be printed on a receipt in front of the rebate on an item or the rebate on the totals.
Type	TextString
Format	
CrossRef	PaymentResponse.LoyaltyResult.Rebates.SaleItemRebate PaymentResponse.LoyaltyResult.Rebates LoyaltyResponse.LoyaltyResult.Rebates.SaleItemRebate LoyaltyResponse.LoyaltyResult.Rebates
XMLCoding	
ASN1Coding	224

Name	ReceiptReprintFlag
Definition	Request to reprint the POI receipt(s).
References	
Usage	Allow to reprint a receipt with a TransactionStatus message
Type	Boolean
Format	
CrossRef	TransactionStatusRequest
XMLCoding	Attribute
ASN1Coding	225

Name	ReconciliationRequest
<i>Definition</i>	Content of the Reconciliation Request message.
<i>References</i>	
<i>Usage</i>	<p>It conveys Information related to the Reconciliation requested by the Sale System:</p> <ul style="list-style-type: none"> Type of reconciliation (with or without closure of the reconciliation time period) The Acquirers identification if they are involved and could be selected for this particular processing The identification of the reconciliation period, if the scope does not concern the current reconciliation period
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	ReconciliationRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	226

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
ReconciliationType	[1..1]		
AcquirerID	[0..n]		Could be present only if ReconciliationType is "AcquirerReconciliation" or "AcquirerSynchronisation"
POIReconciliationID	[0..1]		Absent if ReconciliationType is not "PreviousReconciliation"

Name	ReconciliationResponse
<i>Definition</i>	Content of the Reconciliation Response message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Reconciliation transaction processed by the POI System
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	ReconciliationResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	227

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
ReconciliationType	[1..1]		Copy
POIReconciliationID	[0..1]		Absent if <i>ReconciliationType</i> is "AcquirerReconciliation"
TransactionTotals	[0..n]		if Response.Result is Success

Name	ReconciliationType
<i>Definition</i>	Type of Reconciliation requested by the Sale to the POI.
<i>References</i>	
<i>Usage</i>	Differentiate the various types of reconciliation to process.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	ReconciliationRequest
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	228

<i>Label</i>	<i>Description</i>	<i>Code</i>
SaleReconciliation	Reconciliation with closure of the current period, without any Acquirers synchronisation.	
AcquirerSynchronisation	Reconciliation and closure of the current period, with synchronisation of the reconciliation between the POI and Acquirers.	
AcquirerReconciliation	Reconciliation between the POI and one or several Acquirers only. There is no reconciliation between the Sale System and the POI System.	
PreviousReconciliation	Request result of a previous reconciliation.	

Name	ReferenceID
<i>Definition</i>	Identification of a predefined message to display, print or play.
<i>References</i>	
<i>Usage</i>	Allows the Sale or the POI Terminal to retrieve the message or the sound.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.MessageReference EnableServiceRequest.Output.OutputElement.MessageReference PrintRequest.Output.OutputElement.MessageReference InputRequest.Output.MessageReference InputRequest.MenuEntry.Predefined Content CardReaderInitRequest.Output.OutputElement.MessageReference CardReaderAPDUREquest.Output.OutputElement.MessageReference CardReaderPowerOffRequest.Output.OutputElement.MessageReference SoundRequest.SoundContent
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	229

Name	RejectedMessage
<i>Definition</i>	Message request rejected by the receiver.
<i>References</i>	
<i>Usage</i>	Sent in an event notification for an EventToNotify="Reject", allowing to have reject.
<i>Type</i>	ByteSequence
<i>Format</i>	
<i>CrossRef</i>	EventNotification
<i>XMLCoding</i>	
<i>ASN1Coding</i>	230

Name	RemoveAllFlag
<i>Definition</i>	Flag to remove all the transactions.
<i>References</i>	
<i>Usage</i>	Request in a batch the performed transactions, removing or not the transactions not performed.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	BatchRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	382

Name	RepeatedMessageResponse
<i>Definition</i>	Content of the requested Message Response.
<i>References</i>	
<i>Usage</i>	Allow the knowledge of the last Payment, Loyalty or Reversal transaction
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	TransactionStatusResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	231

Component	Mult.	Constraint	Rule
MessageHeader	[1..1]		
RepeatedResponseMessageBody	[1..1]		

Name	RepeatedResponseMessageBody
<i>Definition</i>	Repeated response message body.
<i>References</i>	
<i>Usage</i>	Used to repeat a response in the Transaction Status response.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	TransactionStatusResponse.RepeatedMessageResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	7

Or	Component	Mult.	Constraint	Rule
{Or}	LoyaltyResponse	[0..1]		
	PaymentResponse	[0..1]		
	ReversalResponse	[0..1]		
	StoredValueResponse	[0..1]		
	CardAcquisitionResponse	[0..1]		
Or}	CardReaderAPDUResponse	[0..1]		

Name	RequestedAmount
<i>Definition</i>	Amount requested by the Sale for the payment.
<i>References</i>	
<i>Usage</i>	Global payment amount including the cashback and the tip amount. In case of split payments, previous paid amounts are provided in the PaidAmount data element. If the PaidAmount is not present, previous paid amount is the difference between the RequestedAmount and the sum of the amount of the Sale Items. In case of OneTimeReservation PaymentType, this data element is present either the maximum amount is known or fixed by the Sale System, either the Sale wants to have a fixed amount. If this data is absent, and the POI gives the maximum authorised amount in the AuthorizedAmount of the PaymentResponse.
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.Amounts
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	232

Name	RequestedValidityDate
<i>Definition</i>	Requested validity date for the reservation.
<i>References</i>	
<i>Usage</i>	Allows a specific period for the reservation according to the need of the Merchant for the first reservation and the reservation updates as well.
<i>Type</i>	ISODate
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.PaymentData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	233

Name	RequestMessageBody
<i>Definition</i>	Request message body of the EPAS Sale To POI protocol.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	SaleToPOIRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	8

<i>Or</i>	<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
{Or}	AbortRequest	[0..1]		
	BalanceInquiryRequest	[0..1]		
	BatchRequest	[0..1]		
	CardAcquisitionRequest	[0..1]		
	AdminRequest	[0..1]		
	DiagnosisRequest	[0..1]		
	DisplayRequest	[0..1]		
	EnableServiceRequest	[0..1]		
	EventNotification	[0..1]		
	GetTotalsRequest	[0..1]		
	InputRequest	[0..1]		
	InputUpdate	[0..1]		
	LoginRequest	[0..1]		
	LogoutRequest	[0..1]		
	LoyaltyRequest	[0..1]		
	PaymentRequest	[0..1]		
	PINRequest	[0..1]		
	PrintRequest	[0..1]		
	CardReaderInitRequest	[0..1]		
	CardReaderAPDUREquest	[0..1]		
	CardReaderPowerOffRequest	[0..1]		
	ReconciliationRequest	[0..1]		
	ReversalRequest	[0..1]		
	SoundRequest	[0..1]		
	StoredValueRequest	[0..1]		
	TransactionStatusRequest	[0..1]		
Or}	TransmitRequest	[0..1]		

Name	RequiredSignatureFlag
Definition	Indicate that the cardholder payment receipt requires a physical signature by the Customer.
References	
Usage	
Type	Boolean
Format	
CrossRef	PaymentResponse.PaymentReceipt PrintRequest.Output
XMLCoding	Attribute
ASN1Coding	234

Name	Response
<i>Definition</i>	Result of a message request processing.
<i>References</i>	
<i>Usage</i>	If Result is Success, ErrorCondition is absent or not used in the processing of the message. In the other cases, the ErrorCondition has to be present and can refine the processing of the message response. AdditionalResponse gives more information about the success or the failure of the message request processing, for logging without real time involvements.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	AdminResponse BalanceInquiryResponse CardAcquisitionResponse DiagnosisResponse DisplayResponse.OutputResult EnableServiceResponse GetTotalsResponse LoyaltyResponse InputResponse.InputResult InputResponse.OutputResult LoginResponse LogoutResponse PaymentResponse PINResponse PrintResponse.OutputResult ReconciliationResponse CardReaderInitResponse CardReaderAPDUREsponse CardReaderPowerOffResponse ReversalResponse StoredValueResponse SoundResponse TransactionStatusResponse TransmitResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	235

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Result	[1..1]		
ErrorCondition	[0..1]		If Result is not Success
AdditionalResponse	[0..1]		If present, the POI logs it for further examination

Name	ResponseMessageBody
<i>Definition</i>	Response message body of the EPAS Sale To POI protocol.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	SaleToPOIResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	9

<i>Or</i>	<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
{Or}	BalanceInquiryResponse	[0..1]		
	BatchResponse	[0..1]		
	CardAcquisitionResponse	[0..1]		
	AdminResponse	[0..1]		
	DiagnosisResponse	[0..1]		
	DisplayResponse	[0..1]		
	EnableServiceResponse	[0..1]		
	GetTotalsResponse	[0..1]		
	InputResponse	[0..1]		
	LoginResponse	[0..1]		
	LogoutResponse	[0..1]		
	LoyaltyResponse	[0..1]		
	PaymentResponse	[0..1]		
	PINResponse	[0..1]		
	PrintResponse	[0..1]		
	CardReaderInitResponse	[0..1]		
	CardReaderAPDUREsponse	[0..1]		
	CardReaderPowerOffResponse	[0..1]		
	ReconciliationResponse	[0..1]		
	ReversalResponse	[0..1]		
	SoundResponse	[0..1]		
	StoredValueResponse	[0..1]		
	TransactionStatusResponse	[0..1]		
Or}	TransmitResponse	[0..1]		

Name	ResponseMode
Definition	Message response awaited by the initiator of the Request
References	
Usage	Allows various type and synchronisation of requests for Print or Sound.
Type	Enumeration
Format	
CrossRef	PrintRequest.Output SoundRequest
XMLCoding	Attribute
ASN1Coding	236

Label	Description	Code
NotRequired	The Message Response is not required, except in case of error.	
Immediate	The Message Response is immediate, after taking into account the request.	
PrintEnd	The Print Response is required at the end of print.	
SoundEnd	The Sound Response is required at the end of play.	

Name	ResponseRequiredFlag
Definition	Request of a message response.
References	
Usage	When a message response is optional, for instance for display message, it allows to request to send not to sent a message response to the display.
Type	Boolean
Format	
CrossRef	DisplayRequest.Output
XMLCoding	Attribute
ASN1Coding	237

Name	Result
<i>Definition</i>	Result of the processing of the message
<i>References</i>	
<i>Usage</i>	Global result of the message decoding and processing
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse CardAcquisitionResponse.Response DiagnosisResponse.Response DisplayResponse.OutputResult.Response EnableServiceResponse.Response GetTotalsResponse InputResponse.InputResult.Response InputResponse.OutputResult.Response LoginResponse.Response LoginResponse.Response LoginResponse.Response PaymentResponse.Response PINResponse.Response PrintResponse.OutputResult.Response ReconciliationResponse.Response CardReaderInitResponse.Response CardReaderAPDUResponse.Response CardReaderPowerOffResponse.Response ReversalResponse.Response SoundResponse.Response StoredValueResponse TransactionStatusResponse.Response TransmitResponse.Response
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	238

Label	Description	Code
Success	Processing OK. Information related to the result of the processing is contained in other parts of the response message.	
Failure	Processing of the request fails for various reasons. Some further processing according to the type of requested service, the context of the process, and some additional precision about the failure notified in the ErrorCondition data element.	
Partial	The transaction has been processed successfully, but the success is not complete (e.g. only a partial amount is available for the payment, the format to be displayed is not supported and was replaced by a default format).	

Name	ReuseCardDataFlag
Definition	Indicate if the card data has to be got from a previous transaction.
References	
Usage	Allows the POI to avoid reading again a card during a refund, or force to read again the card.
Type	Boolean
Format	
CrossRef	PaymentRequest.PaymentTransaction.OriginalPOITransaction
XMLCoding	Attribute
ASN1Coding	239

Name	ReversalReason
Definition	Reason of the payment or loyalty reversal..
References	
Usage	
Type	Enumeration
Format	
CrossRef	ReversalRequest
XMLCoding	Attribute
ASN1Coding	240

Label	Description	Code
CustCancel	Customer cancellation	
MerchantCancel	Cashier cancellation	
Malfunction	Suspected malfunction	
Unable2Compl	Card acceptor device unable to complete transaction	

Name	ReversalRequest
<i>Definition</i>	Content of the Reversal Request message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the reversal of a previous payment or a loyalty transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BatchRequest.TransactionToPerform ReversalRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	242

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
SaleReferenceID	[0..1]		If payment reservation reversal
OriginalPOITransaction	[1..1]		
ReversalReason	[1..1]		
ReversedAmount	[0..1]		ReversedAmount is implicitly the AuthorizedAmount if absent.
CustomerOrder	[0..1]		If the reversal is performed inside a customer order.

Name	ReversalResponse
<i>Definition</i>	Content of the Reversal Response message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the reversal processed by the POI System.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	ReversalResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	243

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
POIData	[0..1]		If Result is Success
OriginalPOITransaction	[0..1]		Present if POITransactionID absent in the request
ReversedAmount	[0..1]		Copy
CustomerOrder	[0..1]		If the reversal is performed inside a customer order.
PaymentReceipt	[0..n]		If Basic profile implementation with no printer on the POI.

Name	ReversedAmount
Definition	Amount of the payment or loyalty to reverse..
References	
Usage	ReversedAmount is implicitly the AuthorizedAmount if absent. ReversedAmount is lower or equal AuthorizedAmount of the OriginalPOITransaction.
Type	SimpleAmountType
Format	
CrossRef	ReversalRequest ReversalResponse
XMLCoding	Attribute
ASN1Coding	241

5.2.2.11 S

Name	SaleCapabilities
Definition	Hardware capabilities of the Sale Terminal.
References	
Usage	<p>Sale device the Sale System allows the POI System uses.</p> <p>These Sale devices include the logical devices enumerated in the Devices data element (all these devices could be on the Sale part), plus card reader devices.</p> <p>Sent in the Login Request to identify the devices of the Sale Terminal which can be used by the POI System during the session.</p> <p>In other message, when a logical device is out of order, this data has changed since the Login.</p>
Type	Cluster
Format	
CrossRef	<p>LoginRequest.SaleTerminalData</p> <p>LoyaltyRequest.SaleData.SaleTerminalData</p> <p>PaymentRequest.SaleData.SaleTerminalData</p> <p>StoredValueRequest.SaleData.POITerminalData</p>
XMLCoding	
ASN1Coding	244

Label	Description	Code
CashierStatus	To display to the Cashier a new state on which the POI is entering. For instance, during a payment, the POI could display to the Cashier that POI request an authorisation to the host acquirer.	
CashierError	To display to the Cashier information related to an error situation occurring on the POI.	
CashierDisplay	Standard Cashier display interface (to ask question, or to show information).	
POIReplication	Information displayed on the Cardholder POI interface, replicated on the Cashier interface.	
CashierInput	Any kind of keyboard allowing all or part of the commands of the Input message request from the Sale System to the POI System (InputCommand data element). The output device attached to this input device is the CashierDisplay device.	
CustomerAssistance	Input of the Cardholder POI interface which can be entered by the Cashier to assist the Customer.	
CustomerDisplay	Standard Customer display interface used by the POI System to ask question, or to show information to the Customer inside a Service dialogue.	
CustomerError	To display to the Customer information related to an error situation occurring on the Sale Terminal during a Sale transaction.	
CustomerInput	Any kind of keyboard allowing all or part of the commands of the Input message request from the Sale System to the POI System (InputCommand data element). The output device attached to this input device is the CashierDisplay device.	
PrinterReceipt	Printer for the Payment receipt.	
PrinterDocument	When the POI System wants to print specific document (check, dynamic currency conversion ...).	
PrinterVoucher	Coupons, voucher or special ticket generated by the POI and to be printed.	
MagStripe	Magnetic stripe card reader	
ICC	Contact ICC card reader	
EMVContactless	Contactless card reader with EMV applications	

Name	SaleChannel
<i>Definition</i>	Commercial or distribution channel associated to the line item.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	DigitString
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.SaleItem LoyaltyRequest.TransactionData.SaleItem StoredValueRequest.StoredValueData.SaleItemRebate
<i>XMLCoding</i>	
<i>ASN1Coding</i>	245

Name	SaleData
<i>Definition</i>	Data related to the Sale System.
<i>References</i>	
<i>Usage</i>	Data associated to the Sale System, with a particular value during the processing of the payment by the POI, including the cards acquisition.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	CardAcquisitionRequest PaymentRequest PaymentResponse StoredValueRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	246

Component	Mult.	Constraint	Rule
OperatorID	[0..1]		if different from the Login and see <i>Login .SaleData</i>
OperatorLanguage	[0..1]		if different from the Login
ShiftNumber	[0..1]		if different from the Login and see <i>Login .SaleData</i>
SaleTransactionID	[1..1]		
SaleReferenceID	[0..1]		If payment reservation
SaleTerminalData	[0..1]		If content is not empty
TokenRequestedType	[0..1]		In a Payment or CardAcquisition request, if a token is requested.
CustomerOrderID	[0..1]		If the payment is related to an open customer order.
CustomerOrderReq	[0..1]		If customer orders must be listed in the response message.
SaleToPOIData	[0..1]		Stored with the transaction
SaleToAcquirerData	[0..1]		Send to the Acquirer if present
SaleToIssuerData	[0..1]		Send to the Acquirer if present

Name	SaleID
Definition	Identification of a Sale System or a Sale Terminal for the Sale to POI protocol
References	
Usage	This is the hierarchical identification of the Sale System and (possibly) a Sale Terminal if necessary for the related message. Its scope is limited on the Sale to POI protocol only, and has a meaning for the POI System only. In a Reversal, identify the Sale Terminal which has initiated the transaction to reverse. In a TransactionStatus, identify the Sale Terminal which has initiated the transaction to enquire.
Type	TextString
Format	
CrossRef	GetTotalsRequest.TotalFilter GetTotalsResponse.Totals.TransactionTotals InputUpdate.MessageReference MessageHeader ReversalRequest.OriginalPOITransaction ReversalResponse.OriginalPOITransaction ReconciliationResponse.Totals.TransactionTotals TransactionStatusRequest.MessageReference AbortRequest.MessageReference
XMLCoding	Attribute
ASN1Coding	248

Name	SaleItem
Definition	Sale items of a transaction.
References	
Usage	<p>In loyalty or value added payment card transaction, the items of the sale are entering in the processing of the transaction.</p> <p>The sum of the item amount could be more than the Requested amount in case of split payment without split of the items (split basket).</p> <p>In the StoredValue request message, sale items differentiate the Stored Value product (e.g. e-topup and the corresponding operator, giftCard...) with the EanUpc data element which has to be present.</p>
Type	defined data structure
Format	
CrossRef	PaymentRequest.TransactionData LoyaltyRequest.TransactionData StoredValueRequest.StoredValueData
XMLCoding	
ASN1Coding	249

Component	Mult.	Constraint	Rule
ItemID	[1..1]		
ProductCode	[1..1]		
EanUpc	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it
UnitOfMeasure	[0..1]		if Quantity present
Quantity	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it
UnitPrice	[0..1]		if Quantity present
ItemAmount	[1..1]		
TaxCode	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it
SaleChannel	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it
ProductLabel	[0..1]		
AdditionalProductInfo	[0..1]		If data sent, POI has to store it and send it if the host protocol allows it

Name	SaleItemRebate
Definition	The awarded amount that is attached to an item as a rebate.
References	
Usage	To be differentiated from the award which is the amount or quantity earned on the loyalty account.
Type	defined data structure
Format	
CrossRef	PaymentResponse.LoyaltyResult.Rebates LoyaltyResponse.LoyaltyResult.Rebates
XMLCoding	
ASN1Coding	250

Component	Mult.	Constraint	Rule
ItemID	[1..1]		
ProductCode	[1..1]		
EanUpc	[0..1]		if present in the related SaleItem
UnitOfMeasure	[0..1]		if Quantity present
Quantity	[0..1]		if rebate is additional units
ItemAmount	[0..1]		if rebate on the line item amount
RebateLabel	[0..1]		If provided by the Acquirer

Name	SaleProfile
Definition	Functional profile of the Sale Terminal.
References	
Usage	Sent in the Login Request to identify the functions that might be requested by the Sale Terminal during the session. The value of this data element contains: One of the generic profile: Basic, Standard or Extended, A list (possibly empty) of services related profiles: Loyalty, Reservation...
Type	defined data structure
Format	
CrossRef	LoginRequest.SaleTerminalData
XMLCoding	
ASN1Coding	252

Component	Mult.	Constraint	Rule
GenericProfile	[0..1]	default Standard	
ServiceProfiles	[0..1]		If a service profile could be requested

Name	SaleReferenceID
Definition	Identification of a Sale global transaction for a sequence of related POI transactions
References	
Usage	Identification of a reservation transaction for the sequence of reservation and the completion.
Type	TextString
Format	
CrossRef	PaymentRequest.PaymentData.OriginalPOITransaction
XMLCoding	Attribute
ASN1Coding	253

Name	SaleSoftware
Definition	Information related to the software of the Sale System which manages the Sale to POI protocol.
References	
Usage	Allows in a session to identify the product features of a Sale System.
Type	defined data structure
Format	
CrossRef	LoginRequest
XMLCoding	
ASN1Coding	251

Component	Mult.	Constraint	Rule
ManufacturerID	[1..1]		
ApplicationName	[1..1]		
SoftwareVersion	[1..1]		
CertificationCode	[1..1]		

Name	SaleTerminalData
Definition	Information related to the software and hardware feature of the Sale Terminal.
References	
Usage	In the Login Request, if a Sale Terminal is involved in the login. In other messages, when a logical device is out of order (SaleCapabilites), or when the other data have changed since or were not in the Login.
Type	defined data structure
Format	
CrossRef	CardAcquisitionRequest.SaleData LoginRequest LoyaltyRequest.SaleData PaymentRequest.SaleData StoredValueRequest.SaleData
XMLCoding	
ASN1Coding	254

Component	Mult.	Constraint	Rule
TerminalEnvironment	[0..1]		
SaleCapabilities	[0..1]		
SaleProfile	[0..1]		If at least one element is present
TotalsGroupID	[0..1]		If present, default value for all transaction.

Name	SaleToAcquirerData
Definition	Sale information intended for the Acquirer.
References	
Usage	The POI System receives this information and sends it to the Acquirer without any change.
Type	TextString
Format	
CrossRef	PaymentRequest.TransactionData LoyaltyRequest.TransactionData
XMLCoding	
ASN1Coding	255

Name	SaleToIssuerData
Definition	Sale information intended for the Issuer.
References	
Usage	The POI System receives this information and sends it to the Acquirer for the Issuer without any change.
Type	defined data structure
Format	
CrossRef	PaymentRequest.TransactionData LoyaltyRequest.TransactionData
XMLCoding	
ASN1Coding	256

Component	Mult.	Constraint	Rule
StatementReference	[0..1]		Information to print on the bank statement

Name	SaleToPOIData
Definition	Sale information intended for the POI.
References	
Usage	The POI System receives this information which is meaningful only for the Sale System. The POI stores this information with the transaction.
Type	TextString
Format	
CrossRef	PaymentRequest.TransactionData LoyaltyRequest.TransactionData
XMLCoding	
ASN1Coding	257

Name	SaleToPOIRequest
Definition	Request message of the EPAS Sale To POI protocol.
References	
Usage	
Type	Message
Format	
CrossRef	
XMLCoding	
ASN1Coding	10

Component	Mult.	Constraint	Rule
MessageHeader	[1..1]		
RequestMessageBody	[1..1]		
SecurityTrailer	[0..1]		

Name	SaleToPOIResponse
Definition	Response message of the EPAS Sale To POI protocol.
References	
Usage	
Type	Message
Format	
CrossRef	
XMLCoding	
ASN1Coding	11

Component	Mult.	Constraint	Rule
MessageHeader	[1..1]		
ResponseMessageBody	[1..1]		
SecurityTrailer	[0..1]		

Name	SaleTransactionID
Definition	Unique identification of a Sale transaction
References	
Usage	To identify the transaction on the Sale Terminal, e.g. ticket number.
Type	TransactionIdentificationType
Format	
CrossRef	CardAcquisitionRequest.SaleData LoyaltyRequest.SaleData PaymentRequest.SaleData StoredValueRequest.SaleData
XMLCoding	
ASN1Coding	258

Name	SecurityOKFlag
Definition	Indicates if the security module of the POI is working and usable.
References	
Usage	
Type	Boolean
Format	
CrossRef	DiagnosisResponse.POIStatus LoginResponse.POIStatus
XMLCoding	Attribute
ASN1Coding	259

Name	SecurityTrailer
Definition	Protection of the whole message
References	
Usage	This optional data structure contains a MAC on the MessageHeader and MessageBody, which is CMS AuthenticatedData alternative.
Type	ContentInformationType
Format	
CrossRef	
XMLCoding	
ASN1Coding	312

Name	SensitiveCardData
<i>Definition</i>	Sensitive information related to the payment card, entered or read by the Sale System.
<i>References</i>	
<i>Usage</i>	This data structure could be CMS protected (EnvelopedData). In this case the data structure SensitiveCardData is replaced by the data structure ProtectedCardData of type ContentInformationType. When this data is protected, the exact content is unknown by the Sale System, and might include all the information which are required by an external backup POI Server to make a batch payment transaction in case of problem with the POI System.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData CardAcquisitionResponse.PaymentInstrumentData.CardData PaymentRequest.PaymentData.PaymentInstrumentData.CardData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	260

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
PAN	[0..1]		if EntryMode is File, Keyed or Manual
CardSeqNumb	[0..1]		if EntryMode is File, Keyed or Manual
ExpiryDate	[0..1]		if EntryMode is File
TrackData	[0..4]		if EntryMode is MagStripe or RFID

Name	SensitiveMobileData
<i>Definition</i>	Sensitive information related to the mobile phone.
<i>References</i>	
<i>Usage</i>	This data structure could be CMS protected (EnvelopedData). In this case the data structure SensitiveMobileData is replaced by the data structure ProtectedMobileData of type ContentInformationType. When this data is protected, the exact content is unknown by the Sale System, and might include all the information which are required by an external backup POI Server to make a batch payment transaction in case of problem with the POI System.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData CardAcquisitionResponse.PaymentInstrumentData.MobileData PaymentRequest.PaymentData.PaymentInstrumentData.MobileData PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	328

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
MSISDN	[1..1]		
IMSI	[0..1]		If data available
IMEI	[0..1]		If data available

Name	SequenceNumber
<i>Definition</i>	Sequence number of the instalment.
<i>References</i>	
<i>Usage</i>	For an instalment payment transaction, number of the payment, from 1 to TotalNbOfPayments.
<i>Type</i>	Integer
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.PaymentData.Instalment
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	369

Name	ServiceID
<i>Definition</i>	Identification of a message pair, which processes a transaction
<i>References</i>	
<i>Usage</i>	<p>It allows a unique identification of a message pair, between a Sale System/Terminal and a POI System/Terminal during period of time, typically one day.</p> <p>The receiver of a request message has to check that this identifier has not the same value than the last one for the same initiator.</p> <p>This identifier is mandatory for messages of the "Service" and "Event" MessageClass.</p> <p>This identifier is mandatory for messages of the "Device" MessageClass, if the request comes from the POI.</p> <p>This identifier is absent for messages of the "Device" MessageClass, if the request comes from the Sale.</p> <p>The ServiceID of the message response is always the same value than the ServiceID of the message request.</p> <p>This identifier allows the recognition of duplicate message and association of a message response to its message request.</p> <p>Value of ServiceID could be a structured string of alpha characters, for instance with a prefix to easily distinguish a particular sequence of messages.</p> <p>In a TransactionStatus, the message body contains the ServiceID value which identifies the "Service" message exchange to enquire.</p>
<i>Type</i>	TextString
<i>Format</i>	{1,10}
<i>CrossRef</i>	MessageHeader InputUpdate.MessageReference TransactionStatusRequest.MessageReference
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	262

Name	ServiceIdentification
<i>Definition</i>	Identification of the administrative service to process.
<i>References</i>	
<i>Usage</i>	Direct identification of the administrative service to process with the Admin exchange. It could be a name, or the sequence of items in the menu separated by comma (csv format).
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	AdminRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	318

Name	ServicesEnabled
<i>Definition</i>	Services which are enabled before the start-up of a transaction
<i>References</i>	
<i>Usage</i>	Allows the "swipe-ahead" mechanism, i.e. the possibility for the Cardholder to start the processing of a transaction, before the request of the Sale System. The transaction processing could be: Either a CardAcquisition, Either a combination of Payment and Loyalty.
<i>Type</i>	Cluster
<i>Format</i>	
<i>CrossRef</i>	EnableServiceRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	261

Label	Description	Code
CardAcquisition	Enable the POI to process a card acquisition before the request of the Sale System (e.g. the same processing than the CardAcquisition command, but no more)	
Payment	Enable the POI to start a payment transaction before the request of the Sale System (e.g. the same processing than the Payment command)	
Loyalty	Enable the POI to start a loyalty transaction before the request of the Sale System (e.g. the same processing than the Loyalty command)	

Name	ServiceProfiles
<i>Definition</i>	Service profiles of the Sale to POI protocol.
<i>References</i>	
<i>Usage</i>	Sent in the Login Request to identify the specific services that might be requested by the Sale Terminal during the session. Sent in the Login Response to identify the specific services that might be processed by the POI Terminal during the session.
<i>Type</i>	Cluster
<i>Format</i>	
<i>CrossRef</i>	LoginRequest.SaleSystemData.SaleTerminalData.SaleProfile LoginResponse.POISystemData.POITerminalData.POIPProfile
<i>XMLCoding</i>	
<i>ASN1Coding</i>	263

<i>Label</i>	<i>Description</i>	<i>Code</i>
Synchro	CardAcquisition and EnableService could be requested by the Sale System to the POI System.	
Batch	When the POI is unable to perform transactions without the Sale system, e.g. payment at delivery, the POI provides in a Batch the performed transactions, and may preload transaction to perform.	
OneTimeRes	One Time Reservation service could be requested by the Sale System (as petrol distribution)	
Reservation	The Reservation services could be requested by the Sale System	
Loyalty	Loyalty services could be requested by the Sale System	
StoredValue	Stored Value service could be requested by the Sale System	
PIN	The Sale System could request the PIN validation services.	
CardReader	The Sale System could request Card Reader services.	
Sound	To produce various forms of sounds to a customer or an operator interface.	
Communication	The POI or Sale System could request communication through the Transmit device messages exchange.	

Name	ShiftNumber
Definition	Shift number.
References	
Usage	Sent in the Login Request for information only, to identify the shift that drives the Sale Terminal during the session
Type	TextString
Format	
CrossRef	GetTotalsRequest.TotalFilter GetTotalsResponse.Totals.TransactionTotals LoginRequest LoyaltyRequest.SaleData PaymentRequest.SaleSystemData ReconciliationResponse.Totals.TransactionTotals StoredValueRequest.SaleData
XMLCoding	Attribute
ASN1Coding	264

Name	SignaturePoint
Definition	Coordinates of a point where the pen changes direction or lift.
References	
Usage	Contain the Coordinates of a point of the written signature where the pen changes direction or lift where (X and Y). When the signer lifts the pen, both X and Y have the value "FFFF".
Type	defined data structure
Format	
CrossRef	PaymentResponse.PaymentResult.CapturedSignature
XMLCoding	
ASN1Coding	344

Component	Mult.	Constraint	Rule
X	[1..1]		
Y	[1..1]		

Name	SoftwareVersion
Definition	Version of the software product
References	
Usage	Sent in the Login Request (resp. Response) to identify the version of the Sale System (resp. POI System) product software during the session.
Type	TextString
Format	
CrossRef	LoginRequest.SaleSoftware LoginResponse.POISystemData.POISoftware
XMLCoding	Attribute
ASN1Coding	265

Name	SoundAction
Definition	Type of action to perform on the sound.
References	
Usage	
Type	Enumeration
Format	
CrossRef	SoundRequest
XMLCoding	Attribute
ASN1Coding	395

Label	Description	Code
StartSound	Start the sound as specified in the message.	
StopSound	Stop the sound in progress.	
SetDefaultVolume	Set the default volume of sounds.	

Name	SoundContent
Definition	Content of the Sound to play.
References	
Usage	
Type	defined data structure
Format	
CrossRef	SoundRequest
XMLCoding	
ASN1Coding	396

Component	Mult.	Constraint	Rule
SoundFormat	[0..1]		
Language	[0..1]		
ReferenceID	[0..1]		Mandatory if SoundFormat is SoundRef or MessageRef
Text	[0..1]		Mandatory if SoundFormat is Text

Name	SoundRequest
<i>Definition</i>	Content of the Sound Request message.
<i>References</i>	
<i>Usage</i>	It conveys the data to start a sound, stop a sound, or modify the default sound volume. The sound to play may be a preloaded sound or a text to play.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	SoundRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	397

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
ResponseMode	[0..1]	default NotRequired	
SoundAction	[1..1]		
SoundVolume	[0..1]		Mandatory if SoundAction is SetDefaultVolume
SoundContent	[1..1]		Absent if SoundAction is SetDefaultVolume, otherwise mandatory.

Name	SoundResponse
<i>Definition</i>	Content of the Sound Response message.
<i>References</i>	
<i>Usage</i>	It conveys the result of the Sound request. The response may be absent, at the beginning of the processing, or at the end of the processing.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	SoundResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	398

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		

Name	SoundFormat
Definition	Type of sound to play.
References	
Usage	
Type	Enumeration
Format	
CrossRef	SoundRequest.SoundContent
XMLCoding	Attribute
ASN1Coding	399

Label	Description	Code
SoundRef	Preloaded sound File.	
MessageRef	Reference of a preloaded text to play.	
Text	Text to play.	

Name	SoundVolume
Definition	Volume of a sound, either in a pourcentage of the maximum volume, or 0 to mute.
References	
Usage	
Type	Integer
Format	
CrossRef	SoundRequest
XMLCoding	Attribute
ASN1Coding	400

Name	SplitPaymentFlag
<i>Definition</i>	Indicates if the payment of the Sale transaction is split.
<i>References</i>	
<i>Usage</i>	Allows the POI to decline payment means that cannot accept split payment.
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.PaymentData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	266

Name	StartColumn
<i>Definition</i>	Column from which the text string has to be displayed or printed.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Integer
<i>Format</i>	[1-100]
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputElement.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	267

Name	StartRow
<i>Definition</i>	Row from which the text string has to be displayed or printed.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Integer
<i>Format</i>	[1-500]
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputElement.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDUREquest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	268

Name	StartDate
Definition	Date time of the beginning of an operation.
References	
Usage	
Type	ISODateTime
Format	
CrossRef	CardAcquisitionResponse.CustomerOrder PaymentRequest.PaymentData.CustomerOrder PaymentResponse.CustomerOrder ReversalRequest.CustomerOrder ReversalResponse.CustomerOrder
XMLCoding	Attribute
ASN1Coding	418

Name	StatementReference
Definition	Label to print on the bank statement.
References	
Usage	To reference a transaction on the bank statement
Type	TextString
Format	
CrossRef	PaymentRequest.TransactionData
XMLCoding	
ASN1Coding	269

Name	StoredValueAccountID
<i>Definition</i>	Identification of the stored value account or the stored value card
<i>References</i>	
<i>Usage</i>	It contains the identifications of the stored value account or the stored value card, and the associated product sold by the Sale System for stored value requests.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest BalanceInquiryRequest.PaymentData.PaymentInstrumentData BalanceInquiryResponse.StoredValueResult BalanceInquiryResponse.PaymentResult.PaymentInstrumentData CardAcquisitionResponse.PaymentInstrumentData PaymentRequest.PaymentData.PaymentInstrumentData PaymentResponse.PaymentResult.PaymentInstrumentData StoredValueRequest.StoredValueData StoredValueResponse.StoredValueResult
<i>XML Coding</i>	
<i>ASN1 Coding</i>	270

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
StoredValueAccountType	[1..1]		
StoredValueProvider	[0..1]		If available for the card or account.
OwnerName	[0..1]		If available for the card or account.
ExpiryDate	[0..1]		If required for the card or account.
EntryMode	[1..1]		
IdentificationType	[1..1]		
StoredValueID	[1..1]		

Name	StoredValueAccountStatus
<i>Definition</i>	Data related to the result of the stored value card transaction.
<i>References</i>	
<i>Usage</i>	It contains: - the identification of the stored value accounts or the stored value cards - the identification of the transaction by the stored value host - the balance of the stored value account if relevant
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse StoredValueResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	271

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
StoredValueAccountID	[1..1]		
CurrentBalance	[0..1]		if relevant and known

Name	StoredValueAccountType
<i>Definition</i>	Type of stored value account
<i>References</i>	
<i>Usage</i>	Allow the distinction of the stored value instrument to access the stored value account.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID CardAcquisitionResponse.PaymentInstrumentData.StoredValueAccountID PaymentRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID PaymentResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID StoredValueRequest.StoredValueData.StoredValueAccountID StoredValueResponse.StoredValueResult.StoredValueAccountID
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	353

<i>Label</i>	<i>Description</i>	<i>Code</i>
GiftCard	Payment mean issued by retailers or banks as a substitute to a non-monetary gift.	
PhoneCard	Stored value instrument used to pay telephone services (e.g. card or identifier).	
Other	Other stored value instrument.	

Name	StoredValueData
<i>Definition</i>	Data related to the stored value card.
<i>References</i>	
<i>Usage</i>	<p>It contains:</p> <ul style="list-style-type: none"> - the identification of the stored value accounts or the stored value cards, if provided by the Sale System, and - the associated products sold by the Sale System..
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	StoredValueRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	272

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
StoredValueProvider	[0..1]		If more than one provider to manage on the POI, and StoredValueAccountID absent.
StoredValueTransactionType	[1..1]		
StoredValueAccountID	[0..1]		If the identification of the Stored Value account or card has been made by the Sale System before the request
OriginalPOITransaction	[0..1]		if StoredValueTransactionType is Reverse or Duplicate
ProductCode	[0..1]		
EanUpc	[0..1]		
ItemAmount	[1..1]		
Currency	[1..1]		

Name	StoredValueID
Definition	Stored value account identification
References	
Usage	The identification of the stored value account conforming to the IdentificationType.
Type	TextString
Format	
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID CardAcquisitionResponse.PaymentInstrumentData.StoredValueAccountID PaymentRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID PaymentResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID StoredValueRequest.StoredValueData.StoredValueAccountID StoredValueResponse.StoredValueResult.StoredValueAccountID
XMLCoding	Enclosing
ASN1Coding	273

Name	StoredValueProvider
Definition	Identification of the provider of the stored value account load/reload
References	
Usage	When the ProductCode is not sufficient to identify the provider host which deliver the load or reload of the stored value account (e. g. it could contain the identification of the "application").
Type	TextString
Format	
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID CardAcquisitionResponse.PaymentInstrumentData.StoredValueAccountID PaymentRequest.PaymentData.PaymentInstrumentData.StoredValueAccountID PaymentResponse.PaymentResult.PaymentInstrumentData.StoredValueAccountID StoredValueRequest.StoredValueData.StoredValueAccountID StoredValueRequest.StoredValueData StoredValueResponse.StoredValueResult.StoredValueAccountID
XMLCoding	Attribute
ASN1Coding	274

Name	StoredValueRequest
<i>Definition</i>	Content of the Stored Value Request message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Stored Value transaction to process
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	StoredValueRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	275

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
SaleData	[1..1]		
CustomerLanguage	[0..1]		If the language is selected by the Sale System before the request to the POI.
StoredValueData	[1..n]		

Name	StoredValueResponse
<i>Definition</i>	Content of the Stored Value Response message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the Stored Value transaction processed by the POI System.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	StoredValueResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	276

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
SaleData	[1..1]		
POIData	[1..1]		
StoredValueResult	[0..n]		If StoredValueResponse.Result is "Success" or "Partial", one entry per StoredValueRequest.StoredValueData loaded or activated

Name	StoredValueResult
<i>Definition</i>	Result of loading/reloading a stored value card..
<i>References</i>	
<i>Usage</i>	For each stored value card loaded or reloaded, in the StoredValue response message
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	StoredValueResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	277

Component	Mult.	Constraint	Rule
StoredValueTransactionType	[1..1]		Copy
ProductCode	[1..1]		Copy
EanUpc	[0..1]		Copy
ItemAmount	[1..1]		
Currency	[1..1]		Copy
StoredValueAccountStatus	[1..1]		
HostTransactionID	[0..1]		If provided by the Host

Name	StoredValueTransactionType
<i>Definition</i>	Identification of operation to proceed on the stored value account or the stored value card
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	StoredValueRequest.StoredValueData StoredValueResponse.StoredValueResult
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	278

Label	Description	Code
Reserve	Reserve the account (e.g. get an activation code)	
Activate	Activate the account or the card	
Load	Load the account or the card with money	
Unload	Unload the account	
Reverse	Reverse an activation or loading.	
Duplicate	Duplicate the code or number provided by the loading or activation	

Name	StringMask
<i>Definition</i>	String mask to get information requiring a specific format.
<i>References</i>	
<i>Usage</i>	For the processing of an Input command TextString, DigitString or DecimalString. Some information as date or plate number required to be entered with a certain format. The String mask can contain the following characters: - 'd': a digit '0' to '9' - 'a': an alphabetic character 'a' to 'z', 'A' to 'Z' - 's': any other printable US ASCII character - any other character will be displayed but not entered (including space) - '\': the escape characters to precede the characters 'd', 'a', 's' or '\', in order to display these characters The string sent in the response contains the string mask replaces the 'a','d' and 's' characters by what is entered by the user.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	InputRequest.InpuData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	279

5.2.2.12 T

Name	TaxCode
<i>Definition</i>	Type of taxes associated to the line item.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	DigitString
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.SaleItem LoyaltyRequest.TransactionData.SaleItem StoredValueRequest.StoredValueData.SaleItemRebate
<i>XMLCoding</i>	
<i>ASN1Coding</i>	280

Name	TerminalEnvironment
<i>Definition</i>	Environment of the Terminal.
<i>References</i>	
<i>Usage</i>	Sent in the Login Request (resp. Response) to identify the environment of the Sale System (resp. POI System) during the session. In other message, when the data has changed since the Login.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	LoginRequest.SaleTerminalData LoyaltyRequest.SaleData.SaleTerminalData LoginResponse.POISystemData.POITerminalData PaymentRequest.SaleData.SaleTerminalData StoredValueRequest.SaleData.POITerminalData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	281

<i>Label</i>	<i>Description</i>	<i>Code</i>
Attended	The Sale Terminal is managed by a Cashier through the Sale System. A Cashier Interface is provided by the POI System during the process of a request from the Sale. The POI Terminal is managed by the Customer through the Customer Interface.	
SemiAttended	Without a Cashier Interface. The POI Terminal is managed by the Customer through the Customer Interface. A Cashier could help the Cardholder during the transaction if he asks.	
Unattended	The Sale Terminal is managed as a logical terminal without any Cashier Interface (typically a background process). The POI Terminal is managed by the Customer through the Customer Interface.	

Name	Text
<i>Definition</i>	Content of text message to display, print or play.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	DisplayRequest.Output.OutputElement.OutputText EnableServiceRequest.Output.OutputElement.OutputText InputRequest.Output.OutputText InputUpdate.Output.OutputText PaymentResponse.PaymentReceipt.OutputElement.OutputText PrintRequest.Output.OutputElement.OutputText CardReaderInitRequest.Output.OutputElement.OutputText CardReaderAPDURequest.Output.OutputElement.OutputText CardReaderPowerOffRequest.Output.OutputElement.OutputText SoundRequest.SoundContent
<i>XMLCoding</i>	Enclosing
<i>ASN1Coding</i>	283

Name	TextInput
Definition	The text which is typed by the Customer on the POI or the Cashier on the Sale Terminal.
References	
Usage	Input entered by the user, in response to an <i>InputCommand</i> "TextString" or "Password" for a plaintext password.
Type	TextString
Format	
CrossRef	InputResponse.InputResult.Input
XMLCoding	
ASN1Coding	282

Name	TimeStamp
Definition	Date and time of a transaction for the Sale System, the POI System or the Acquirer.
References	
Usage	Ensure the uniqueness of a transaction for Sale System, the POI System or the Acquirer. Indicates the time when the event occurs in the EventNotification message.
Type	ISODateTime
Format	
CrossRef	EventNotification TransactionIdentificationType
XMLCoding	Attribute
ASN1Coding	284

Name	TipAmount
Definition	Amount paid for a tip.
References	
Usage	Allow the printing of the tip on the receipt, and to qualify the tip part of the amount.
Type	SimpleAmountType
Format	
CrossRef	PaymentRequest.TransactionData.Amounts PaymentResponse.PaymentResult.Amounts
XMLCoding	Attribute
ASN1Coding	285

Name	TokenRequestedType
Definition	Type of token replacing the PAN of a payment card to identify the payment mean of the customer.
References	
Usage	It allows, for a merchant, to use a token for a transaction only or for a longer period.
Type	Enumeration
Format	
CrossRef	CardAcquisitionRequest.SaleData CardAcquisitionResponse.PaymentInstrumentData.CardData.PaymentToken PaymentRequest.SaleData PaymentResponse.PaymentResult.PaymentInstrumentData.CardData.PaymentToken
XMLCoding	Attribute
ASN1Coding	408

Label	Description	Code
Transaction	The token is generated to recognise a customer during the time of a transaction.	
Customer	The token is generated to recognise a customer for a longer period.	

Name	TokenValue
Definition	Payment token replacing the PAN of the payment card to identify the payment mean of the customer.
References	
Usage	
Type	TextString
Format	
CrossRef	CardAcquisitionResponse.PaymentInstrumentData.CardData.PaymentToken PaymentResponse.PaymentResult.PaymentInstrumentData.CardData.PaymentToken
XMLCoding	Attribute
ASN1Coding	409

Name	TotalAmount
Definition	Amount of a transaction.
References	
Usage	In the Loyalty Request message, it notifies the amount of the payment transaction on which the loyalty transaction has to be processed. In the Card Acquisition Request message, it allows the processing of a contactless card.
Type	SimpleAmountType
Format	
CrossRef	CardAcquisitionRequest.CardAcquisitionTransaction LoyaltyRequest.TransactionData
XMLCoding	Attribute
ASN1Coding	286

Name	TotalDetails
Definition	Indicates the hierarchical structure of the reconciliation result of the Sale to POI reconciliation.
References	
Usage	If present, this data element contains the classification of the transaction to compute the totals (see GetTotalsResponse.Totals).
Type	Cluster
Format	
CrossRef	GetTotalsRequest
XMLCoding	
ASN1Coding	287

Label	Description	Code
POIID	Give the totals result per POIID value.	
SaleID	Give the totals result per SaleID value.	
OperatorID	Give the totals result per OperatorID value.	
ShiftNumber	Give the totals result per ShiftNumber value.	
TotalsGroupID	Give the totals result per TotalsGroupID value.	

Name	TotalFeesAmount
Definition	Total amount of financial fees.
References	
Usage	Fees for financial services associated to the payment transaction.
Type	SimpleAmountType
Format	
CrossRef	PaymentResponse.PaymentResult.Amounts
XMLCoding	Attribute
ASN1Coding	97

Name	TotalFilter
<i>Definition</i>	Filter to compute the totals.
<i>References</i>	
<i>Usage</i>	Used for the Get Totals, to request totals for a (or a combination of) particular value of the POI Terminal, Sale Terminal, Cashier, Shift or TotalsGroupID.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	GetTotalsRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	288

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
POIID	[0..1]		If totals in the response have to be computed only for this particular value of POIID
SaleID	[0..1]		If totals in the response have to be computed only for this particular value of SaleID
OperatorID	[0..1]		If totals in the response have to be computed only for this particular value of OperatorID
ShiftNumber	[0..1]		If totals in the response have to be computed only for this particular value of ShiftNumber
TotalsGroupID	[0..1]		If totals in the response have to be computed only for this particular value of TotalsGroupID

Name	TotalsGroupID
<i>Definition</i>	Identification of a group of transaction on a POI Terminal, having the same Sale features.
<i>References</i>	
<i>Usage</i>	Could be used to group POI for reconciliation or other purpose defined by the Sale System. The default value is assigned by the Login Request. In other message, when the data has changed since or was not in the Login.
<i>Type</i>	TextString
<i>Format</i>	{1,16}
<i>CrossRef</i>	GetTotalsRequest.TotalFilter GetTotalsResponse.Totals.TransactionTotals LoginRequest.SaleTerminalData LoyaltyRequest.SaleData.SaleTerminalData PaymentRequest.SaleData.SaleTerminalData ReconciliationResponse.Totals.TransactionTotals StoredValueRequest.SaleData.POITerminalData
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	247

Name	TotalNbOfPayments
<i>Definition</i>	Total number of payments.
<i>References</i>	
<i>Usage</i>	For instalment, the number of payments, including the first one.
<i>Type</i>	Integer
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.PaymentData.Instalment
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	374

Name	TotalRebate
<i>Definition</i>	The global awarded amount that is not attached to an item.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.LoyaltyResult.Rebates LoyaltyResponse.LoyaltyResult.Rebates
<i>XMLCoding</i>	
<i>ASN1Coding</i>	289

Name	TotalRebatesAmount
<i>Definition</i>	Sum of rebates in amount (total amount or line item amount) for all the loyalty programs.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentResponse.PaymentResult.Amounts
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	290

Name	TrackData
Definition	Magnetic track or magnetic ink characters line.
References	ISO 7813 - ISO 4909
Usage	Generic data structure for a card track, used when the magstripe card reader is located on the Sale Terminal, or for magstripe Card Reader device request. The data structure is also used to store the line at the bottom of a bank check containing the bank account and the check number.
Type	defined data structure
Format	
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData CardAcquisitionResponse.PaymentInstrumentData.CardData.SensitiveCardData PaymentRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData CardReaderInitResponse.SensitiveCardData
XMLCoding	
ASN1Coding	291

Component	Mult.	Constraint	Rule
TrackNumb	[0..1]	default 2	
TrackFormat	[0..1]	default ISO	
TrackValue	[1..1]		

Name	TrackFormat
Definition	Card track format
References	
Usage	
Type	Enumeration
Format	
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData.TrackData BalanceInquiryResponse. PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData.TrackData CardAcquisitionResponse.PaymentInstrumentData.CardData.SensitiveCardData.TrackData PaymentRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData.TrackData CardReaderInitResponse.SensitiveCardData.TrackData
XMLCoding	Attribute
ASN1Coding	293

Label	Description	Code
ISO	ISO card track format - ISO 7813 - ISO 4909	
JIS-I	Japenese track format I	
JIS-II	Japenese track format II	
AAMVA	American driver license	
CMC-7	((Magnetic Ink Character Recognition, using the CMC-7 font - ISO 1004) Line at the bottom of a check containing the bank account and the check number.	
E-13B	(Magnetic Ink Character Recognition, using the E-13B font) Line at the bottom of a check containing the bank account and the check number.	

Name	TrackNumb
Definition	Card track number
References	ISO 7813 - ISO 4909
Usage	ISO track number
Type	Integer
Format	[1-3]
CrossRef	BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData.TrackData BalanceInquiryResponse. PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData.TrackData CardAcquisitionResponse.PaymentInstrumentData.CardData.SensitiveCardData.TrackData PaymentRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData.TrackData CardReaderInitResponse.SensitiveCardData.TrackData
XMLCoding	Attribute
ASN1Coding	292

Name	TrackValue
<i>Definition</i>	Card track content
<i>References</i>	ISO 7813 - ISO 4909
<i>Usage</i>	<p>For an ISO format:</p> <ul style="list-style-type: none"> Track 1 contains up to 73 alphanumeric data characters. Track 2 contains up to 37 alphanumeric data characters. Track 3 contains up to 104 alphanumeric characters. <p>For a CMC7 format:</p> <ul style="list-style-type: none"> Contains up to 35 alphanumeric data characters.
<i>Type</i>	TextString
<i>Format</i>	{1,104}
<i>CrossRef</i>	<p>BalanceInquiryRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData.TrackData</p> <p>BalanceInquiryResponse.</p> <p>PaymentResult.PaymentInstrumentData.CardData.SensitiveCardData.TrackData</p> <p>PaymentRequest.PaymentData.PaymentInstrumentData.CardData.SensitiveCardData.TrackData</p> <p>CardReaderInitResponse.SensitiveCardData.TrackData</p>
<i>XMLCoding</i>	Enclosing
<i>ASN1Coding</i>	294

Name	TrainingModeFlag
<i>Definition</i>	Training mode
<i>References</i>	
<i>Usage</i>	<p>This flag indicates to the POI that the entire session will be not used to make real transaction, but is used for test of system or operator training.</p> <p>The system configuration necessary to use this mode, as test cards for the transaction, or tests tools to simulate an Acquirer or TMS server, is out of the scope of the protocol.</p> <p>If the training or test mode cannot be set, the POI answers by a failure with the Login Response Message.</p>
<i>Type</i>	Boolean
<i>Format</i>	
<i>CrossRef</i>	LoginRequest
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	295

Name	TransmitRequest
Definition	Content of the Transmit Request message.
References	
Usage	It contains a message to transmit.
Type	defined data structure
Format	
CrossRef	TransmitRequest
XMLCoding	
ASN1Coding	401

Component	Mult.	Constraint	Rule
WaitResponseFlag	[0..1]	default False	
MaximumTransmitTime	[1..1]		
DestinationAddress	[1..1]		
Message	[1..1]		

Name	TransmitResponse
Definition	Content of the Transmit Response message.
References	
Usage	It conveys the response of the transmission.
Type	defined data structure
Format	
CrossRef	TransmitResponse
XMLCoding	
ASN1Coding	402

Component	Mult.	Constraint	Rule
Response	[1..1]		
Message	[0..1]		

Name	TransactionAmount
<i>Definition</i>	Sum of amount of processed transaction during the period.
<i>References</i>	
<i>Usage</i>	Total amount of transaction in the reconciliation result
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	GetTotalsResponse.Totals.TransactionTotals.PaymentTotals GetTotalsResponse.Totals.TransactionTotals.LoyaltyTotals ReconciliationResponse.Totals.TransactionTotals.PaymentTotals ReconciliationResponse.Totals.TransactionTotals.LoyaltyTotals
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	297

Name	TransactionConditions
<i>Definition</i>	Conditions on which the transaction must be processed.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData
<i>XMLCoding</i>	
<i>ASN1Coding</i>	296

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
AllowedPaymentBrand	[0..n]		Restrict brand if data sent
AcquirerID	[0..n]		Restrict to these Acquirer if present
DebitPreferredFlag	[0..1]	default False	The preferred type of payment is a debit transaction rather a credit transaction.
AllowedLoyaltyBrand	[0..n]		Restrict brand if data sent
LoyaltyHandling	[0..1]	default Forbidden	
CustomerLanguage	[0..1]		If the language is selected by the Sale System before the request to the POI.
ForceOnlineFlag	[0..1]	default False	Go online if data sent
ForceEntryMode	[0..n]		Restrict entry mode if sent
MerchantCategoryCode	[0..1]		The payment implies a specific MCC.

Name	TransactionCount
Definition	Number of processed transaction during the period.
References	
Usage	Total number of transaction in the reconciliation result
Type	Integer
Format	
CrossRef	GetTotalsResponse.Totals.TransactionTotals.PaymentTotals GetTotalsResponse.Totals.TransactionTotals.LoyaltyTotals ReconciliationResponse.Totals.TransactionTotals.PaymentTotals ReconciliationResponse.Totals.TransactionTotals.LoyaltyTotals
XMLCoding	Attribute
ASN1Coding	299

Name	TransactionID
Definition	Unique identification of a transaction
References	
Usage	To identify the transaction on the Sale System (e.g. ticket number), or the POI System.
Type	TextString
Format	
CrossRef	TransactionIdentificationType
XMLCoding	Attribute
ASN1Coding	298

Name	TransactionStatusRequest
<i>Definition</i>	Content of the TransactionStatus Request message.
<i>References</i>	
<i>Usage</i>	It conveys Information requested for status of the last or current Payment, Loyalty or Reversal transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	TransactionStatusRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	301

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
MessageReference	[0..1]		Present if it contains any data
ReceiptReprintFlag	[0..1]	default False	
DocumentQualifier	[0..2]		CustomerReceipt or CashierReceipt. Mandatory if ReceiptReprintFlag is True, otherwise absent.

Name	TransactionStatusResponse
<i>Definition</i>	Content of the TransactionStatus Response message.
<i>References</i>	
<i>Usage</i>	It conveys Information related to the status of the last or current Payment, Loyalty or Reversal transaction.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	TransactionStatusResponse
<i>XMLCoding</i>	
<i>ASN1Coding</i>	302

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
Response	[1..1]		
MessageReference	[0..1]		if Response.Result is Success
RepeatedMessageResponse	[0..1]		If Result is Success (process completed)

Name	TransactionTotals
Definition	Result of the Sale to POI Reconciliation processing.
References	
Usage	If Result is Success, contains all the totals, classified as required by the Sale in the message request. At least, transaction totals are provided per Acquirer, Acquirer Settlement, and Card Brand.
Type	defined data structure
Format	
CrossRef	GetTotalsResponse ReconciliationResponse
XMLCoding	
ASN1Coding	303

Component	Mult.	Constraint	Rule
PaymentInstrumentType	[1..1]		
AcquirerID	[0..1]		If available
ErrorCondition	[0..1]		if Response.Result is Partial, and the reconciliation with this Acquirer failed.
HostReconciliationID	[0..1]		If available
CardBrand	[0..1]		If configured to present totals per card brand, and Response.Result is Success
POIID	[0..1]		If requested in the message request
SaleID	[0..1]		If requested in the message request
OperatorID	[0..1]		If requested in the message request
ShiftNumber	[0..1]		If requested in the message request
TotalsGroupID	[0..1]		If requested in the message request
PaymentCurrency	[0..1]		
PaymentTotals	[0..10]		If both TransactionCount and TransactionAmount are not equal to zero
LoyaltyUnit	[0..1]	default Point	
LoyaltyCurrency	[0..1]		If LoyaltyUnit is Monetary
LoyaltyTotals	[0..6]		If both TransactionCount and TransactionAmount are not equal to zero

Name	TransactionAction
<i>Definition</i>	Action to realise on a transaction.
<i>References</i>	
<i>Usage</i>	On an EnableService request message: - Start a transaction by a swipe-ahead mechanism, with the services which are enabled - Abort a swipe-ahead transaction or started by a CardAcquisition, and not followed by a service request from the Sale System to complete the transaction.
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	EnableServiceRequest
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	304

<i>Label</i>	<i>Description</i>	<i>Code</i>
StartTransaction	Start a transaction by a swipe ahead mechanism, with the services which are enabled.	
AbortTransaction	Abort a transaction started either by a CardAcquisition or EnableService with TransactionAction to "StartTransaction", not followed by a service request from the Sale System to complete the transaction.	

Name	TransactionToPerform
<i>Definition</i>	Content of the Batch Request message.
<i>References</i>	
<i>Usage</i>	It conveys a collection of Payment, Loyalty and Reversal request in a batch message or file.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BatchRequest
<i>XMLCoding</i>	
<i>ASN1Coding</i>	379

<i>Or</i>	<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
{Or}	PaymentRequest	[0..1]		
	LoyaltyRequest	[0..1]		
Or}	ReversalRequest	[0..1]		

Name	TransactionType
<i>Definition</i>	Type of transaction for which totals are grouped.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	Enumeration
<i>Format</i>	
<i>CrossRef</i>	GetTotalsResponse.Totals.TransactionTotals.PaymentTotals GetTotalsResponse.Totals.TransactionTotals.LoyaltyTotals ReconciliationResponse.Totals.TransactionTotals.PaymentTotals ReconciliationResponse.Totals.TransactionTotals.LoyaltyTotals
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	305

<i>Label</i>	<i>Description</i>	<i>Code</i>
Debit	Payment Debit transactions (e.g. if PaymentType is "Normal")	
Credit	Payment Credit transactions (e.g. if PaymentType is "Refund")	
ReverseDebit	Payment Reversal Debit transactions	
ReverseCredit	Payment Reversal Credit transactions	
OneTimeReservation	Outstanding OneTimeReservation transactions, i.e. between OneTimeReservation and Completion	
CompletedDeffered	OneTimeReservation transactions which have been completed by the Completion.	
FirstReservation	Outstanding FirstReservation transactions, i.e. between FirstReservation and UpdateReservation or Completion	
UpdateReservation	Outstanding UpdateReservation transactions, i.e. between UpdateReservation and UpdateReservation or Completion	
CompletedReservation	Reservation transactions which have been completed by the Completion.	
CashAdvance	Cash Advance transactions.	
IssuerInstalment	Issuer instalment transactions.	
Declined	Transactions which has not been approved (Result = "Failure" and ErrorCondition = "Refusal").	
Failed	Transactions which have not successfully completed (Result = "Failure" and ErrorCondition not equal to "Refusal").	
Award	Loyalty Award Transaction	
ReverseAward	Loyalty Reversal Award Transaction	
Redemption	Loyalty Redemption Transaction	
ReverseRedemption	Loyalty Reversal Redemption Transaction	
Rebate	Loyalty Rebate Transaction	
ReverseRebate	Loyalty Reversal Rebate Transaction	

5.2.2.13 U-Z

Name	UnitOfMeasure
Definition	Unit of measure of a quantity
References	
Usage	
Type	Enumeration
Format	
CrossRef	PaymentRequest.TransactionData.SaleItem PaymentResponse.LoyaltyResult.Rebates.SaleItemRebate LoyaltyRequest.TransactionData.SaleItem LoyaltyResponse.LoyaltyResult.Rebates.SaleItemRebate StoredValueRequest.StoredValueData.SaleItemRebate
XMLCoding	
ASN1Coding	306

Label	Description	Code
Case	Case or Carton	
Foot	Foot	
UKGallon	Gallon (UK)	
USGallon	Gallon (US)	
Gram	Gram	
Inch	Inch	
Kilogram	Kilogram	
Pound	Pound	
Meter	Meter	
Centimetre	Centimetre	
Litre	Litre	
Centilitre	Centilitre	
Ounce	Ounce	
Quart	Quart	
Pint	Pint	
Mile	Mile	
Kilometre	Kilometre	
Yard	Yard	
Other	Other unit than the previous one	

Name	UnitPrice
<i>Definition</i>	Price per unit of product
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	PaymentRequest.TransactionData.SaleItem LoyaltyRequest.TransactionData.SaleItem StoredValueRequest.StoredValueData.SaleItemRebate
<i>XMLCoding</i>	
<i>ASN1Coding</i>	307

Name	UnitValue
<i>Definition</i>	Value of a coin or bill.
<i>References</i>	
<i>Usage</i>	
<i>Type</i>	SimpleAmountType
<i>Format</i>	
<i>CrossRef</i>	DiagnosisResponse.POIStatus.CashHandlingDevice.CoinsOrBills LoginResponse.POIStatus.CashHandlingDevice.CoinsOrBills
<i>XMLCoding</i>	Attribute
<i>ASN1Coding</i>	366

Name	UTMCoordinates
<i>Definition</i>	Location on the Earth specified by the Universal Transverse Mercator coordinate system.
<i>References</i>	
<i>Usage</i>	Identifies the geographic location of a mobile phone by GPS using the WGS84 ellipsoid spatial reference system.
<i>Type</i>	defined data structure
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation CardAcquisitionResponse.PaymentInstrumentData.MobileData.Geolocation PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation
<i>XMLCoding</i>	
<i>ASN1Coding</i>	357

<i>Component</i>	<i>Mult.</i>	<i>Constraint</i>	<i>Rule</i>
UTMZone	[1..1]		
UTMEastward	[1..1]		
UTMNorthward	[1..1]		

Name	UTMEastward
<i>Definition</i>	X-coordinate of the Universal Transverse Mercator coordinate system.
<i>References</i>	
<i>Usage</i>	UTM coordinates.
<i>Type</i>	TextString
<i>Format</i>	
<i>CrossRef</i>	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates CardAcquisitionResponse.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates
<i>XMLCoding</i>	
<i>ASN1Coding</i>	359

Name	UTMNorthward
Definition	Y-coordinate of the Universal Transverse Mercator coordinate system.
References	
Usage	UTM coordinates.
Type	TextString
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates CardAcquisitionResponse.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates
XMLCoding	
ASN1Coding	360

Name	UTMZone
Definition	UTM grid zone combination of the longitude zone (1 to 60) and the latitude band (C to X, excluding I and O).
References	
Usage	UTM coordinates.
Type	TextString
Format	
CrossRef	BalanceInquiryResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates BalanceInquiryRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates CardAcquisitionResponse.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates PaymentRequest.PaymentData.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates PaymentResponse.PaymentResult.PaymentInstrumentData.MobileData.Geolocation. UTMCoordinates
XMLCoding	
ASN1Coding	358

Name	ValidityDate
Definition	End of the validity period for the reservation.
References	
Usage	Allows a specific period for the reservation for the first reservation and the reservation updates as well.
Type	ISODate
Format	
CrossRef	PaymentResponse.PaymentResult
XMLCoding	Attribute
ASN1Coding	25

Name	WaitUserValidationFlag
Definition	Indicates that the user must confirm the entered characters, when the maximum allowed length is reached.
References	
Usage	During the processing of an Input command TString, DigitString or DecimalString with MaxLength or MaxDecimalLength present in the request.
Type	Boolean
Format	
CrossRef	InputRequest.InpuData
XMLCoding	Attribute
ASN1Coding	308

Name	WaitResponseFlag
Definition	Indicates that a response message has to be received.
References	
Usage	Processing of a Transmit request message.
Type	Boolean
Format	
CrossRef	TransmitRequest
XMLCoding	Attribute
ASN1Coding	403

Name	WarmResetFlag
Definition	Flag to request a warm reset on a chip.
References	
Usage	Allow to ask a warm reset to the chip already initialised.
Type	Boolean
Format	
CrossRef	CardReaderInitRequest
XMLCoding	Attribute
ASN1Coding	309

Name	X
Definition	Abscissa of a point coordinates.
References	
Usage	The hexadecimal value in text of the abscissa of the coordinates of a point. Leading zero can be removed (e.g. 3BC, 0, and 1287).
Type	TextString
Format	[0-9A-F]{1,4}
CrossRef	PaymentResponse.PaymentResult.CapturedSignature.AreaSize PaymentResponse.PaymentResult.CapturedSignature.SignaturePoint
XMLCoding	Attribute
ASN1Coding	350

Name	Y
Definition	Ordinate of a point coordinates.
References	
Usage	The hexadecimal value in text of the ordinate of the coordinates of a point. Leading zero can be removed (e.g. 3BC, 0, and 1287).
Type	TextString
Format	[0-9A-F]{1,4}
CrossRef	PaymentResponse.PaymentResult.CapturedSignature.AreaSize PaymentResponse.PaymentResult.CapturedSignature.SignaturePoint
XMLCoding	Attribute
ASN1Coding	351

2 5.2.3 Data Types

3

4

Name	AmountType
Definition	Common amount definition with currency
References	
Usage	Decimal unsigned amount with currency and amount before conversion.
Type	defined data structure
Format	

5

Component	Mult.	Constraint	Rule
AmountValue	[1..1]		
Currency	[0..1]		Mandatory if this data structure is used

6

7

Name	ISOCurrency3A
Definition	Currency identification
References	ISO 4217 "Codes for representation of currencies and funds", three contiguous capital letters
Usage	
Type	TextString
Format	{3,3}

8

9

Name	ISOCountry3A
Definition	Country Code
References	ISO 3166-1
Usage	Identification of the country according to the ISO 3166-1 codes, using three alpha characters.
Type	TextString
Format	{3,3}

10

11

Name	ISODATETIME
<i>Definition</i>	Date and Time
<i>References</i>	ISO 8601 W3C XML/Schema Part 2 - dateTime primitive data type
<i>Usage</i>	<p>This is the dateTime defined in the XML/Schema recommendation on the canonical form, derived from the ISO 8601 date/time extended standard format.</p> <p>The date/time format is: yyyy '-' mm '-' dd 'T' hh ':' mm ':' ss ('.' sss)? zzzzzz where</p> <ul style="list-style-type: none"> '-' is the character separator between the date elements, 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) 'T' is the character separator between the date and the time, ':' is the character separator between the time elements, hh is a 2-digits numeral representing the hours (from 00 to 23) mm (the second one) is a 2-digits numeral representing the minute (from 00 to 59) ss is a 2-digits numeral representing the integer part of the seconds (from 00 to 59) '.' is the character separator between the time and the fractional seconds, sss is a 1-digit to 3-digits numeral representing the fractional seconds, <p>zzzzzz represent the time zone which is the character 'Z' for a GMT time, or the delta from the GMT time, with a string of the form (('+' '-') hh ':' mm) where</p> <ul style="list-style-type: none"> '+' represent a positive delta from the GMT time '-' represent a negative delta from the GMT time hh is a 2-digits numeral representing the delta hours (from 00 to 14) mm is a 2-digits numeral representing the delta minute (from 00 to 59) <p>Requesting a mandatory time zone resolves the problem of Daylight Saving Time or Summer Time, because the time is absolute.</p> <p>Examples</p> <p>2008-04-12T23:20:50.275 represents the date of 12 April 2008 on the local time of 20 minutes, 50 seconds and 275 milliseconds past 23 hours.</p> <p>2008-04-12T22:20:50.275+01:00 represents the same date and time in Geneva.</p> <p>2008-04-12T17:20:50.275-05:00 represents the same date and time in New-York.</p>
<i>Type</i>	TextString
<i>Format</i>	{19,29}

12

13

Name	ISODATE
<i>Definition</i>	Date
<i>References</i>	ISO 8601 W3C XML/Schema Part 2 - dateTime primitive data type
<i>Usage</i>	<p>This is the date defined in the XML/Schema recommendation on the canonical form, derived from the ISO 8601 date.</p> <p>Example: 2008-04-12</p>
<i>Type</i>	TextString
<i>Format</i>	

14

15

Name ISOLanguage2A

Definition Language identification.

References ISO 639-1

Usage Identification of the language name according to the ISO 639-1 codes, using two alpha characters.

Type TextString

Format {2,2}

16

17

18

Name SimpleAmountType

Definition Common amount value definition

References EPAS Acquirer Protocol

Usage Decimal unsigned amount without an implied currency.

Amount value has a maximum of 8 digits at the left of the decimal point, and a maximum of 6 digits at the right of the decimal point.

The decimal point and the fractional part (i.e. at the right of the decimal point) are optional if the fractional part is equal to zero.

It is recommended to remove the digits equal to zero on left and the right of the value and any useless decimal point (ex. 00320.00 is expressed 320, and 56.10 is expressed 56.1)

Type Decimal

Format

19

20

Name TransactionIdentificationType

Definition Identification of a transaction for the Sale System or the POI System.

References

Usage

Type defined data structure

Format

21

Component	Mult.	Constraint	Rule
TransactionID	[1..1]		
TimeStamp	[1..1]		

22

23

24

6 Annex A Configuration Parameters

Considering the variety of architecture and implementation, there are necessary a lot of configurations not covered in this section. Presented implementations retain the terminology used in these concrete implementations, the linked to the wording we have defined in generic architecture is indicated when necessary.

6.1 Sale System Parameters

Sale and POI Systems Identification

The Sale and the POI Systems have to know identification of each system to allow exchange of Sale to POI messages.

Name	POI System Name
<i>Definition</i>	POI System Name identification.
<i>Usage</i>	Prefix of all the POI System components identifiers POIID
<i>Specification</i>	2.5.1 Identification of Systems and Components

Name	Sale System Name
<i>Definition</i>	Sale System Name identification.
<i>Usage</i>	Prefix of all the Sale System components identifiers SaleID
<i>Specification</i>	2.5.1 Identification of Systems and Components

Terminals Identification and Logical Connections

For each Terminal to Terminal relationship, the type of link and the identification of the remote Terminal:

Name	Terminal Relation
<i>Definition</i>	Type of Terminal to Terminal relationship.
<i>Usage</i>	Transport connection and message dialogue.
<i>Specification</i>	2.5.4 Logical Connections Between Terminals

Label	Description	
<i>Connected</i>	Connected POI System	
<i>Term2Term</i>	Terminal to Terminal link.	
<i>Term2Serv</i>	Terminal to Server link.	
	<i>Serv2Term</i>	Server to Terminal link.
	<i>Serv2Serv</i>	Server to Server link.

Name	Sale (or POI) Terminal Identifier
<i>Definition</i>	Identification of the Terminal.
<i>Usage</i>	Logical addressing of the Terminal.
<i>Specification</i>	2.5.1 Identification of Systems and Components

Servers Identification and Logical Connections

The same information for the Server to Server relationship:

Name	Server Relation
<i>Definition</i>	Type of Server to Server relationship.
<i>Usage</i>	Transport connection and message dialogue.
<i>Specification</i>	2.5.5 Logical Connections Between Servers

Label	Description
<i>Connected</i>	Connected POI System
<i>Direct</i>	Direct Server to Server link.
<i>Indirect</i>	Indirect Server to Server link.

Name	Sale (or POI) Server Identifier
<i>Definition</i>	Identification of the Server.
<i>Usage</i>	Logical addressing of the Server.
<i>Specification</i>	2.5.1 Identification of Systems and Components

Transport Service

The Sale and the POI Systems have the following configuration parameters for the transport service management:

Name	TC2
<i>Definition</i>	Application message reception timer
<i>Usage</i>	This timer is armed at the reception of the application message prefix to supervise the reception of the complete application message. It allows the detection of an incomplete application message reception, avoiding a deadlock of the transport connection.
<i>Specification</i>	3.2.2 Data Transfer 3.2.4 Transport Error Handling (Message Too Big, Incomplete Application Message)

Name	Max Number of Connections
<i>Definition</i>	Maximum number of transport connections the Sale or The POI System component can manage simultaneously.
<i>Usage</i>	To avoid error on opening too many transport connections, from the application protocol layer or the remote application protocol.
<i>Specification</i>	3.2.4 Transport Error Handling (Unable to Establish a Transport Connection, Max Global Number of Connections)

Name	Max Message Size
<i>Definition</i>	Maximum size of a message the Sale or The POI System component can receive.
<i>Usage</i>	To detect error of messages length, in particular synchronisation of messages exchange, with the prefix length and the content of the message.
<i>Specification</i>	3.2.4 Transport Error Handling (Message Too Big)

Sale Transport Service

The Sale System has in addition the following configuration parameters for the transport service management:

Name	TC1
Definition	Transport connection establishment timer
Usage	After the sending of a transport connection request, the Requestor arms the TC1 timer to watch on the response from the Responder. TC1 is reset on the transport connection confirmation reception.
Specification	3.2.1 Transport Connection Handling 3.2.4 Transport Error Handling (Unable to Establish a Transport Connection)

Transport Connection

For each Terminal to Terminal relationship, and the Server to Server if any:

Name	Number of Connections
Definition	Number of transport connections the Sale System component may establish simultaneously toward this POI System component.
Usage	To avoid error on opening too many transport connections between two components. The number depends on the type of relation between these two components defined by the parameter "Terminal Relation" and "Server Relation" in 2.6 Configuration and Examples Architecture Configuration Parameters.
Specification	3.2.4 Transport Error Handling (Other Errors)

Name	POI Component Address
Definition	Transport address of the POI Component
Usage	This (or these) transport address is used by the Sale System to establish a transport connection with a POI System component.
Specification	3.2.3 Addressing

Sale and POI Protocol Version

Name	POI Version
Definition	Minimum and maximum protocol version that the POI can manage.
Usage	Protocol version management.
Specification	3.4.3 Protocol Version Management

Name	Sale Version
Definition	Minimum and maximum protocol version that the Sale can manage.
Usage	Protocol version management.
Specification	3.4.3 Protocol Version Management

Sale and POI Terminal Profiles

Name	Generic Profile
Definition	Type of interface the Sale or the POI Terminal can manage.
Usage	Indicates what group of messages of the protocol are used or provided
Specification	4.1.2 Profiles

Name	Service Profile
Definition	List of optional services the Sale or the POI Terminal can manage.
Usage	List (possibly empty) of services which are implemented either in term of messages, either in term of optional part of the messages.
Specification	4.1.2 Profiles

Server Login

Name	Server Login
Definition	The Sale System sends a Login to the POI System before any Terminal Login.
Usage	Allows a dialogue between the Sale Server and the POI Server.
Specification	4.2.1 Session Management and 4.2.3 Login

Time Synchronisation

Name	Time Synchronisation
Definition	The POI System and POI Terminals synchronise their clock at the Login.
Usage	
Specification	4.2.1 Session Management and 4.2.3 Login

Reconciliation Details

The Sale System and the POI System have the following configuration parameters:

Name	Reconciliation Details
<i>Definition</i>	Details of the transaction totals sent in the reconciliation response: per card brand per POI/Acquirer reconciliation period identification per POI Terminal per Sale Terminal per Cashier per shift per TotalsGroupID
<i>Usage</i>	Allows computation of transaction totals according to the various criteria.
<i>Specification</i>	4.4.1.4 Reconciliation Processing

Event Notification Outside Sessions

Name	Event Notification Outside Sessions
<i>Definition</i>	The POI Server and the POI Terminals may send Event Notification outside sessions.
<i>Usage</i>	Allows the Sale System to know activity and problem occurring in the POI System.
<i>Specification</i>	4.7.3 Event Notification Message

6.2 POI System Parameters

Sale and POI Systems Identification

The Sale and the POI Systems have to know identification of each system to allow exchange of Sale to POI messages.

Name	POI System Name
<i>Definition</i>	POI System Name identification.
<i>Usage</i>	Prefix of all the POI System components identifiers POIID
<i>Specification</i>	2.5.1 Identification of Systems and Components

Name	Sale System Name
<i>Definition</i>	Sale System Name identification.
<i>Usage</i>	Prefix of all the Sale System components identifiers SaleID
<i>Specification</i>	2.5.1 Identification of Systems and Components

Terminals Identification and Logical Connections

For each Terminal to Terminal relationship, the type of link and the identification of the remote Terminal:

Name	Terminal Relation
<i>Definition</i>	Type of Terminal to Terminal relationship.
<i>Usage</i>	Transport connection and message dialogue.
<i>Specification</i>	2.5.4 Logical Connections Between Terminals

Label	Description	
<i>Connected</i>	Connected POI System	
<i>Term2Term</i>	Terminal to Terminal link.	
<i>Term2Serv</i>	Terminal to Server link.	
	<i>Serv2Term</i>	Server to Terminal link.
	<i>Serv2Serv</i>	Server to Server link.

Name	Sale (or POI) Terminal Identifier
<i>Definition</i>	Identification of the Terminal.
<i>Usage</i>	Logical addressing of the Terminal.
<i>Specification</i>	2.5.1 Identification of Systems and Components

Servers Identification and Logical Connections

The same information for the Server to Server relationship:

Name	Server Relation
<i>Definition</i>	Type of Server to Server relationship.
<i>Usage</i>	Transport connection and message dialogue.
<i>Specification</i>	2.5.5 Logical Connections Between Servers

Label	Description
<i>Connected</i>	Connected POI System
<i>Direct</i>	Direct Server to Server link.
<i>Indirect</i>	Indirect Server to Server link.

Name	Sale (or POI) Server Identifier
<i>Definition</i>	Identification of the Server.
<i>Usage</i>	Logical addressing of the Server.
<i>Specification</i>	2.5.1 Identification of Systems and Components

Transport Service

The Sale and the POI Systems have the following configuration parameters for the transport service management:

Name	TC2
<i>Definition</i>	Application message reception timer
<i>Usage</i>	This timer is armed at the reception of the application message prefix to supervise the reception of the complete application message. It allows the detection of an incomplete application message reception, avoiding a deadlock of the transport connection.
<i>Specification</i>	3.2.2 Data Transfer 3.2.4 Transport Error Handling (Message Too Big, Incomplete Application Message)

Name	Max Number of Connections
<i>Definition</i>	Maximum number of transport connections the Sale or The POI System component can manage simultaneously.
<i>Usage</i>	To avoid error on opening too many transport connections, from the application protocol layer or the remote application protocol.
<i>Specification</i>	3.2.4 Transport Error Handling (Unable to Establish a Transport Connection, Max Global Number of Connections)

Name	Max Message Size
<i>Definition</i>	Maximum size of a message the Sale or The POI System component can receive.
<i>Usage</i>	To detect error of messages length, in particular synchronisation of messages exchange, with the prefix length and the content of the message.
<i>Specification</i>	3.2.4 Transport Error Handling (Message Too Big)

Transport Connection

For each Terminal to Terminal relationship, and the Server to Server if any:

Name	Number of Connections
Definition	Number of transport connections the Sale System component may establish simultaneously toward this POI System component.
Usage	To avoid error on opening too many transport connections between two components. The number depends on the type of relation between these two components defined by the parameter "Terminal Relation" and "Server Relation" in 2.6 Configuration and Examples Architecture Configuration Parameters.
Specification	3.2.4 Transport Error Handling (Other Errors)

Name	POI Component Address
Definition	Transport address of the POI Component
Usage	This (or these) transport address is used by the Sale System to establish a transport connection with a POI System component.
Specification	3.2.3 Addressing

Sale and POI Protocol Version

Name	POI Version
Definition	Minimum and maximum protocol version that the POI can manage.
Usage	Protocol version management.
Specification	3.4.3 Protocol Version Management

Name	Sale Version
Definition	Minimum and maximum protocol version that the Sale can manage.
Usage	Protocol version management.
Specification	3.4.3 Protocol Version Management

Sale and POI Terminal Profiles

Name	Generic Profile
Definition	Type of interface the Sale or the POI Terminal can manage.
Usage	Indicates what group of messages of the protocol are used or provided
Specification	4.1.2 Profiles

Name	Service Profile
Definition	List of optional services the Sale or the POI Terminal can manage.
Usage	List (possibly empty) of services which are implemented either in term of messages, either in term of optional part of the messages.
Specification	4.1.2 Profiles

Time Synchronisation

Name	Time Synchronisation
Definition	The POI System and POI Terminals synchronise their clock at the Login.
Usage	
Specification	4.2.1 Session Management and 4.2.3 Login

Reconciliation Details

The Sale System and the POI System have the following configuration parameters:

Name	Reconciliation Details
<i>Definition</i>	Details of the transaction totals sent in the reconciliation response: per card brand per POI/Acquirer reconciliation period identification per POI Terminal per Sale Terminal per Cashier per <i>shift</i> per TotalsGroupID
<i>Usage</i>	Allows computation of transaction totals according to the various criteria.
<i>Specification</i>	4.4.1.4 Reconciliation Processing

Event Notification Outside Sessions

Name	Event Notification Outside Sessions
<i>Definition</i>	The POI Server and the POI Terminals may send Event Notification outside sessions.
<i>Usage</i>	Allows the Sale System to know activity and problem occurring in the POI System.
<i>Specification</i>	4.7.3 Event Notification Message

7 Annex B Messages Examples

This section presents examples of the message building and encoding:

- Transported messages with the various data codings and transport protocols.
- Security of messages
- Encoded messages examples of the specifications

7.1 Transported Message Examples

We consider the following Logout Request message:

SaleToPOIRequest	
MessageHeader	
MessageClass	Service
MessageCategory	Logout
MessageType	Request
ServiceID	613
SaleID	SaleTermA
POIID	POITerm1
LogoutRequest	

7.1.1 XML and JSON Coding of the Message

XML Coding

For the message presented above, the XML coding is:

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Logout"
    MessageType="Request" ServiceID="613" SaleID="SaleTermA" POIID="POITerm1"/>
  <LogoutRequest/>
</SaleToPOIRequest>
```

The XML canonical form is presented below, all the useless spaces, tabulations and ends of line are stripped, the length of the message is 341 bytes, or 0155 in hexadecimal.

0000	3C	3F	78	6D	6C	20	76	65	72	73	69	6F	6E	3D	22	31	<?xml version="1
0010	2E	30	22	20	65	6E	63	6F	64	69	6E	67	3D	22	55	54	.0" encoding="UT
0020	46	2D	38	22	3F	3E	3C	53	61	6C	65	54	6F	50	4F	49	F-8"?><SaleToPOI
0030	52	65	71	75	65	73	74	20	78	6D	6C	6E	73	3A	78	73	Request xmlns:xs
0040	69	3D	22	68	74	74	70	3A	2F	2F	77	77	77	2E	77	33	i="http://www.w3
0050	2E	6F	72	67	2F	32	30	30	31	2F	58	4D	4C	53	63	68	.org/2001/XMLSchema-
0060	65	6D	61	2D	69	6E	73	74	61	6E	63	65	22	20	78	73	ema-instance" xs
0070	69	3A	6E	6F	4E	61	6D	65	73	70	61	63	65	53	63	68	i:noNamespaceSch
0080	65	6D	61	4C	6F	63	61	74	69	6F	6E	3D	22	45	70	61	emaLocation="Epa
0090	73	53	61	6C	65	54	6F	50	4F	49	4D	65	73	73	61	67	ssSaleToPOIMessag
00A0	65	73	2E	78	73	64	22	3E	3C	4D	65	73	73	61	67	65	es.xsd"><Message
00B0	48	65	61	64	65	72	20	4D	65	73	73	61	67	65	43	6C	Header MessageCl
00C0	61	73	73	3D	22	53	65	72	76	69	63	65	22	20	4D	65	ass="Service" Me
00D0	73	73	61	67	65	43	61	74	65	67	6F	72	79	3D	22	4C	ssageCategory="L
00E0	6F	67	6F	75	74	22	20	4D	65	73	73	61	67	65	54	79	ogout" MessageTy
00F0	70	65	3D	22	52	65	71	75	65	73	74	22	20	53	65	72	pe="Request" Ser
0100	76	69	63	65	49	44	3D	22	36	31	33	22	20	53	61	6C	viceID="613" Sal
0110	65	49	44	3D	22	53	61	6C	65	54	65	72	6D	41	22	20	eID="SaleTermA"
0120	50	4F	49	49	44	3D	22	50	4F	49	54	65	72	6D	31	22	POIID="POITerm1"
0130	2F	3E	3C	4C	6F	67	6F	75	74	52	65	71	75	65	73	74	/><LogoutRequest
0140	2F	3E	3C	2F	53	61	6C	65	54	6F	50	4F	49	52	65	71	/></SaleToPOIReq
0150	75	65	73	74	3E												uest>

JSON Coding

For the message presented above, the JSON coding is:

```
{  
    "SaleToPOIRequest": {  
        "MessageHeader": {  
            "ProtocolVersion": "2.1",  
            "MessageClass": "Service",  
            "MessageCategory": "Login",  
            "MessageType": "Request",  
            "ServiceID": "498",  
            "SaleID": "SaleTermA",  
            "POIID": "POITerm1"  
        },  
        "LogoutRequest": {}  
    }  
}
```

The JSON canonical form is presented below, all the useless spaces, tabulations and ends of line are stripped, the length of the message is 216 bytes, or 00D8 in hexadecimal.

0000	7B	22	53	61	6C	65	54	6F	50	4F	49	52	65	71	75	65	{"SaleToPOIReque
0010	73	74	22	3A	7B	22	4D	65	73	73	61	67	65	48	65	61	st": {"MessageHea
0020	64	65	72	22	3A	7B	22	50	72	6F	74	6F	63	6F	6C	56	der": {"ProtocolV
0030	65	72	73	69	6F	6E	22	3A	22	32	2E	31	22	2C	22	4D	ersion": "2.1", "M
0040	65	73	73	61	67	65	43	6C	61	73	73	22	3A	22	53	65	essageClass": "Se
0050	72	76	69	63	65	22	2C	22	4D	65	73	73	61	67	65	43	rvice", "MessageC
0060	61	74	65	67	6F	72	79	22	3A	22	4C	6F	67	69	6E	22	ategory": "Login"
0070	2C	22	4D	65	73	73	61	67	65	54	79	70	65	22	3A	22	, "MessageType": "
0080	52	65	71	75	65	73	74	22	2C	22	53	65	72	76	69	63	Request", "Servic
0090	65	49	44	22	3A	22	34	39	38	22	2C	22	53	61	6C	65	eID": "498", "Sale
00A0	49	44	22	3A	22	53	61	6C	65	54	65	72	6D	41	22	2C	ID": "SaleTermA",
00B0	22	50	4F	49	49	44	22	3A	22	50	4F	49	54	65	72	6D	"POIID": "POITerm
00C0	31	22	7D	2C	22	4C	6F	67	6F	75	74	52	65	71	75	65	1"}, "LogoutReque
00D0	73	74	22	3A	7B	7D	7D	7D									st": {}}}

7.1.2 TCP Transport of the Message

XML Coding

The prefix to add for the message length is:

00 00 01 55

So the message sent by the TCP transport protocol start with this prefix length, a dump of the message sent by the transport protocol is presented below.

0000	00	00	01	55	3C	3F	78	6D	6C	20	76	65	72	73	69	6F	...U<?xml versio
0010	6E	3D	22	31	2E	30	22	20	65	6E	63	6F	64	69	6E	67	n="1.0" encoding
0020	3D	22	55	54	46	2D	38	22	3F	3E	3C	53	61	6C	65	54	="UTF-8"?><SaleT
0030	6F	50	4F	49	52	65	71	75	65	73	74	20	78	6D	6C	6E	oPOIRequest xmln
0040	73	3A	78	73	69	3D	22	68	74	74	70	3A	2F	2F	77	77	s:xsi="http://ww
0050	77	2E	77	33	2E	6F	72	67	2F	32	30	30	31	2F	58	4D	w.w3.org/2001/XM
0060	4C	53	63	68	65	6D	61	2D	69	6E	73	74	61	6E	63	65	LSchema-instance
0070	22	20	78	73	69	3A	6E	6F	4E	61	6D	65	73	70	61	63	" xsi:noNamespac
0080	65	53	63	68	65	6D	61	4C	6F	63	61	74	69	6F	6E	3D	eSchemaLocation=
0090	22	45	70	61	73	53	61	6C	65	54	6F	50	4F	49	4D	65	"EpasSaleToPOIMe
00A0	73	73	61	67	65	73	2E	78	73	64	22	3E	3C	4D	65	73	ssages.xsd"><Mes
00B0	73	61	67	65	48	65	61	64	65	72	20	4D	65	73	73	61	sageHeader Messa
00C0	67	65	43	6C	61	73	73	3D	22	53	65	72	76	69	63	65	geClass="Service
00D0	22	20	4D	65	73	73	61	67	65	43	61	74	65	67	6F	72	" MessageCategor
00E0	79	3D	22	4C	6F	67	6F	75	74	22	20	4D	65	73	73	61	y="Logout" Messa
00F0	67	65	54	79	70	65	3D	22	52	65	71	75	65	73	74	22	geType="Request"
0100	20	53	65	72	76	69	63	65	49	44	3D	22	36	31	33	22	ServiceID="613"
0110	20	53	61	6C	65	49	44	3D	22	53	61	6C	65	54	65	72	SaleID="SaleTer
0120	6D	41	22	20	50	4F	49	49	44	3D	22	50	4F	49	54	65	mA" POIID="POITe
0130	72	6D	31	22	2F	3E	3C	4C	6F	67	6F	75	74	52	65	71	rml"/><LogoutReq
0140	75	65	73	74	2F	3E	3C	2F	53	61	6C	65	54	6F	50	4F	uest/></SaleToPO
0150	49	52	65	71	75	65	73	74	3E								IRequest>

JSON Coding

The prefix to add for the message length is:

00 00 00 D8

So the message sent by the TCP transport protocol start with this prefix length, a dump of the message sent by the transport protocol is presented below.

0000	00	00	00	D8	7B	22	53	61	6C	65	54	6F	50	4F	49	52 {"SaleToPOIR
0010	65	71	75	65	73	74	22	3A	7B	22	4D	65	73	73	61	67	equest": {"Messag
0020	65	48	65	61	64	65	72	22	3A	7B	22	50	72	6F	74	6F	eHeader": {"Proto
0030	63	6F	6C	56	65	72	73	69	6F	6E	22	3A	22	32	2E	30	colVersion": "2.0
0040	22	2C	22	4D	65	73	73	61	67	65	43	6C	61	73	73	22	", "MessageClass"
0050	3A	22	53	65	72	76	69	63	65	22	2C	22	4D	65	73	73	:"Service", "Mess
0060	61	67	65	43	61	74	65	67	6F	72	79	22	3A	22	4C	6F	ageCategory": "Lo
0070	67	69	6E	22	2C	22	4D	65	73	73	61	67	65	54	79	70	gin", "MessageTyp
0080	65	22	3A	22	52	65	71	75	65	73	74	22	2C	22	53	65	e": "Request", "Se
0090	72	76	69	63	65	49	44	22	3A	22	34	39	38	22	2C	22	rviceID": "498", "
00A0	53	61	6C	65	49	44	22	3A	22	53	61	6C	65	54	65	72	SaleID": "SaleTer
00B0	6D	41	22	2C	22	50	4F	49	49	44	22	3A	22	50	4F	49	mA", "POIID": "POI
00C0	54	65	72	6D	31	22	7D	2C	22	4C	6F	67	6F	75	74	52	Term1"}, "LogoutR
00D0	65	71	75	65	73	74	22	3A	7B	7D	7D	7D					equest": {} } }

7.1.3 HTTP Transport of the Message

This section presents the XML and JSON Logout request examples of HTTP message.

XML Coding

The prefix to add for the message length is:

```
POST /EPASSaleToPOI/3.0/LogoutRequest?
  Class=Service&Category=Logout&SaleID=SaleTermA&POIID=POITerm1 HTTP/1.1
  Host: test.epasorg.eu:8080
  User-Agent: EPASTest/2.1 (Windows NT 6.1)
  Accept: text/xml
  Accept-charset: utf-8
  Content-type: text/xml; charset=utf-8
  Content-length: 341

  <?xml version="1.0" encoding="UTF-8"?>
  <SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
    <MessageHeader MessageClass="Service" MessageCategory="Logout"
      MessageType="Request" ServiceID="613" SaleID="SaleTermA" POIID="POITerm1"/>
    <LogoutRequest/>
  </SaleToPOIRequest>
```

So the message sent by the TCP transport protocol start with this prefix length, a dump of the message sent by the transport protocol is presented below.

0000	50	4F	53	54	20	2F	45	50	41	53	53	61	6C	65	54	6F	POST /EPASSaleTo
0010	50	4F	49	2F	32	2E	31	2F	4C	6F	67	6F	75	74	52	65	POI/2.1/LogoutRe
0020	71	75	65	73	74	3F	20	43	6C	61	73	73	3D	53	65	72	quest? Class=Ser
0030	76	69	63	65	26	43	61	74	65	67	6F	72	79	3D	4C	6F	vice&Category=Lo
0040	67	6F	75	74	26	53	61	6C	65	49	44	3D	53	61	6C	65	gout&SaleID=Sale
0050	54	65	72	6D	41	26	50	4F	49	49	44	3D	50	4F	49	54	TermA&POIID=POIT
0060	65	72	6D	31	20	48	54	54	50	2F	31	2E	31	0D	0A	48	erm1 HTTP/1.1..H
0070	6F	73	74	3A	20	74	65	73	74	2E	65	70	61	73	6F	72	ost: test.epasor
0080	67	2E	65	75	3A	38	30	38	30	0D	0A	55	73	65	72	2D	g.eu:8080..User-
0090	41	67	65	6E	74	3A	20	45	50	41	53	54	65	73	74	2F	Agent: EPASTest/
00A0	32	2E	31	20	28	57	69	6E	64	6F	77	73	20	4E	54	20	2.1 (Windows NT
00B0	36	2E	31	29	0D	0A	41	63	63	65	70	74	3A	20	74	65	6.1)..Accept: te
00C0	78	74	2F	78	6D	6C	0D	0A	41	63	63	65	70	74	2D	63	xt/xml..Accept-c
00D0	68	61	72	73	65	74	3A	20	75	74	66	2D	38	0D	0A	43	harset: utf-8..C
00E0	6F	6E	74	65	6E	74	2D	74	79	70	65	3A	20	74	65	78	ontent-type: tex
00F0	74	2F	78	6D	6C	3B	20	63	68	61	72	73	65	74	3D	75	t/xml; charset=u
0100	74	66	2D	38	0D	0A	43	6F	6E	74	65	6E	74	2D	6C	65	tf-8..Content-le
0110	6E	67	74	68	3A	20	33	34	31	0D	0A	0D	0A	3C	3F	78	ngth: 341....<?x
0120	6D	6C	20	76	65	72	73	69	6F	6E	3D	22	31	2E	30	22	ml version="1.0"
0130	20	65	6E	63	6F	64	69	6E	67	3D	22	55	54	46	2D	38	encoding="UTF-8"
0140	22	3F	3E	3C	53	61	6C	65	54	6F	50	4F	49	52	65	71	"?><SaleToPOIReq
0150	75	65	73	74	20	78	6D	6C	6E	73	3A	78	73	69	3D	22	uest xmlns:xsi="
0160	68	74	74	70	3A	2F	77	77	77	2E	77	33	2E	6F	72	http://www.w3.or	
0170	67	2F	32	30	30	31	2F	58	4D	4C	53	63	68	65	6D	61	g/2001/XMLSchema
0180	2D	69	6E	73	74	61	6E	63	65	22	20	78	73	69	3A	6E	-instance" xsi:n
0190	6F	4E	61	6D	65	73	70	61	63	65	53	63	68	65	6D	61	oNamespaceSchema
01A0	4C	6F	63	61	74	69	6F	6E	3D	22	45	70	61	73	53	61	Location="EpasSa
01B0	6C	65	54	6F	50	4F	49	4D	65	73	73	61	67	65	73	2E	leToPOIMessages.
01C0	78	73	64	22	3E	3C	4D	65	73	73	61	67	65	48	65	61	xsd"><MessageHea
01D0	64	65	72	20	4D	65	73	73	61	67	65	43	6C	61	73	73	der MessageClass
01E0	3D	22	53	65	72	76	69	63	65	22	20	4D	65	73	73	61	="Service" Messa
01F0	67	65	43	61	74	65	67	6F	72	79	3D	22	4C	6F	67	6F	geCategory="Logo
0200	75	74	22	20	4D	65	73	73	61	67	65	54	79	70	65	3D	ut" MessageType=
0210	22	52	65	71	75	65	73	74	22	20	53	65	72	76	69	63	"Request" Servic
0220	65	49	44	3D	22	36	31	33	22	20	53	61	6C	65	49	44	eID="613" SaleID
0230	3D	22	53	61	6C	65	54	65	72	6D	41	22	20	50	4F	49	="SaleTermA" POI
0240	49	44	3D	22	50	4F	49	54	65	72	6D	31	22	2F	3E	3C	ID="POITerm1"/><
0250	4C	6F	67	6F	75	74	52	65	71	75	65	73	74	2F	3E	3C	LogoutRequest/><
0260	2F	53	61	6C	65	54	6F	50	4F	49	52	65	71	75	65	73	/SaleToPOIReques
0270	74	3E															t>

JSON Coding

The prefix to add for the message length is:

```
POST
/EPASSaleToPOI/2.1/LogoutRequest?MessageClass=Service&MessageCategory=Logout&SaleID=SaleTermA&POIID=POITerm1 HTTP/1.1
Cache-Control: no-cache
Host: test.epasorg.eu:8080
User-Agent: EPASTest/2.1 (Windows NT 6.1)
Accept: application/json
Content-type: application/json; charset=utf-8
Content-length: 216

{
  "SaleToPOIRequest": {
    "MessageHeader": {
      "ProtocolVersion": "2.1",
      "MessageClass": "Service",
      "MessageCategory": "Logout",
      "MessageType": "Request",
      "ServiceID": "498",
      "SaleID": "SaleTermA",
      "POIID": "POITerm1"
    },
    "LogoutRequest": {}
  }
}
```

So the message sent by the TCP transport protocol start with this prefix length, a dump of the message sent by the transport protocol is presented below.

0000	50 4F 53 54 20 2F 45 50 41 53 53 61 6C 65 54 6F	POST /EPASSaleTo
0010	50 4F 49 2F 32 2E 31 2F 4C 6F 67 6F 75 74 52 65	POI/2.1/LogoutRe
0020	71 75 65 73 74 3F 4D 65 73 73 61 67 65 43 6C 61	quest?MessageCla
0030	73 73 3D 53 65 72 76 69 63 65 26 4D 65 73 73 61	ss=Service&Messa
0040	67 65 43 61 74 65 67 6F 72 79 3D 4C 6F 67 6F 75	geCategory=Logou
0050	74 26 53 61 6C 65 49 44 3D 53 61 6C 65 54 65 72	t&SaleID=SaleTer
0060	6D 41 26 50 4F 49 49 44 3D 50 4F 49 54 65 72 6D	mA&POIID=POITerm
0070	31 20 48 54 54 50 2F 31 2E 31 0D 0A 43 61 63 68	1 HTTP/1.1..Cach
0080	65 2D 43 6F 6E 74 72 6F 6C 3A 20 6E 6F 2D 63 61	e-Control: no-ca
0090	63 68 65 0D 0A 48 6F 73 74 3A 20 74 65 73 74 2E	che..Host: test.
00A0	65 70 61 73 6F 72 67 2E 65 75 3A 38 30 38 30 0D	epasorg.eu:8080.
00B0	0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 45 50 41	.User-Agent: EPA
00C0	53 54 65 73 74 2F 32 2E 31 20 28 57 69 6E 64 6F	STTest/2.1 (Windo
00D0	77 73 20 4E 54 20 36 2E 31 29 0D 0A 41 63 63 65	ws NT 6.1)..Accel
00E0	70 74 3A 20 61 70 70 6C 69 63 61 74 69 6F 6E 2F	pt: application/
00F0	6A 73 6F 6E 0D 0A 43 6F 6E 74 65 6E 74 2D 74 79	json..Content-typ
0100	70 65 3A 20 61 70 70 6C 69 63 61 74 69 6F 6E 2F	pe: application/
0110	6A 73 6F 6E 2B 20 63 68 61 72 73 65 74 3D 75 74	json; charset=ut
0120	66 2D 38 0D 0A 43 6F 6E 74 65 6E 74 2D 6C 65 6E	f-8..Content-len
0130	67 74 68 3A 20 32 31 36 0D 0A 0D 0A 7B 22 53 61	gth: 216....{"Sa
0140	6C 65 54 6F 50 4F 49 52 65 71 75 65 73 74 22 3A	leToPOIRequest":
0150	7B 22 4D 65 73 73 61 67 65 48 65 61 64 65 72 22	{"MessageHeader":
0160	3A 7B 22 50 72 6F 74 6F 63 6F 6C 56 65 72 73 69	:{"ProtocolVersi
0170	6F 6E 22 3A 22 32 2E 31 22 2C 22 4D 65 73 73 61	on": "2.1", "Messa
0180	67 65 43 6C 61 73 73 22 3A 22 53 65 72 76 69 63	geClass": "Servic
0190	65 22 2C 22 4D 65 73 73 61 67 65 43 61 74 65 67	e", "MessageCateg
01A0	6F 72 79 22 3A 22 4C 6F 67 69 6E 22 2C 22 4D 65	ory": "Login", "Me
01B0	73 73 61 67 65 54 79 70 65 22 3A 22 52 65 71 75	ssageType": "Requ
01C0	65 73 74 22 2C 22 53 65 72 76 69 63 65 49 44 22	est", "ServiceID":
01D0	3A 22 34 39 38 22 2C 22 53 61 6C 65 49 44 22 3A	:"498", "SaleID":
01E0	22 53 61 6C 65 54 65 72 6D 41 22 2C 22 50 4F 49	"SaleTermA", "POI
01F0	49 44 22 3A 22 50 4F 49 54 65 72 6D 31 22 7D 2C	ID": "POITerm1"},
0200	22 4C 6F 67 6F 75 74 52 65 71 75 65 73 74 22 3A	"LogoutRequest":
0210	7B 7D 7D 7D	{}}}

7.2 Key Encryption Example

The Key Encryption Key is the transport key, named "SpecV1TestDATKey", with the version "2010060715", and the value: "37233E89 0B0104E9 BC943D0E 45EAE5A7"

The generated session key with parity has the value: "E64AEADA 2A6E34B6 DF790DE3 0E46E9BF"

The session key, encrypted by the KEK using Triple-DES in CBC mode, has the value: "9CF4E2DE 12A260E4 5D082D5D CCE4D050"

The *Recipient* data structure is presented in the table below.

Recipient	
KEK	
Version	v4
KEKIdentifier	
KeyIdentifier	SpecV1TestDATKey
KeyVersion	2010060715
KeyEncryptionAlgorithm	
Algorithm	des-ed3-cbc
EncryptedKey	9CF4E2DE12A260E45D082D5DCCE4D050

7.3 Data Encryption Example

The card data, PAN 4978678252755678 and expiry date 0411, must be protected by encryption. The data to encrypt, *SensitiveData*, is presented below:

SensitiveCardData

PAN	4978678252755678
ExpiryDate	0411

Before encryption, *SensitiveData* is coded in XML:

```
<SensitiveCardData PAN="4978678252755678" ExpiryDate="0411"/>
```

The binary value of the XML encoded *SensitiveData* is:

0000	3C	53	65	6E	73	69	74	69	76	65	43	61	72	64	44	61	<SensitiveCardDa
0010	74	61	20	50	41	4E	3D	22	34	39	37	38	36	37	38	32	ta PAN="49786782
0020	35	32	37	35	35	36	37	38	22	20	45	78	70	69	72	61	52755678" Expira
0030	74	69	6F	6E	44	61	74	65	3D	22	30	34	31	31	22	2F	tionDate="0411"/
0040	3E															>	

The hexadecimal byte 80 is added, following by 6 null bytes to reach a length, multiple of 8 (64 bytes):

0000	3C	53	65	6E	73	69	74	69	76	65	43	61	72	64	44	61	<SensitiveCardDa
0010	74	61	20	50	41	4E	3D	22	34	39	37	38	36	37	38	32	ta PAN="49786782
0020	35	32	37	35	35	36	37	38	22	20	45	78	70	69	72	61	52755678" Expira
0030	74	69	6F	6E	44	61	74	65	3D	22	30	34	31	31	22	2F	tionDate="0411"/
0040	3E	80	00	00	00	00	00	00								>.....	

Taking the example of the previous section, the session key has the value: "E64AEADA 2A6E34B6 DF790DE3 0E46E9BF"

The Triple DES CBC encryption uses the Initialisation Vector: "A27BB46D 1C306E09"

The result of the encryption is then:

0000	FE	CC	2C	96	0F	55	25	D8	2A	82	B9	E6	41	93	45	45	.,.,U%.*....A.EE
0010	A6	8A	55	6B	01	E0	59	49	C8	88	F7	52	DC	42	96	AE	..Uk..YI....R.B..
0020	A3	77	04	4E	0E	92	6F	40	84	2A	84	EE	F5	D6	DC	27	.w.N..o@.*.....'
0030	1C	28	D7	F7	ED	DB	40	CD	2D	CA	F3	8D	02	38	AA	.(....@.-....>.8.	
0040	58	11	C8	79	EF	92	BF	32								X..y...2	

The *ProtectedCardData* data structure is a CMS containing the encrypted result of *SensitiveData* with all the related information is presented in the table below.

ProtectedCardData

ContentType	id-envelopedData
EnvelopedData	
Version	v0
Recipient	
KEK	
Version	v4
KEKIdentifier	
KeyIdentifier	SpecV1TestDATKey
KeyVersion	2010060715
KeyEncryptionAlgorithm	
Algorithm	des-ed3-cbc
EncryptedKey	9CF4E2DE12A260E45D082D5DCCE4D050
EncryptedContent	
ContentType	id-data
ContentEncryptionAlgorithm	
Algorithm	des-ed3-cbc
Parameter	
InitialisationVector	A27BB46D1C306E09
EncryptedData	FECC2C960F5525D82A82B9E641934545 A68A556B01E05949C888F752DC4296AE A377044E0E926F40842A84EEF5D6DC27 1C28D7F7EDDB40CD2DCAF38D3E0238AA 5811C879EF92BF32

7.4 MAC Protection Example

This example presents a *PaymentRequest* and a *PaymentResponse* message with a MAC.

The message request before addition of the *SecurityTrailer* containing the MAC is presented in the table below:

MessageHeader	
MessageClass	Service
MessageCategory	Payment
MessageType	Request
ServiceID	642
SaleID	SaleTermA
POIID	POITerm1
PaymentRequest	
SaleData	
SaleTransactionID	
TransactionID	580
TimeStamp	2010-06-10T22:53:12.6+01:00
PaymentTransaction	
AmountsReq	
Currency	EUR
RequestedAmount	31.00
TransactionConditions	
LoyaltyHandling	Forbidden

The concatenation of the header *MessageHeader* and the body *PaymentRequest*, using an XML/Schema data coding is presented below:

```
<MessageHeader MessageClass="Service" MessageCategory="Payment"
  MessageType="Request" ServiceID="642" SaleID="SaleTermA" POIID="POITerm1"/>
<PaymentRequest>
  <SaleData>
    <SaleTransactionID TransactionID="580" TimeStamp="2010-06-
      10T22:53:12.6+01:00"/>
  </SaleData>
  <PaymentTransaction>
    <AmountsReq Currency="EUR" RequestedAmount="31.00"/>
    <TransactionConditions LoyaltyHandling="Forbidden"/>
  </PaymentTransaction>
</PaymentRequest>
```

The dump of the canonical form of the XML data is:

```

000000 3C 4D 65 73 73 61 67 65 48 65 61 64 65 72 20 4D |<MessageHeader M|
000010 65 73 73 61 67 65 43 6C 61 73 73 3D 22 53 65 72 |essageClass="Ser|
000020 76 69 63 65 22 20 4D 65 73 73 61 67 65 43 61 74 |vice" MessageCat|
000030 65 67 6F 72 79 3D 22 50 61 79 6D 65 6E 74 22 20 |egory="Payment" |
000040 4D 65 73 73 61 67 65 54 79 70 65 3D 22 52 65 71 |MessageType="Req|
000050 75 65 73 74 22 20 53 65 72 76 69 63 65 49 44 3D |uest" ServiceID=|
000060 22 36 34 32 22 20 53 61 6C 65 49 44 3D 22 53 61 |"642" SaleID="Sa|
000070 6C 65 54 65 72 6D 41 22 20 50 4F 49 49 44 3D 22 |leTermA" POIID="|
000080 50 4F 49 54 65 72 6D 31 22 2F 3E 3C 50 61 79 6D |POITerm1"/><Paym|
000090 65 6E 74 52 65 71 75 65 73 74 3E 3C 53 61 6C 65 |entRequest><Sale|
0000A0 44 61 74 61 3E 3C 53 61 6C 65 54 72 61 6E 73 61 |Data><SaleTransa|
0000B0 63 74 69 6F 6E 49 44 20 54 72 61 6E 73 61 63 74 |ctionID Transact|
0000C0 69 6F 6E 49 44 3D 22 35 38 30 22 20 54 69 6D 65 |onID="580" Time|
0000D0 53 74 61 6D 70 3D 22 32 30 31 30 2D 30 36 2D 31 |Stamp="2010-06-1|
0000E0 30 54 32 32 3A 35 33 3A 31 32 2E 36 2B 30 31 3A |0T22:53:12.6+01:|
0000F0 30 30 22 2F 3E 3C 2F 53 61 6C 65 44 61 74 61 3E |00"/></SaleData>|
000100 3C 50 61 79 6D 65 6E 74 54 72 61 6E 73 61 63 74 |<PaymentTransact|
000110 69 6F 6E 3E 3C 41 6D 6F 75 6E 74 73 52 65 71 20 |ion><AmountsReq|
000120 43 75 72 72 65 6E 63 79 3D 22 45 55 52 22 20 52 |Currency="EUR" R|
000130 65 71 75 65 73 74 65 64 41 6D 6F 75 6E 74 3D 22 |equestedAmount="|
000140 33 31 2E 30 30 22 2F 3E 3C 54 72 61 6E 73 61 63 |31.00"/><Transac|
000150 74 69 6F 6E 43 6F 6E 64 69 74 69 6F 6E 73 20 4C |tionConditions L|
000160 6F 79 61 6C 74 79 48 61 6E 64 6C 69 6E 67 3D 22 |oyaltyHandling="|
000170 46 6F 72 62 69 64 64 65 6E 22 2F 3E 3C 2F 50 61 |Forbidden"/></Pa|
000180 79 6D 65 6E 74 54 72 61 6E 73 61 63 74 69 6F 6E |ymentTransaction|
000190 3E 3C 2F 50 61 79 6D 65 6E 74 52 65 71 75 65 73 |></PaymentReques|
0001A0 74 3E |t>|

```

The SHA-256 digest of the canonical form is:

0000 13 4D F4 38 07 E0 FA A8 23 C1 0C 21 4C 97 F3 33 .M.8....#.!L..3
0010 E5 69 44 FE FC 37 5D 72 9D 8A 87 C6 2C 33 95 E0 .i.D..7]r.....,3..

Padding of the digest provides the result:

0000 13 4D F4 38 07 E0 FA A8 23 C1 0C 21 4C 97 F3 33 .M.8....#.!L..3
0010 E5 69 44 FE FC 37 5D 72 9D 8A 87 C6 2C 33 95 E0 .i.D..7]r.....,3..
0020 80 00 00 00 00 00 00 00

Taking the key value of a previous example, the session key has the value: "E64AEADA 2A6E34B6 DF790DE3 0E46E9BF"

The Triple DES retail CBC uses the null Initialisation Vector: "00000000 00000000"

The intermediate result of the retail CBC is:

0000 CA A5 6A CF AB 04 FE 09 CE F0 6D D1 E2 C1 37 F8 ...j.....m....7.
0010 C8 E0 AC FA A6 5E 90 B3 36 C1 AB 70 C5 B8 DA 3B ^..6..p...;
0020 F4 41 1A E4 4D 2A 71 7B .A..M*q{

The MAC is then: F4411AE44D2A717B

The *SecurityTrailer* with the MAC to add to the message is presented in the table below.

SecurityTrailer

ContentType	id-ct-authData
AuthenticatedData	
Version	v0
Recipient	
KEK	
Version	v4
KEKIdentifier	
KeyIdentifier	SpecV1TestMACKey
KeyVersion	2010060715
KeyEncryptionAlgorithm	
Algorithm	des-ed3-cbc
EncryptedKey	9CF4E2DE12A260E45D082D5DCCE4D050
MACAlgorithm	
Algorithm	id-retail-cbc-mac-sha-256
EncapsulatedContent	
ContentType	id-data
MAC	F4411AE44D2A717B

The message response before addition of the *SecurityTrailer* containing the MAC is presented in the table below:

MessageHeader	
MessageClass	Service
MessageCategory	Payment
MessageType	Response
ServiceID	642
SaleID	SaleTermA
POIID	POITerm1
PaymentResponse	
Response	
SaleData	
SaleTransactionID	
TransactionID	580
TimeStamp	2010-06-10T22:53:12.6+01:00
Result	Success
POIData	
POITransactionID	
TransactionID	482
TimeStamp	2010-06-10T22:53:12.6+01:00
POIReconciliationID	200903101
PaymentResult	
PaymentInstrumentData	
PaymentInstrumentType	Card
CardData	
PaymentBrand	VISA
EntryMode	MagStripe
ProtectedCardData	
ContentType	id-envelopeData
EnvelopedData	
Version	v0
Recipient	
KEK	
Version	v4
KEKIdentifier	
KeyIdentifier	SpecV1TestDATKey
KeyVersion	2010060715
KeyEncryptionAlgorithm	
Algorithm	des-ed3-cbc
EncryptedKey	9CF4E2DE12A260E45D082D5DCCE4D050
EncryptedContent	
ContentType	id-data
ContentEncryptionAlgorithm	
Algorithm	des-ed3-cbc
Parameter	
InitialisationVector	A27BB46D1C306E09
EncryptedData	FECC2C960F5525D82A82B9E641934545 A68A556B01E05949C888F752DC4296AE A377044E0E926F40842A84EEF5D6DC27 1C28D7F7EDDB40CD2DCAF38D3E0238AA 5811C879EF92BF32
AmountsResp	
AuthorizedAmount	31.00
PaymentAcquirerData	
AcquirerID	497867
MerchantID	mer77-130209
AcquirerPOIID	456

ApprovalCode

9473

The complete *PaymentRequest* application message, including the *SecurityTrailer* containing the MAC, and using an XML/Schema encoding is presented in the section on messages examples in XML (section 7.5.5.2 *Protected Card Data and MAC* page 780).

The concatenation of the header *MessageHeader* and the body *PaymentResponse*, using an XML/Schema data coding is presented below:

```
<MessageHeader MessageClass="Service" MessageCategory="Payment"
MessageID="642" SaleID="SaleTermA" POIID="POITerm1"/>
<PaymentResponse>
    <Response Result="Success"/>
    <SaleData>
        <SaleTransactionID TransactionID="580" TimeStamp="2010-06-
            10T22:53:12.6+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
        <POITransactionID TransactionID="482" TimeStamp="2010-06-
            10T22:53:12.6+01:00"/>
    </POIData>
    <PaymentResult>
        <PaymentInstrumentData PaymentInstrumentType="Card">
            <CardData PaymentBrand="VISA" EntryMode="MagStripe">
                <ProtectedCardData ContentType="id-envelopedData">
                    <EnvelopedData Version="v0">
                        <KEK Version="v4" EncryptedKey="nPTi3hKiYORdCC1dzOTQUA==">
                            <KEKIdentifier KeyIdentifier="SpecV1TestDATKey"
                                KeyVersion="2010060715"/>
                            <KeyEncryptionAlgorithm Algorithm="des-ed3-cbc"/>
                        </KEK>
                        <EncryptedContent ContentType="id-data">
                            <ContentEncryptionAlgorithm Algorithm="des-ed3-cbc">
                                <Parameter InitialisationVector="onu0bRwrbgk="/>
                            </ContentEncryptionAlgorithm>
                            <EncryptedData>/sws1g9VJdgqgrnmQZNFRaaKVWsB4FlJyIj3UtxC1q6jdwRODp
                                JvQIQqh0711twnHCjX9+3bQM0tyvONPgI4qlgRyHnvkr8y</EncryptedData>
                        </EncryptedContent>
                    </EnvelopedData>
                </ProtectedCardData>
            </CardData>
        </PaymentInstrumentData>
        <AmountsResp AuthorizedAmount="31.00"/>
        <PaymentAcquirerData AcquirerID="497867" MerchantID="mer77-130209"
            AcquirerPOIID="456">
            <ApprovalCode>9473</ApprovalCode>
        </PaymentAcquirerData>
    </PaymentResult>
</PaymentResponse>
```

The dump of the canonical form of the XML data is:

0000	3C	4D	65	73	73	61	67	65	48	65	61	64	65	72	20	4D	<MessageHeader M
0010	65	73	73	61	67	65	43	6C	61	73	73	3D	22	53	65	72	essageClass="Ser
0020	76	69	63	65	22	20	4D	65	73	73	61	67	65	43	61	74	vice" MessageCat
0030	65	67	6F	72	79	3D	22	50	61	79	6D	65	6E	74	22	20	egory="Payment"
0040	4D	65	73	73	61	67	65	54	79	70	65	3D	22	52	65	73	MessageType="Res
0050	70	6F	6E	73	65	22	20	53	65	72	76	69	63	65	49	44	ponse" ServiceID
0060	3D	22	36	34	32	22	20	53	61	6C	65	49	44	3D	22	53	="642" SaleID="S
0070	61	6C	65	54	65	72	6D	41	22	20	50	4F	49	49	44	3D	aleTermA" POIID=
0080	22	50	4F	49	54	65	72	6D	31	22	2F	3E	3C	50	61	79	"POITerm1"/><Pay
0090	6D	65	6E	74	52	65	73	70	6F	6E	73	65	3E	3C	52	65	mentResponse><Re
00AA0	73	70	6F	6E	73	65	20	52	65	73	75	6C	74	3D	22	53	sponse Result="S
00B00	75	63	63	65	73	73	22	2F	3E	3C	53	61	6C	65	44	61	uccess"/><SaleDa
00C00	74	61	3E	3C	53	61	6C	65	54	72	61	6E	73	61	63	74	ta><SaleTransact
00DD0	69	6F	6E	49	44	20	54	72	61	6E	73	61	63	74	69	6F	ionID Transaction
00EE0	6E	49	44	3D	22	35	38	30	22	20	54	69	6D	65	53	74	nID="580" TimeSt
00FF0	61	6D	70	3D	22	32	30	31	30	2D	30	36	2D	31	30	54	amp="2010-06-10T
0100	32	32	3A	35	33	3A	31	32	2E	36	2B	30	31	3A	30	30	22:53:12.6+01:00
0110	22	2F	3E	3C	2F	53	61	6C	65	44	61	74	61	3E	3C	50	"/></SaleData><P
0120	4F	49	44	61	74	61	20	50	4F	49	52	65	63	6F	6E	63	OIData POIReconc
0130	69	6C	69	61	74	69	6F	6E	49	44	3D	22	32	30	30	39	iliationID="2009
0140	30	33	31	30	31	22	3E	3C	50	4F	49	54	72	61	6E	73	03101"><POITrans
0150	61	63	74	69	6F	6E	49	44	20	54	72	61	6E	73	61	63	actionID Transac
0160	74	69	6F	6E	49	44	3D	22	34	38	32	22	20	54	69	6D	tionID="482" Tim
0170	65	53	74	61	6D	70	3D	22	32	30	31	30	2D	30	36	2D	eStamp="2010-06-
0180	31	30	54	32	32	3A	35	33	3A	31	32	2E	36	2B	30	31	10T22:53:12.6+01
0190	3A	30	30	22	2F	3E	3C	2F	50	4F	49	44	61	74	61	3E	:00"/></POIData>
01AA0	3C	50	61	79	6D	65	6E	74	52	65	73	75	6C	74	3E	3C	<PaymentResult><
01BB0	50	61	79	6D	65	6E	74	49	6E	73	74	72	75	6D	65	6E	PaymentInstrumen
01CC0	74	44	61	74	61	20	50	61	79	6D	65	6E	74	49	6E	73	tData PaymentIns
01DD0	74	72	75	6D	65	6E	74	54	79	70	65	3D	22	43	61	72	InstrumentType="Car
01EE0	64	22	3E	3C	43	61	72	64	44	61	74	61	20	50	61	79	d"><CardData Pay
01FF0	6D	65	6E	74	42	72	61	6E	64	3D	22	56	49	53	41	22	mentBrand="VISA"
0200	20	45	6E	74	72	79	4D	6F	64	65	3D	22	4D	61	67	53	EntryMode="MagS
0210	74	72	69	70	65	22	3E	3C	50	72	6F	74	65	63	74	65	tripe"><Protecte
0220	64	43	61	72	64	44	61	74	61	20	43	6F	6E	74	65	6E	dCardData Conten
0230	74	54	79	70	65	3D	22	69	64	2D	65	6E	76	65	6C	6F	tType="id-envelo
0240	70	65	64	44	61	74	61	22	3E	3C	45	6E	76	65	6C	6F	pedData"><Envelo
0250	70	65	64	44	61	74	61	20	56	65	72	73	69	6F	6E	3D	pedData Version=
0260	22	76	30	22	3E	3C	4B	45	4B	20	56	65	72	73	69	6F	"v0"><KEK Versio
0270	6E	3D	22	76	34	22	20	45	6E	63	72	79	70	74	65	64	n="v4" Encrypted
0280	4B	65	79	3D	22	6E	50	54	69	33	68	4B	69	59	4F	52	Key="nPti3hKiYOR
0290	64	43	43	31	64	7A	4F	54	51	55	41	3D	3D	22	3E	3C	dCC1dzOTQUA=="><
02AA0	4B	45	4B	49	64	65	6E	74	69	66	69	65	72	20	4B	65	KEKIdentifier Ke
02BB0	79	49	64	65	6E	74	69	66	69	65	72	3D	22	53	70	65	yIdentifier="Spe
02CC0	63	56	31	54	65	73	74	44	41	54	4B	65	79	22	20	4B	cV1TestDATKey" Ki
02DD0	65	79	56	65	72	73	69	6F	6E	3D	22	32	30	31	30	30	eyVersion="20100
02EE0	36	30	37	31	35	22	2F	3E	3C	4B	65	79	45	6E	63	72	60715"/><KeyEncr
02FF0	79	70	74	69	6F	6E	41	6C	67	6F	72	69	74	68	6D	20	ptionAlgorithm
0300	41	6C	67	6F	72	69	74	68	6D	3D	22	64	65	73	2D	65	Algorithm="des-e
0310	64	65	33	2D	63	62	63	22	2F	3E	3C	2F	4B	45	4B	3E	de3-cbc"/></KEK>
0320	3C	45	6E	63	72	79	70	74	65	64	43	6F	6E	74	65	6E	<EncryptedConten
0330	74	20	43	6F	6E	74	65	6E	74	54	79	70	65	3D	22	69	t ContentType="i
0340	64	2D	64	61	74	61	22	3E	3C	43	6F	6E	74	65	6E	74	d-data"><Content
0350	45	6E	63	72	79	70	74	69	6F	6E	41	6C	67	6F	72	69	EncryptionAlgori
0360	74	68	6D	20	41	6C	67	6F	72	69	74	68	6D	3D	22	64	thm Algorithm="d
0370	65	73	2D	65	64	65	33	2D	63	62	63	22	3E	3C	50	61	es-edede3-cbc"><Pa
0380	72	61	6D	65	74	65	72	20	49	6E	69	74	69	61	6C	69	rameter Initiali
0390	73	61	74	69	6F	6E	56	65	63	74	6F	72	3D	22	6F	6E	sationVector="on
03AA0	75	30	62	52	77	77	62	67	6B	3D	22	2F	3E	3C	2F	43	u0bRwwbgk="/></C
03BB0	6F	74	65	6E	74	45	6E	63	72	79	70	74	69	6F	6E	74	ontentEncryption
03CC0	41	6C	67	6F	72	69	74	68	6D	3E	3C	45	6E	63	72	79	Algorithm><Encry
03DD0	70	74	65	64	44	61	74	61	3E	2F	73	77	73	6C	67	39	ptedData>/swslg9
03EE0	56	4A	64	67	71	67	72	6E	6D	51	5A	4E	46	52	61	61	VJdgqgrnmQZNFRaa
03FF0	4B	56	57	73	42	34	46	6C	4A	79	49	6A	33	55	74	78	KVWsB4FlJyIj3Utx

0400	43	6C	71	36	6A	64	77	52	4F	44	70	4A	76	51	49	51	Clq6jdwRODpJvQIQ
0410	71	68	4F	37	31	31	74	77	6E	48	43	6A	58	39	2B	33	qhO711twnHCjX9+3
0420	62	51	4D	30	74	79	76	4F	4E	50	67	49	34	71	6C	67	bQM0tyvONPgI4qlg
0430	52	79	48	6E	76	6B	72	38	79	3C	2F	45	6E	63	72	79	RyHnvkr8y</Encry
0440	70	74	65	64	44	61	74	61	3E	3C	2F	45	6E	63	72	79	ptedData></Encry
0450	70	74	65	64	43	6F	6E	74	65	6E	74	3E	3C	2F	45	6E	ptedContent></En
0460	76	65	6C	6F	70	65	64	44	61	74	61	3E	3C	2F	50	72	velopedData></Pr
0470	6F	74	65	63	74	65	64	43	61	72	64	44	61	74	61	3E	otectedCardData>
0480	3C	2F	43	61	72	64	44	61	74	61	3E	3C	2F	50	61	79	</CardData></Pay
0490	6D	65	6E	74	49	6E	73	74	72	75	6D	65	6E	74	44	61	mentInstrumentDa
04A0	74	61	3E	3C	41	6D	6F	75	6E	74	73	52	65	73	70	20	ta><AmountsResp
04B0	41	75	74	68	6F	72	69	7A	65	64	41	6D	6F	75	6E	74	AuthorizedAmount
04C0	3D	22	33	31	2E	30	30	22	2F	3E	3C	50	61	79	6D	65	="31.00"/><Payme
04D0	6E	74	41	63	71	75	69	72	65	72	44	61	74	61	20	41	ntAcquirerData A
04E0	63	71	75	69	72	65	72	49	44	3D	22	34	39	37	38	36	cquirerID="49786
04F0	37	22	20	4D	65	72	63	68	61	6E	74	49	44	3D	22	6D	7" MerchantID="m
0500	65	72	37	37	2D	31	33	30	32	30	39	22	20	41	63	71	er77-130209" Acq
0510	75	69	72	65	72	50	4F	49	49	44	3D	22	34	35	36	22	uirerPOIID="456"
0520	3E	3C	41	70	70	72	6F	76	61	6C	43	6F	64	65	3E	39	><ApprovalCode>9
0530	34	37	33	3C	2F	41	70	70	72	6F	76	61	6C	43	6F	64	473</ApprovalCod
0540	65	3E	3C	2F	50	61	79	6D	65	6E	74	41	63	71	75	69	e></PaymentAcqui
0550	72	65	72	44	61	74	61	3E	3C	2F	50	61	79	6D	65	6E	rerData></Paymen
0560	74	52	65	73	75	6C	74	3E	3C	2F	50	61	79	6D	65	6E	tResult></Paymen
0570	74	52	65	73	70	6F	6E	73	65	3E						tResponse>	

The SHA-256 digest of the canonical form is:

0000 DF B7 29 E3 FE F1 35 D1 42 01 27 22 9E 75 B5 B1 ...)...5.B.'".u..
0010 8A 75 45 15 D4 2E 3B FE A9 3B C8 5A A2 E3 55 64 .uE...;...;.Z..Ud

Padding of the digest provides the result:

0000 DF B7 29 E3 FE F1 35 D1 42 01 27 22 9E 75 B5 B1 ...)...5.B.'".u..
0010 8A 75 45 15 D4 2E 3B FE A9 3B C8 5A A2 E3 55 64 .uE...;...;.Z..Ud
0020 80 00 00 00 00 00 00 00

Taking the key value of a previous example, the session key has the value: "E64AEADA 2A6E34B6 DF790DE3 0E46E9BF"

The Triple DES retail CBC uses the null Initialisation Vector: "00000000 00000000"

The intermediate result of the retail CBC is:

0000 AA 35 A9 F2 6C CA 4A 36 54 A7 DC 72 A7 21 52 D4 .5..1.J6T..r.!R.
0010 97 5A 74 35 84 AF 8A 35 79 26 19 C3 0E 5F 1B F5 .Zt5...5y&...._
0020 C9 98 B3 51 E3 9F E2 D0 ...Q....

The MAC is then: C998B351E39FE2D0

The *SecurityTrailer* with the MAC to add to the message is presented in the table below.

SecurityTrailer	
ContentType	id-ct-authData
AuthenticatedData	
Version	v0
Recipient	
KEK	
Version	v4
KEKIdentifier	
KeyIdentifier	SpecV1TestMACKey
KeyVersion	2010060715
KeyEncryptionAlgorithm	
Algorithm	des-edc3-cbc
EncryptedKey	9CF4E2DE12A260E45D082D5DCCE4D050
MACAlgorithm	
Algorithm	id-retail-cbc-mac-sha-256
EncapsulatedContent	
ContentType	id-data
MAC	C998B351E39FE2D0

The complete *PaymentResponse* application message, including the *SecurityTrailer* containing the MAC, and using an XML/Schema encoding is presented in the section on messages examples in XML (section 7.5.5.2 *Protected Card Data and MAC* page 780).

7.5 Message Examples in XML

This section presents all the examples of messages presented in the chapter 4, *Application Protocol Specifications*. These messages are encoded in XML/Schema²⁹.

These messages are reformatted for the presentation. Files containing binary messages in canonical form are provided with the XML/Schema definition files. The names of the files are `Ex-XX-MessageName.xml`, where `XX` is the number of the example, and `MessageName` is the name of the message (for instance, the first example coming from the section 4.2.3, *Login*, has the name `Ex-01-LoginRequest.xml`).

7.5.1 LoginRequest - LoginResponse

This section presents message examples 1 and 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader ProtocolVersion="3.0" MessageClass="Service"
    MessageCategory="Login" MessageType="Request" ServiceID="498"
    SaleID="SaleTermA" POIID="POITerm1"/>
  <LoginRequest OperatorLanguage="sp" OperatorID="Cashier16" ShiftNumber="2"
    POISerialNumber="78910AA46010005">
    <DateTime>2015-03-08T09:13:51.0+01:00</DateTime>
    <SaleSoftware ManufacturerID="PointOfSaleCo" ApplicationName="SaleSys"
      SoftwareVersion="01.98.01" CertificationCode="ECTS2PS001"/>
    <SaleTerminalData TerminalEnvironment="Attended">
      <SaleCapabilities>PrinterReceipt CashierStatus CashierError CashierDisplay
        CashierInput</SaleCapabilities>
      <SaleProfile GenericProfile="Extended">
        <ServiceProfiles>Loyalty PIN CardReader</ServiceProfiles>
      </SaleProfile>
    </SaleTerminalData>
  </LoginRequest>
</SaleToPOIRequest>
```

²⁹ All these messages are encoded in an optimised form of XML (i.e. without spaces and line feed, an .xml file per example, along with the specifications). In case of difference, the message files take the precedence on this section, as they have been automatically generated.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader ProtocolVersion="3.0" MessageClass="Service"
    MessageCategory="Login" MessageType="Response" ServiceID="498"
    SaleID="SaleTermA" POIID="POITerm1"/>
  <LoginResponse>
    <Response Result="Success"/>
    <POISystemData>
      <DateTime>2015-03-08T09:13:49.8+01:00</DateTime>
      <POISoftware ManufacturerID="ElecFundsCo" ApplicationName="POISys"
        SoftwareVersion="5.0.001" CertificationCode="ECTS2PP001"/>
      <POITerminalData TerminalEnvironment="Attended"
        POISerialNumber="78910AA46010005">
        <POICapabilities>CustomerDisplay CustomerError MagStripe
          ICC</POICapabilities>
        <POIPattern GenericProfile="Extended">
          <ServiceProfiles>Loyalty PIN CardReader</ServiceProfiles>
        </POIPattern>
      </POITerminalData>
      <POIStatus GlobalStatus="OK" SecurityOKFlag="true" PEDOKFlag="true"
        CardReaderOKFlag="true" PrinterStatus="OK" CommunicationOKFlag="true">
        <CashHandlingDevice CashHandlingOKFlag="true" Currency="EUR">
          <CoinsOrBills UnitValue="2" Number="8"\>
          <CoinsOrBills UnitValue="1" Number="43"\>
          <CoinsOrBills UnitValue="0.50" Number="19"\>
          <CoinsOrBills UnitValue="0.20" Number="38"\>
          <CoinsOrBills UnitValue="0.10" Number="27"\>
          <CoinsOrBills UnitValue="0.05" Number="0"\>
        </CashHandlingDevice>
      </POIStatus>
    </POISystemData>
  </LoginResponse>
</SaleToPOIResponse>
```

7.5.2 LogoutRequest LogoutResponse

This section presents message examples 3 and 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Logout"
    MessageType="Request" ServiceID="613" SaleID="SaleTermA" POIID="POITerm1"/>
  <LogoutRequest MaintenanceAllowed="true"/>
</SaleToPOIRequest>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Logout"
    MessageType="Response" ServiceID="613" SaleID="SaleTermA" POIID="POITerm1"/>
  <LogoutResponse>
    <Response Result="Success"/>
  </LogoutResponse>
</SaleToPOIResponse>
```

7.5.3 EnableServiceRequest EnableServiceResponse

This section presents message examples 5 and 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="EnableService"
    MessageType="Request" ServiceID="640" SaleID="SaleTermA" POIID="POITerm1"/>
  <EnableServiceRequest TransactionAction="StartTransaction">
    <ServicesEnabled>Payment</ServicesEnabled>
    <DisplayOutput Device="CustomerDisplay" InfoQualify="Display">
      <OutputContent OutputFormat="Text">
        <OutputText>Please, enter your card</OutputText>
      </OutputContent>
    </DisplayOutput>
  </EnableServiceRequest>
</SaleToPOIRequest>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="EnableService"
    MessageType="Response" ServiceID="640" SaleID="SaleTermA" POIID="POITerm1"/>
  <EnableServiceResponse>
    <Response Result="Success"/>
  </EnableServiceResponse>
</SaleToPOIResponse>
```

7.5.4 AdminRequest AdminResponse

This section presents message examples 7 and 8.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Admin"
    MessageType="Request" ServiceID="613" SaleID="SaleTermA" POIID="POITerm1"/>
  <AdminRequest/>
</SaleToPOIRequest>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Admin"
    MessageType="Response" ServiceID="613" SaleID="SaleTermA" POIID="POITerm1"/>
  <AdminResponse>
    <Response Result="Success"/>
  </AdminResponse>
</SaleToPOIResponse>
```

7.5.5 PaymentRequest PaymentResponse

7.5.5.1 Simple Payment

This section presents message examples 9 and 10.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="642" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="579"TimeStamp="2009-03-
        10T23:08:42.4+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="104.11"/>
      <TransactionConditions LoyaltyHandling="Forbidden"/>
    </PaymentTransaction>
    <PaymentData PaymentType="Normal"/>
  </PaymentRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Response" ServiceID="642" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="579"TimeStamp="2009-03-
        10T23:08:42.4+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="481"TimeStamp="2009-03-
        10T23:08:42.4+01:00"/>
    </POIData>
    <PaymentResult PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData PaymentBrand="CardPlus" EntryMode="MagStripe">
          <SensitiveCardData PAN="0011014570541535" ExpiryDate="0411"/>
        </CardData>
      </PaymentInstrumentData>
      <AmountsResp AuthorizedAmount="104.11"/>
      <PaymentAcquirerData AcquirerID="400012" MerchantID="mer77-130209">
        <AcquirerPOIID="963276433">
          <ApprovalCode>8347</ApprovalCode>
        </PaymentAcquirerData>
      </PaymentResult>
    </PaymentResponse>
  </SaleToPOIResponse>
```

7.5.5.2 Protected Card Data and MAC

This section presents message examples 11 and 12.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="642" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="580"TimeStamp="2010-06-
        10T22:53:12.6+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="31.00"/>
      <TransactionConditions LoyaltyHandling="Forbidden"/>
    </PaymentTransaction>
  </PaymentRequest>
  <SecurityTrailer ContentType="id-ct-authData">
    <AuthenticatedData Version="v0" MAC="9EEa5E0qcXs">
      <KEK Version="v4" EncryptedKey="nPTi3hKiYORdCC1dzOTQUA==">
        <KEKIdentifier KeyIdentifier="SpecV1TestMACKey" KeyVersion="2010060715"/>
        <KeyEncryptionAlgorithm Algorithm="des-ed3-cbc"/>
      </KEK>
      <MACAlgorithm Algorithm="id-retail-cbc-mac-sha-256"/>
      <EncapsulatedContent ContentType="id-data"/>
    </AuthenticatedData>
  </SecurityTrailer>
</SaleToPOIRequest>
```

```

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Response" ServiceID="642" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="580"TimeStamp="2010-06-
        10T22:53:12.6+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="482"TimeStamp="2010-06-
        10T22:53:12.6+01:00"/>
    </POIData>
    <PaymentResult>
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData PaymentBrand="VISA" EntryMode="MagStripe">
          <ProtectedCardData ContentType="id-envelopedData">
            <EnvelopedData Version="v0">
              <KEK Version="v4" EncryptedKey="nPTi3hKiYORdCC1dzOTQUA==">
                <KEKIdentifier KeyIdentifier="SpecV1TestDATKey"
                  KeyVersion="2010060715"/>
                <KeyEncryptionAlgorithm Algorithm="des-ed3-cbc"/>
              </KEK>
              <EncryptedContent ContentType="id-data">
                <ContentEncryptionAlgorithm Algorithm="des-ed3-cbc">
                  <Parameter InitialisationVector="onu0bRwwbgk="/>
                </ContentEncryptionAlgorithm>
                <EncryptedData>
                  /swslg9VJdgqgrnmQZNFRaaKVWsB4FlJyIj3UtxClq6jdwRODpJvQIQqh071
                  1twnHCjX9+3bQM0tyvONPgI4qlgRyHnvkr8y
                </EncryptedData>
              </EncryptedContent>
            </EnvelopedData>
          </ProtectedCardData>
        </CardData>
      </PaymentInstrumentData>
      <AmountsResp AuthorizedAmount="31.00"/>
      <PaymentAcquirerData AcquirerID="497867" MerchantID="mer77-130209"
        AcquirerPOIID="456">
        <ApprovalCode>9473</ApprovalCode>
      </PaymentAcquirerData>
    </PaymentResult>
  </PaymentResponse>
  <SecurityTrailer ContentType="id-ct-authData">
    <AuthenticatedData Version="v0" MAC="yZizUeOf4tA==">
      <KEK Version="v4" EncryptedKey="nPTi3hKiYORdCC1dzOTQUA==">
        <KEKIdentifier KeyIdentifier="SpecV1TestMACKey" KeyVersion="2010060715"/>
        <KeyEncryptionAlgorithm Algorithm="des-ed3-cbc"/>
      </KEK>
      <MACAlgorithm Algorithm="id-retail-cbc-mac-sha-256"/>
      <EncapsulatedContent ContentType="id-data"/>
    </AuthenticatedData>
  </SecurityTrailer>
</SaleToPOIResponse>

```

7.5.5.3 Payment with Products

This section presents message examples 13 and 14.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="643" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="581"TimeStamp="2015-04-
        06T10:42:34.1+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="38.52"/>
      <TransactionConditions LoyaltyHandling="Forbidden"/>
      <SaleItem ItemID="1" ProductCode="673" EanUpc="84116369" ItemAmount="2.30">
        <Quantity>2</Quantity>
        <UnitPrice>1.15</UnitPrice>
        <ProductLabel>Mineral Water 1,5L</ProductLabel>
      </SaleItem>
      <SaleItem ItemID="2" ProductCode="101" ItemAmount="33.22">
        <UnitOfMeasure>Litre</UnitOfMeasure>
        <Quantity>38.23</Quantity>
        <UnitPrice>0.869</UnitPrice>
        <ProductLabel>Diesel Fuel</ProductLabel>
      </SaleItem>
    </PaymentTransaction>
    <PaymentData PaymentType="Normal"/>
  </PaymentRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Response" ServiceID="643" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="581"TimeStamp="2015-04-
        06T10:42:34.1+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="479"TimeStamp="2015-04-
        06T10:43:02.9+01:00"/>
    </POIData>
    <PaymentResult PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData PaymentBrand="FleetUniverse" EntryMode="MagStripe">
          <SensitiveCardData PAN="6900014570541535" ExpiryDate="0411"/>
        </CardData>
      </PaymentInstrumentData>
      <AmountsResp AuthorizedAmount="38.52"/>
      <PaymentAcquirerData AcquirerID="690001" MerchantID="mer77-130209">
        <AcquirerPOIID="963276433">
          <ApprovalCode>6541</ApprovalCode>
        </PaymentAcquirerData>
      </PaymentResult>
    </PaymentResponse>
</SaleToPOIResponse>
```

7.5.5.4 Payment with a Local Card

This section presents message examples 15 and 16.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="644" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="581"TimeStamp="2009-03-
        11T10:37:21.9+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="38.01"/>
      <TransactionConditions LoyaltyHandling="Forbidden"/>
    </PaymentTransaction>
    <PaymentData PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData EntryMode="Scanned">
          <SensitiveCardData PAN="7011014570541535"/>
        </CardData>
      </PaymentInstrumentData>
    </PaymentData>
  </PaymentRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Response" ServiceID="644" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentResponse>
    <Response Result="Partial"/>
    <SaleData>
      <SaleTransactionID TransactionID="581"TimeStamp="2009-07-
        31T10:42:34.1+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="481"TimeStamp="2009-03-
        11T10:42:52.4+01:00"/>
    </POIData>
    <PaymentResult PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData PaymentBrand="MerRel" EntryMode="MagStripe">
          <SensitiveCardData PAN="7011014570541535"/>
        </CardData>
      </PaymentInstrumentData>
      <AmountsResp AuthorizedAmount="31.12" TotalFeesAmount="0.19"/>
      <PaymentAcquirerData MerchantID="mer77-130209" AcquirerPOIID="65-
        POITerm1"/>
    </PaymentResult>
  </PaymentResponse>
</SaleToPOIResponse>
```

7.5.6 CardAcquisitionRequest CardAcquisitionResponse

This section presents message examples 17 and 18.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="CardAcquisition"
    MessageType="Request" ServiceID="653" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardAcquisitionRequest>
    <SaleData>
      <SaleTransactionID TransactionID="589"TimeStamp="2009-08-
        05T22:17:52.33+01:00"/>
    </SaleData>
    <CardAcquisitionTransaction LoyaltyHandling="Forbidden"
      ForceCustomerSelectionFlag="true" />
  </CardAcquisitionRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="CardAcquisition"
    MessageType="Response" ServiceID="653" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardAcquisitionResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="589"TimeStamp="2009-08-
        05T22:17:52.33+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="491"TimeStamp="2009-08-
        05T22:18:34.06+01:00"/>
    </POIData>
    <PaymentBrand>CardFlash</PaymentBrand>
  </CardAcquisitionResponse>
</SaleToPOIResponse>
```

Followed by the messages example 19 and 20.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="654" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="589"TimeStamp="2009-08-
        05T22:17:52.33+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="174"/>
      <TransactionConditions LoyaltyHandling="Forbidden"/>
    </PaymentTransaction>
    <PaymentData PaymentType="Normal">
      <CardAcquisitionReference TransactionID="491"TimeStamp="2009-08-
        05T22:18:34.06+01:00"/>
    </PaymentData>
  </PaymentRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Response" ServiceID="654" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="589"TimeStamp="2009-08-
        05T22:17:52.33+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="491"TimeStamp="2009-08-
        05T22:18:34.06+01:00"/>
    </POIData>
    <PaymentResult PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData PaymentBrand="CardFlash" EntryMode="ICC">
          <ProtectedCardData ContentType="id-digestedData">
            <DigestedData>
              <DigestAlgorithm Algorithm="id-sha256"/>
              <EncapsulatedContent ContentType="id-data">
                <Digest>/1L/8m0+3TKHRrbFQU01mnGIPPqUUmGH2ByP+vdYcio=</Digest>
              </DigestedData>
            </ProtectedCardData>
          </CardData>
        </PaymentInstrumentData>
        <AmountsResp AuthorizedAmount="174"/>
        <PaymentAcquirerData AcquirerID="497867" MerchantID="mer77-130209"
          AcquirerPOIID="456">
          <ApprovalCode>9473</ApprovalCode>
        </PaymentAcquirerData>
      </PaymentResult>
    </PaymentResponse>
</SaleToPOIResponse>
```

7.5.7 Loyalty Services

7.5.7.1 Loyalty with Payment

This section presents message examples 21 and 22.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="651" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="581"TimeStamp="2009-08-
        09T17:17:21.9+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="38.01"/>
      <TransactionConditions LoyaltyHandling="Require"/>
    </PaymentTransaction>
    <PaymentData PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData EntryMode="Scanned">
          <SensitiveCardData PAN="7011014570541535"/>
        </CardData>
      </PaymentInstrumentData>
    </PaymentData>
    <LoyaltyData>
      <LoyaltyAccountID EntryMode="Scanned" IdentificationType="PAN">
        7011014570541535
      </LoyaltyAccountID>
    </LoyaltyData>
  </PaymentRequest>
</SaleToPOIRequest>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
    <MessageHeader MessageClass="Service" MessageCategory="Payment"
        MessageType="Response" ServiceID="651" SaleID="SaleTermA" POIID="POITerm1"/>
    <PaymentResponse>
        <Response Result="Success"/>
        <SaleData>
            <SaleTransactionID TransactionID="581"TimeStamp="2009-08-
                09T17:17:21.9+01:00"/>
        </SaleData>
        <POIData POIReconciliationID="200903101">
            <POITransactionID TransactionID="491"TimeStamp="2009-08-
                09T17:20:52.4+01:00"/>
        </POIData>
        <PaymentResult PaymentType="Normal">
            <PaymentInstrumentData PaymentInstrumentType="Card">
                <CardData PaymentBrand="MerRel" EntryMode="MagStripe">
                    <SensitiveCardData PAN="7011014570541535"/>
                </CardData>
            </PaymentInstrumentData>
            <AmountsResp AuthorizedAmount="38.01"/>
            <PaymentAcquirerData MerchantID="mer77-130209" AcquirerPOIID="65-POITerm1"/>
        </PaymentResult>
        <LoyaltyResult CurrentBalance="15">
            <LoyaltyAccount LoyaltyBrand="MerBonus">
                <LoyaltyAccountID EntryMode="Scanned" IdentificationType="PAN"
                    IdentificationSupport="HybridCard">7011014570541535</LoyaltyAccountID>
            </LoyaltyAccount>
            <LoyaltyAmount>5</LoyaltyAmount>
        </LoyaltyResult>
    </PaymentResponse>
</SaleToPOIResponse>
```

7.5.7.2 Payment with Rebates

This section presents message examples 23 and 24.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="643" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="592"TimeStamp="2015-04-
        06T18:26:34.1+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="38.52"/>
      <TransactionConditions LoyaltyHandling="Forbidden"/>
      <SaleItem ItemID="1" ProductCode="673" EanUpc="84116369" ItemAmount="2.30">
        <Quantity>5</Quantity>
        <UnitPrice>0.46</UnitPrice>
        <ProductLabel>Mineral Water 1,5L</ProductLabel>
      </SaleItem>
      <SaleItem ItemID="2" ProductCode="101" ItemAmount="33.22">
        <UnitOfMeasure>Litre</UnitOfMeasure>
        <Quantity>38.23</Quantity>
        <UnitPrice>0.869</UnitPrice>
        <ProductLabel>Diesel Fuel</ProductLabel>
      </SaleItem>
    </PaymentTransaction>
    <PaymentData PaymentType="Normal"/>
  </PaymentRequest>
</SaleToPOIRequest>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Response" ServiceID="643" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="592"TimeStamp="2015-04-
        06T18:26:34.1+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="479"TimeStamp="2015-04-
        06T18:26:43.9+01:00"/>
    </POIData>
    <PaymentResult PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData PaymentBrand="FleetUniverse" EntryMode="MagStripe">
          <SensitiveCardData PAN="6900014570541535" ExpiryDate="0411"/>
        </CardData>
      </PaymentInstrumentData>
      <AmountsResp AuthorizedAmount="38.37"/>
      <PaymentAcquirerData AcquirerID="690001" MerchantID="mer77-130209"
        AcquirerPOIID="963276433">
        <ApprovalCode>6541</ApprovalCode>
      </PaymentAcquirerData>
    </PaymentResult>
    <LoyaltyResult>
      <LoyaltyAccount LoyaltyBrand="FleetBonus">
        <LoyaltyAccountID EntryMode="MagStripe" IdentificationType="PAN"
          IdentificationSupport="LinkedCard">6900014570541535</LoyaltyAccountID>
      </LoyaltyAccount>
      <Rebates>
        <SaleItemRebate ItemID="1" ProductCode="673" ItemAmount="0.15">
          <RebateLabel>Special promotion</RebateLabel>
        </SaleItemRebate>
      </Rebates>
    </LoyaltyResult>
  </PaymentResponse>
</SaleToPOIResponse>
```

7.5.7.3 Loyalty Award Refund

This section presents message examples 25 to 28.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Loyalty"
    MessageType="Request" ServiceID="666" SaleID="SaleTermA" POIID="POITerm1"/>
  <LoyaltyRequest>
    <SaleData>
      <SaleTransactionID TransactionID="582"TimeStamp="2009-08-
        09T18:33:21.9+01:00"/>
    </SaleData>
    <LoyaltyTransaction LoyaltyTransactionType="Award" Currency="EUR"
      TotalAmount="48.19"/>
  </LoyaltyRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Loyalty"
    MessageType="Response" ServiceID="666" SaleID="SaleTermA" POIID="POITerm1"/>
  <LoyaltyResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="582"TimeStamp="2009-08-
        09T18:33:21.9+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="491"TimeStamp="2009-08-
        09T18:33:52.4+01:00"/>
    </POIData>
    <LoyaltyResult CurrentBalance="42">
      <LoyaltyAccount LoyaltyBrand="MerBonus">
        <LoyaltyAccountID EntryMode="Scanned" IdentificationType="AccountNumber"
          IdentificationSupport="NoCard">201401</LoyaltyAccountID>
      </LoyaltyAccount>
      <LoyaltyAmount>5</LoyaltyAmount>
    </LoyaltyResult>
  </LoyaltyResponse>
</SaleToPOIResponse>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Loyalty"
    MessageType="Request" ServiceID="683" SaleID="SaleTermA" POIID="POITerm1"/>
  <LoyaltyRequest>
    <SaleData>
      <SaleTransactionID TransactionID="593"TimeStamp="2009-08-
        09T18:35:44.9+01:00"/>
    </SaleData>
    <LoyaltyTransaction LoyaltyTransactionType="AwardRefund">
      <OriginalPOITransaction ReuseCardDataFlag="true">
        <POITransactionID TransactionID="491"TimeStamp="2009-08-
          09T18:33:52.4+01:00"/>
      </OriginalPOITransaction>
    </LoyaltyTransaction>
    <LoyaltyData>
      <LoyaltyAmount>5</LoyaltyAmount>
    </LoyaltyData>
  </LoyaltyRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Loyalty"
    MessageType="Response" ServiceID="683" SaleID="SaleTermA" POIID="POITerm1"/>
  <LoyaltyResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="593"TimeStamp="2009-08-
        09T18:35:44.9+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="504"TimeStamp="2009-08-
        09T18:36:52.2+01:00"/>
    </POIData>
    <LoyaltyResult CurrentBalance="37">
      <LoyaltyAccount LoyaltyBrand="MerBonus">
        <LoyaltyAccountID EntryMode="Scanned" IdentificationType="AccountNumber"
          IdentificationSupport="NoCard">201401</LoyaltyAccountID>
      </LoyaltyAccount>
      <LoyaltyAmount>5</LoyaltyAmount>
    </LoyaltyResult>
  </LoyaltyResponse>
</SaleToPOIResponse>
```

7.5.8 StoredValueRequest StoredValueResponse

This section presents message examples 29 and 30.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="StoredValue"
    MessageType="Request" ServiceID="630" SaleID="SaleTermA" POIID="POITerm1"/>
  <StoredValueRequest>
    <SaleData>
      <SaleTransactionID TransactionID="580" TimeStamp="2009-07-
        22T23:08:42.4+01:00"/>
    </SaleData>
    <StoredValueData StoredValueTransactionType="Load" ProductCode="512"
      ItemAmount="14.99" Currency="EUR"/>
  </StoredValueRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="StoredValue"
    MessageType="Response" ServiceID="630" SaleID="SaleTermA" POIID="POITerm1"/>
  <StoredValueResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="580" TimeStamp="2009-07-
        22T23:08:42.4+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="481" TimeStamp="2009-07-
        22T23:09:05.1+01:00"/>
    </POIData>
    <StoredValueResult StoredValueTransactionType="Load" ProductCode="512"
      ItemAmount="14.99" Currency="EUR">
      <StoredValueAccountStatus CurrentBalance="14.99">
        <StoredValueAccountID StoredValueAccountType="GiftCard"
          EntryMode="MagStripe" IdentificationType="AccountNumber">
          8678252755678</StoredValueAccountID>
        </StoredValueAccountStatus>
      </StoredValueResult>
    </StoredValueResponse>
</SaleToPOIResponse>
```

7.5.9 ReversalRequest ReversalResponse

This section presents message examples 31 to 34.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="633" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="582"TimeStamp="2009-08-
        09T20:06:33.1+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="38.01"/>
      <TransactionConditions LoyaltyHandling="Require"/>
    </PaymentTransaction>
    <PaymentData PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData EntryMode="Scanned">
          <SensitiveCardData PAN="7011014570541535"/>
        </CardData>
      </PaymentInstrumentData>
    </PaymentData>
  </PaymentRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Response" ServiceID="633" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentResponse>
    <Response Result="Success"/>
    <SaleData>
      <SaleTransactionID TransactionID="582"TimeStamp="2009-08-
        09T20:06:33.1+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200903101">
      <POITransactionID TransactionID="492"TimeStamp="2009-08-
        09T20:33:52.0+01:00"/>
    </POIData>
    <PaymentResult PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData PaymentBrand="MerRel" EntryMode="MagStripe">
          <SensitiveCardData PAN="7011014570541535"/>
        </CardData>
      </PaymentInstrumentData>
      <AmountsResp AuthorizedAmount="68.95"/>
      <PaymentAcquirerData MerchantID="mer77-130209" AcquirerPOIID="65-POITerm1"/>
    </PaymentResult>
  </PaymentResponse>
</SaleToPOIResponse>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Reversal"
    MessageType="Request" ServiceID="15" SaleID="SaleTermB" POIID="POIServer"/>
  <ReversalRequest ReversalReason="CustCancel">
    <OriginalPOITransaction SaleID="SaleTermA" POIID="POITerm1">
      <POITransactionID TransactionID="492"TimeStamp="2009-08-
        09T20:33:52.0+01:00"/>
    </OriginalPOITransaction>
  </ReversalRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Reversal"
    MessageType="Response" ServiceID="15" SaleID="SaleTermB" POIID="POIServer"/>
  <ReversalResponse>
    <Response Result="Success"/>
    <POIData>
      <POITransactionID TransactionID="178"TimeStamp="2009-08-
        09T21:19:15.3+01:00"/>
    </POIData>
  </ReversalResponse>
</SaleToPOIResponse>
```

7.5.10 BatchRequest BatchResponse

This section presents message examples 35 and 36.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Batch"
    MessageType="Request" ServiceID="670" SaleID="SaleTermA" POIID="POITerm1"/>
  <BatchRequest/>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Batch"
    MessageType="Response" ServiceID="670" SaleID="SaleTermA" POIID="POITerm1"/>
  <BatchResponse>
    <Response Result="Success"/>
    <PerformedTransaction>
      <Response Result="Success"/>
      <POIData POIReconciliationID="201406221">
        <POITransactionID TransactionID="481" TimeStamp="2014-06-
          22T23:08:42.4+01:00"/>
      </POIData/>
      <PaymentResult PaymentType="Normal">
        <PaymentInstrumentData PaymentInstrumentType="Card">
          <CardData PaymentBrand="CardPlus" EntryMode="MagStripe">
            <SensitiveCardData PAN="0011014570541535" ExpiryDate="0417"/>
          </CardData>
        </PaymentInstrumentData>
        <AmountsResp AuthorizedAmount="104.11"/>
        <PaymentAcquirerData AcquirerID="400012" MerchantID="mer77-130209">
          AcquirerPOIID="963276433">
            <ApprovalCode>8347</ApprovalCode>
          </PaymentAcquirerData>
        </PaymentResult>
      </PerformedTransaction>
      <PerformedTransaction>
        <Response Result="Partial"/>
        <POIData POIReconciliationID="201406221">
          <POITransactionID TransactionID="482" TimeStamp="2014-06-
            22T23:30:02.4+01:00"/>
        </POIData/>
        <PaymentResult PaymentType="Normal">
          <PaymentInstrumentData PaymentInstrumentType="Card">
            <CardData PaymentBrand="MerRel" EntryMode="MagStripe">
              <SensitiveCardData PAN="7011014570541535"/>
            </CardData>
          </PaymentInstrumentData>
          <AmountsResp AuthorizedAmount="31.12" TotalFeesAmount="0.19"/>
          <PaymentAcquirerData MerchantID="mer77-130209" AcquirerPOIID="65-
            POITerm1"/>
        </PaymentResult>
      </PerformedTransaction>
    </BatchResponse>
</SaleToPOIResponse>
```

7.5.11 ReconciliationRequest ReconciliationResponse

This section presents message examples 37 and 38.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Reconciliation"
    MessageType="Request" ServiceID="617" SaleID="SaleServer" POIID="POIServer"/>
  <ReconciliationRequest ReconciliationType="SaleReconciliation"/>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Reconciliation"
    MessageType="Response" ServiceID="617" SaleID="SaleServer" POIID="POIServer"/>
  <ReconciliationResponse POIReconciliationID="8926"
    ReconciliationType="SaleReconciliation">
    <Response Result="Success"/>
    <TransactionTotals PaymentInstrumentType="Card" AcquirerID="876355543"
      HostReconciliationID="98535" CardBrand="Visa" PaymentCurrency="EUR">
      <PaymentTotals TransactionType="Debit" TransactionCount="182"
        TransactionAmount="47184.17"/>
      <PaymentTotals TransactionType="Credit" TransactionCount="1"
        TransactionAmount="27.01"/>
    </TransactionTotals>
    <TransactionTotals PaymentInstrumentType="Card" AcquirerID="876355543"
      HostReconciliationID="98535" CardBrand="ExpressCard" PaymentCurrency="EUR">
      <PaymentTotals TransactionType="Debit" TransactionCount="157"
        TransactionAmount="3753.61"/>
      <PaymentTotals TransactionType="ReverseDebit" TransactionCount="1"
        TransactionAmount="37.99"/>
    </TransactionTotals>
    <TransactionTotals PaymentInstrumentType="Card" AcquirerID="93582"
      CardBrand="SuperBonus">
      <LoyaltyTotals TransactionType="Award" TransactionCount="143"
        TransactionAmount="1889"/>
    </TransactionTotals>
  </ReconciliationResponse>
</SaleToPOIResponse>
```

7.5.12 GetTotalsRequest GetTotalsResponse

This section presents message examples 39 and 40.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="GetTotals"
    MessageType="Request" ServiceID="619" SaleID="SaleServer"
    POIID="POIServer"/>
  <GetTotalsRequest>
    <TotalDetails>SaleID ShiftNumber</TotalDetails>
    <TotalFilter OperatorID="213"/>
  </GetTotalsRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="GetTotals"
    MessageType="Response" ServiceID="619" SaleID="SaleServer"
    POIID="POIServer"/>
  <GetTotalsResponse POIReconciliationID="8927">
    <Response Result="Success"/>
    <TransactionTotals PaymentInstrumentType="Card" AcquirerID="876355543"
      HostReconciliationID="98535" CardBrand="CardPlus" SaleID="SaleTermA"
      OperatorID="213" ShiftNumber="1" PaymentCurrency="EUR">
      <PaymentTotals TransactionType="Debit" TransactionCount="61"
        TransactionAmount="4253.19"/>
      <PaymentTotals TransactionType="Credit" TransactionCount="1"
        TransactionAmount="27.01"/>
    </TransactionTotals>
    <TransactionTotals PaymentInstrumentType="Card" AcquirerID="876355543"
      HostReconciliationID="98535" CardBrand="CardPlus" SaleID="SaleTermD"
      OperatorID="213" ShiftNumber="2" PaymentCurrency="EUR">
      <PaymentTotals TransactionType="Debit" TransactionCount="45"
        TransactionAmount="744.79"/>
    </TransactionTotals>
    <TransactionTotals PaymentInstrumentType="Card" AcquirerID="876355543"
      CardBrand="ExpressCard" HostReconciliationID="98535" SaleID="SaleTermD"
      OperatorID="29" ShiftNumber="1" PaymentCurrency="EUR">
      <PaymentTotals TransactionType="Debit" TransactionCount="17"
        TransactionAmount="353.61"/>
    </TransactionTotals>
    <TransactionTotals PaymentInstrumentType="Card" AcquirerID="876355543"
      CardBrand="ExpressCard" HostReconciliationID="98535" SaleID="SaleTermD"
      OperatorID="213" ShiftNumber="2" PaymentCurrency="EUR">
      <PaymentTotals TransactionType="Debit" TransactionCount="18"
        TransactionAmount="711.48"/>
      <PaymentTotals TransactionType="ReverseDebit" TransactionCount="1"
        TransactionAmount="37.99"/>
    </TransactionTotals>
    <TransactionTotals PaymentInstrumentType="Card" AcquirerID="93582"
      CardBrand="SuperBonus" SaleID="SaleTermD" OperatorID="213"
      ShiftNumber="1">
      <LoyaltyTotals TransactionType="Award" TransactionCount="25"
        TransactionAmount="278"/>
    </TransactionTotals>
    <TransactionTotals PaymentInstrumentType="Card" AcquirerID="93582"
      CardBrand="SuperBonus" SaleID="SaleTermD" OperatorID="213"
      ShiftNumber="2">
      <LoyaltyTotals TransactionType="Award" TransactionCount="23"
        TransactionAmount="182"/>
    </TransactionTotals>
  </GetTotalsResponse>
```

</SaleToPOIResponse>

7.5.13 BalanceInquiryRequest BalanceInquiryResponse

This section presents message examples 41 and 42.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="BalanceInquiry"
    MessageType="Request" ServiceID="629" SaleID="SaleTermA" POIID="POITerm1"/>
  <BalanceInquiryRequest/>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="BalanceInquiry"
    MessageType="Response" ServiceID="629" SaleID="SaleTermA" POIID="POITerm1"/>
  <BalanceInquiryResponse>
    <Response Result="Success"/>
    <PaymentAccountStatus CurrentBalance="832.59" Currency="EUR">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData PaymentBrand="CardPlus" EntryMode="MagStripe">
          <SensitiveCardData PAN="6011014570541535" ExpiryDate="0411"/>
        </CardData>
      </PaymentInstrumentData>
      <PaymentAcquirerData AcquirerID="400012" MerchantID="mer77-130209"
        AcquirerPOIID="963276433">
        <ApprovalCode>8347</ApprovalCode>
      </PaymentAcquirerData>
    </PaymentAccountStatus>
    <LoyaltyAccountStatus CurrentBalance="112" LoyaltyUnit="Point">
      <LoyaltyAccount LoyaltyBrand="SuperBonus">
        <LoyaltyAccountID EntryMode="MagStripe" IdentificationType="PAN"
          IdentificationSupport="LinkedCard">8678252755678</LoyaltyAccountID>
      </LoyaltyAccount>
    </LoyaltyAccountStatus>
  </BalanceInquiryResponse>
</SaleToPOIResponse>
```

7.5.14 DisplayRequest DisplayResponse

This section presents message examples 43 and 44.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Display"
    MessageType="Request" ServiceID="674" DeviceID="2" SaleID="SaleTermA"
    POIID="POITerm1"/>
  <DisplayRequest>
    <DisplayOutput Device="CashierDisplay" InfoQualify="Status">
      <OutputContent OutputFormat="Text">
        <OutputText>Bank Authorization</OutputText>
      </OutputContent>
    </DisplayOutput>
    <DisplayOutput ResponseRequiredFlag="false" Device="CashierDisplay"
      InfoQualify="POIReplication">
      <OutputContent OutputFormat="Text">
        <OutputText>EMV Payment Processing</OutputText>
      </OutputContent>
    </DisplayOutput>
    <DisplayOutput Device="CashierDisplay" InfoQualify="Error">
      <OutputContent OutputFormat="Text">
        <OutputText>Card request online authorisation</OutputText>
      </OutputContent>
    </DisplayOutput>
  </DisplayRequest>
</SaleToPOIRequest>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Display"
    MessageType="Response" ServiceID="674" DeviceID="2" SaleID="SaleTermA"
    POIID="POITerm1"/>
  <DisplayResponse>
    <OutputResult Device="CashierDisplay" InfoQualify="Status">
      <Response Result="Success"/>
    </OutputResult>
    <OutputResult Device="CashierDisplay" InfoQualify="Error">
      <Response Result="Success"/>
    </OutputResult>
  </DisplayResponse>
</SaleToPOIResponse>
```

7.5.15 InputRequest InputResponse

7.5.15.1 Display Confirmation

This section presents message examples 45 and 46.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Input"
    MessageType="Request" DeviceID="1374" SaleID="SaleTermA" POIID="POITerm1"/>
  <InputRequest>
    <DisplayOutput Device="CustomerDisplay" InfoQualify="Display">
      <OutputContent OutputFormat="Text">
        <OutputText>Confirm the Car Wash</OutputText>
      </OutputContent>
    </DisplayOutput>
    <InputData Device="CustomerInput" InfoQualify="Input"
      InputCommand="GetAnyKey"/>
  </InputRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Input"
    MessageType="Response" DeviceID="1374" SaleID="SaleTermA" POIID="POITerm1"/>
  <InputResponse>
    <InputResult Device="CustomerInput" InfoQualify="Input">
      <Response Result="Success"/>
      <Input InputCommand="GetAnyKey"/>
    </InputResult>
  </InputResponse>
</SaleToPOIResponse>
```

7.5.15.2 Information Request

This section presents message examples 47 and 48.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Input"
    MessageType="Request" DeviceID="1375" SaleID="SaleTermA" POIID="POITerm1"/>
  <InputRequest>
    <DisplayOutput Device="CustomerDisplay" InfoQualify="Display">
      <OutputContent OutputFormat="Text">
        <OutputText>Enter the mileage in miles :</OutputText>
      </OutputContent>
    </DisplayOutput>
    <InputData Device="CustomerInput" InfoQualify="Input"
      InputCommand="DigitString"/>
  </InputRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Input"
    MessageType="Response" DeviceID="1375" SaleID="SaleTermA" POIID="POITerm1"/>
  <InputResponse>
    <InputResult Device="CustomerInput" InfoQualify="Input">
      <Response Result="Success"/>
      <Input InputCommand="DigitString">
        <DigitInput>43824</DigitInput>
      </Input>
    </InputResult>
  </InputResponse>
</SaleToPOIResponse>
```

7.5.15.3 Menu Item Selection

This section presents message examples 49 and 50.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Input"
    MessageType="Request" ServiceID="675" DeviceID="4" SaleID="SaleTermA"
    POIID="POITerm1"/>
  <InputRequest>
    <DisplayOutput Device="CustomerDisplay" InfoQualify="Display">
      <OutputContent OutputFormat="Text">
        <OutputText>Choose the report:</OutputText>
      </OutputContent>
      <MenuEntry OutputFormat="Text">
        <OutputText>1: Daily Payment Report</OutputText>
      </MenuEntry>
      <MenuEntry OutputFormat="Text" MenuEntryTag="NonSelectable">
        <OutputText>2: Daily Loyalty Report</OutputText>
      </MenuEntry>
      <MenuEntry OutputFormat="Text" MenuEntryTag="SubMenu">
        <OutputText>3: Other Reports</OutputText>
      </MenuEntry>
    </DisplayOutput>
    <InputData Device="CustomerInput" InfoQualify="Input"
      InputCommand="GetMenuEntry"/>
  </InputRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Input"
    MessageType="Response" ServiceID="675" DeviceID="4" SaleID="SaleTermA"
    POIID="POITerm1"/>
  <InputResponse>
    <InputResult Device="CustomerInput" InfoQualify="Input">
      <Response Result="Success"/>
      <Input InputCommand="GetMenuEntry">
        <MenuEntryNumber>1</MenuEntryNumber>
      </Input>
    </InputResult>
  </InputResponse>
</SaleToPOIResponse>
```

7.5.16 Input Update

This section presents message examples 51.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="InputUpdate"
    MessageType="Request" DeviceID="260" SaleID="SaleTermA" POIID="POITerm1"/>
  <InputUpdate>
    <MessageReference DeviceID="259">
      <OutputContent OutputFormat="Text">
        <OutputText>Select your car wash program (credit=5 euros)</OutputText>
      </OutputContent>
    </InputUpdate>
  </SaleToPOIRequest>
```

7.5.17 PrintRequest PrintResponse

This section presents message examples 52 and 53.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Print"
    MessageType="Request" ServiceID="675" DeviceID="1375" SaleID="SaleTermA"
    POIID="POITerm1"/>
  <PrintRequest>
    <PrintOutput DocumentQualifier="CustomerReceipt" ResponseMode="PrintEnd">
      <OutputContent OutputFormat="XHTML">
        <OutputXHTML>
          PGJpZz48Yj5QYXBhZGVhdXggU2VhZm9vZCBLaXRjaGVuPC9iPjwvYmlnPjxwPjEwNzkgRGF2a
          XMgRHJuIC0gQWxwaGFyZXR0YSBHQSAzMjwvD48cD4xMjo1NDowNyAwNC8zMC8yMDA5PC
          9wPjxwPlZpc2EgKioqKio1OTczPC9wPjxiaoWc+PHA+QW1vdW50Oia8Yj4kMTguMDA8L2I+PC9
          wPjxwPkF1dGggQ29kZTogMjM3MTYyPC9wPjwvYmlnPj48cD5DdXN0b211ciBDb3B5PC9wPg==
        </OutputXHTML>
      </OutputContent>
    </PrintOutput>
  </PrintRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Print"
    MessageType="Response" ServiceID="675" DeviceID="1375" SaleID="SaleTermA"
    POIID="POITerm1"/>
  <PrintResponse DocumentQualifier="CustomerReceipt">
    <Response Result="Success"/>
  </PrintResponse>
</SaleToPOIResponse>
```

7.5.18 SoundRequest SoundResponse

This section presents message examples 54 and 55.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Sound"
    MessageType="Request" DeviceID="259" SaleID="SaleTermA" POIID="POITerm1"/>
  <SoundRequest ResponseMode="Immediate" SoundAction="StartSound" SoundVolume="50">
    <SoundContent SoundFormat="SoundRef" ReferenceID="ambiance02.mp3"/>
  </SoundRequest>
</SaleToPOIRequest>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Sound"
    MessageType="Response" DeviceID="259" SaleID="SaleTermA" POIID="POITerm1"/>
  <SoundResponse>
    <Response Result="Success"/>
  </SoundResponse>
</SaleToPOIResponse>
```

7.5.19 PINRequest PINResponse

This section presents message examples 56 and 57.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="PIN" MessageType="Request"
    DeviceID="375" SaleID="SaleTermA" POIID="POITerm1"/>
  <PINRequest PINRequestType="PINVerify" PINVerifMethod="Local1"
    AdditionalInput="05543"/>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="PIN" MessageType="Response"
    DeviceID="375" SaleID="SaleTermA" POIID="POITerm1"/>
  <PINResponse>
    <Response Result="Success"/>
  </PINResponse>
</SaleToPOIResponse>
```

7.5.20 CardReaderInitRequest CardReaderInitResponse

7.5.20.1 Magstripe Card Reading

This section presents message examples 58 and 59.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="CardReaderInit"
    MessageType="Request" DeviceID="385" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardReaderInitRequest>
    <DisplayOutput Device="CustomerDisplay" InfoQualify="Display">
      <OutputContent OutputFormat="Text">
        <OutputText>Please Enter Your Card</OutputText>
      </OutputContent>
    </DisplayOutput>
  </CardReaderInitRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="CardReaderInit"
    MessageType="Response" DeviceID="385" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardReaderInitResponse EntryMode="MagStripe">
    <Response Result="Success"/>
    <TrackData TrackNumb="1">
      %B4970100202013617^CARTE DE DEVELOPPEMENT^110590195809700000000000000000000?
    </TrackData>
    <TrackData>4970100202013617=11059019580970000000?</TrackData>
  </CardReaderInitResponse>
</SaleToPOIResponse>
```

7.5.20.2 ICC Card Reading

This section presents message examples 60 and 61.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="CardReaderInit"
    MessageType="Request" DeviceID="421" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardReaderInitRequest>
    <DisplayOutput Device="CustomerDisplay" InfoQualify="Display">
      <OutputContent OutputFormat="Text">
        <OutputText>Please Enter Your Card</OutputText>
      </OutputContent>
    </DisplayOutput>
  </CardReaderInitRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="CardReaderInit"
    MessageType="Response" DeviceID="421" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardReaderInitResponse EntryMode="ICC">
    <Response Result="Success"/>
    <ICCResetData ATRValue="YAA7ZQAAAAAAAAAAAAAVYBGyQAAAAAAAAAAAAA=" CardStatusWords="kAA="/>
  </CardReaderInitResponse>
</SaleToPOIResponse>
```

7.5.20.3 Card Reader APDU

This section presents message examples 62 and 63.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="CardReaderAPDU"
    MessageType="Request" DeviceID="422" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardReaderAPDUREquest APDUClass="AA==" APDUInstruction="pA==" APDUPar1="BA=="
    APDUPar2="AA==" APDUExpectedLength="Bw==">
    <APDUData>oAAAAEIQEA==</APDUData>
  </CardReaderAPDUREquest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="CardReaderAPDU"
    MessageType="Response" DeviceID="422" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardReaderAPDUREsponse CardStatusWords="kAA==">
    <Response Result="Success"/>
    <APDUData>
      by2EB6AAAABCEBC1I1ACQ0KHAQGfEgdURVNUIENCnxEBAb8MBd9gAgsUXy0CZnI=
    </APDUData>
  </CardReaderAPDUREsponse>
</SaleToPOIResponse>
```

7.5.20.4 Card Reader Power-Off

This section presents message examples 64 and 65.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="CardReaderPowerOff"
    MessageType="Request" DeviceID="423" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardReaderPowerOffRequest>
    <DisplayOutput Device="CustomerDisplay" InfoQualify="Display">
      <OutputContent OutputFormat="Text">
        <OutputText>Please Remove Your Card</OutputText>
      </OutputContent>
    </DisplayOutput>
  </CardReaderPowerOffRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="CardReaderPowerOff"
    MessageType="Response" DeviceID="423" SaleID="SaleTermA" POIID="POITerm1"/>
  <CardReaderPowerOffResponse>
    <Response Result="Success"/>
  </CardReaderPowerOffResponse>
</SaleToPOIResponse>
```

7.5.21 TransmitRequest TransmitResponse

This section presents message examples 66 and 67.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Transmit"
    MessageType="Request" ServiceID="675" DeviceID="1" SaleID="SaleTermA"
    POIID="POITerm1"/>
  <TransmitRequest MaximumTransmitTime="30" DestinationAddress="acquirer.com:8080">
    <Message>
      PEF1dGhSZXAgQW1vdW50PSIxMDAwIiBQQU49IjkxNjQ2NSI+PC9BdXR0UmVxPg==
    </Message>
  </TransmitRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Device" MessageCategory="Transmit"
    MessageType="Response" ServiceID="675" DeviceID="1" SaleID="SaleTermA"
    POIID="POITerm1"/>
  <TransmitResponse>
    <Response Result="Success"/>
    <Message>
      PEF1dGhSZXAgQW1vdW50PSIxMDAwIiBSZXNwb25zZT0iQXBwcm92ZWQiIEF1dGhDb2RlPSI3OTQ1
      ICAiPjwvQXV0aFJlcD4=
    </Message>
  </TransmitResponse>
</SaleToPOIResponse>
```

7.5.22 TransactionStatusRequest TransactionStatusResponse

This section presents message examples 68 to 70.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="642" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="579" TimeStamp="2009-06-
        07T23:08:42.4+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="104.11"/>
      <TransactionConditions LoyaltyHandling="Forbidden"/>
    </PaymentTransaction>
    <PaymentData PaymentType="Normal"/>
  </PaymentRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="TransactionStatus"
    MessageType="Request" ServiceID="498" SaleID="SaleTermB" POIID="POITerm1"/>
  <TransactionStatusRequest>
    <MessageReference MessageCategory="Payment" ServiceID="642"
      SaleID="SaleTermA"/>
  </TransactionStatusRequest>
</SaleToPOIRequest>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="TransactionStatus"
    MessageType="Response" ServiceID="498" SaleID="SaleTermB" POIID="POITerm1"/>
  <TransactionStatusResponse>
    <Response Result="Success"/>
    <MessageReference MessageCategory="Payment" ServiceID="642"
      SaleID="SaleTermA"/>
    <RepeatedMessageResponse>
      <MessageHeader MessageClass="Service" MessageCategory="Payment"
        MessageType="Response" ServiceID="642" SaleID="SaleTermA"
        POIID="POITerm1"/>
    <PaymentResponse>
      <Response Result="Success"/>
      <SaleData>
        <SaleTransactionID TransactionID="579" TimeStamp="2009-06-
          07T23:08:42.4+01:00"/>
      </SaleData>
      <POIData POIReconciliationID="200903101">
        <POITransactionID TransactionID="481" TimeStamp="2009-03-
          10T23:08:42.4+01:00"/>
      </POIData>
      <PaymentResult PaymentType="Normal">
        <PaymentInstrumentData PaymentInstrumentType="Card">
          <CardData PaymentBrand="CardPlus" EntryMode="MagStripe">
            <SensitiveCardData PAN="0011014570541535" ExpiryDate="0411"/>
          </CardData>
        </PaymentInstrumentData>
        <AmountsResp AuthorizedAmount="104.11"/>
        <PaymentAcquirerData AcquirerID="400012" MerchantID="mer77-130209"
          AcquirerPOIID="963276433">
          <ApprovalCode>8347</ApprovalCode>
        </PaymentAcquirerData>
      </PaymentResult>
    </PaymentResponse>
    </RepeatedMessageResponse>
  </TransactionStatusResponse>
</SaleToPOIResponse>
```

7.5.23 AbortRequest

This section presents message example 71 to 73.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Request" ServiceID="642" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentRequest>
    <SaleData>
      <SaleTransactionID TransactionID="579" TimeStamp="2009-06-
        09T23:12:42.4+01:00"/>
    </SaleData>
    <PaymentTransaction>
      <AmountsReq Currency="EUR" RequestedAmount="104.11"/>
      <TransactionConditions LoyaltyHandling="Forbidden"/>
    </PaymentTransaction>
    <PaymentData PaymentType="Normal"/>
  </PaymentRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Abort"
    MessageType="Request" ServiceID="498" SaleID="SaleTermA" POIID="POITerm1"/>
  <AbortRequest>
    <MessageReference MessageCategory="Payment" ServiceID="642"/>
    <AbortReason>Waiting time</AbortReason>
  </AbortRequest>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Service" MessageCategory="Payment"
    MessageType="Response" ServiceID="642" SaleID="SaleTermA" POIID="POITerm1"/>
  <PaymentResponse>
    <Response Result="Failure" ErrorCondition="Aborted">
      <AdditionalResponse>Service Aborted during Online Authorization - Reason:
        Waiting time - from: SaleTermA - MessageID: 498</AdditionalResponse>
    </Response>
    <SaleData>
      <SaleTransactionID TransactionID="579" TimeStamp="2009-06-
        09T23:12:42.4+01:00"/>
    </SaleData>
    <POIData POIReconciliationID="200906091">
      <POITransactionID TransactionID="481" TimeStamp="2009-06-
        09T23:15:12.4+01:00"/>
    </POIData>
    <PaymentResult PaymentType="Normal">
      <PaymentInstrumentData PaymentInstrumentType="Card">
        <CardData PaymentBrand="CardPlus" EntryMode="MagStripe">
          <SensitiveCardData PAN="0011014570541535" ExpiryDate="0411"/>
        </CardData>
      </PaymentInstrumentData>
      <AmountsResp AuthorizedAmount="104.11"/>
    </PaymentResult>
  </PaymentResponse>
</SaleToPOIResponse>
```

7.5.24 EventNotification

This section presents message example 74.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader MessageClass="Event" MessageCategory="Event"
    MessageType="Notification" DeviceID="1328" SaleID="SaleTermA"
    POIID="POITerm1"/>
  <EventNotification TimeStamp="2009-12-13T23:11:16.4+01:00"
    EventToNotify="BeginMaintenance">
    <EventDetails>POITerm1 Maintenance: Parameters Update</EventDetails>
  </EventNotification>
</SaleToPOIRequest>
```

7.5.25 DiagnosisRequest DiagnosisResponse

This section presents message examples 75 and 76.

```
<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader ProtocolVersion="3.0" MessageClass="Service"
    MessageCategory="Diagnosis" MessageType="Request" ServiceID="660"
    SaleID="SaleTermA" POIID="POITerm1"/>
  <DiagnosisRequest HostDiagnosisFlag="true"/>
</SaleToPOIRequest>

<?xml version="1.0" encoding="UTF-8"?>
<SaleToPOIResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="EpasSaleToPOIMessages.xsd">
  <MessageHeader ProtocolVersion="3.0" MessageClass="Service"
    MessageCategory="Diagnosis" MessageType="Response" ServiceID="660"
    SaleID="SaleTermA" POIID="POITerm1"/>
  <DiagnosisResponse>
    <Response Result="Success"/>
    <LoggedSaleID>SaleTermA</LoggedSaleID>
    <POIStatus GlobalStatus="OK" SecurityOKFlag="true" PEDOKFlag="true"
      CardReaderOKFlag="true" PrinterStatus="OK" CommunicationOKFlag="true"/>
    <HostStatus AcquirerID="64592" IsReachableFlag="true"/>
    <HostStatus AcquirerID="50265" IsReachableFlag="true"/>
  </DiagnosisResponse>
</SaleToPOIResponse>
```

8 Annex C Examples of Architectures

Considering the variety of architecture and implementation, there are necessary a lot of configurations not covered in this section. Presented implementations retain the terminology used in these concrete implementations, the linked to the wording we have defined in generic architecture is indicated when necessary.

8.1 Petrol Station

A typical petrol station includes two different types of payment:

- *Indoor Payments*, which are comparable to payment organisation in a convenient store,
- *Outdoor Payments*, which are unattended payment mainly at the petrol pump, but also for car wash or other services.

Indoor payments are managed by a Sale System and a Payment System, with the type we have already seen in the previous section.

In the other hand, outdoor payments are quite specifics:

1. The payment terminal inside the pump is often designed by the pump manufacturer, and managed by the pump.
2. The forecourt controller, which constitutes part of the Delivery System, manages all the pumps, and is operated manually or automatically by the cashier or the POS³⁰.

The logical architecture of a petrol station is presented below, including the Sale System (ECR and ECR Controller), handles the Delivery System (Petrol Pump distribution and Forecourt Controller) and the POI System (Indoor Payment Terminals, Outdoor Payment Terminals and Payment Server).

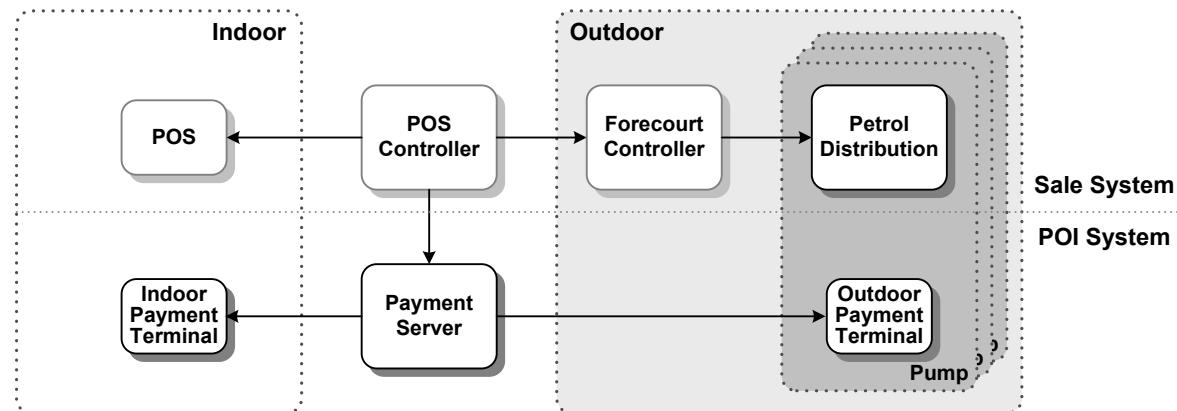


Figure 257: Logical Architecture of a Petrol Station

This is the architecture of Distributed POI System, where in case of outdoor payment terminal, the POI Terminal is associated to a logical Sale Terminal.

³⁰ Forecourt controller can be used as a communication gateway by the outdoor payment terminal. The outdoor payment terminals could however be linked to the Payment Server independently from the forecourt controller.

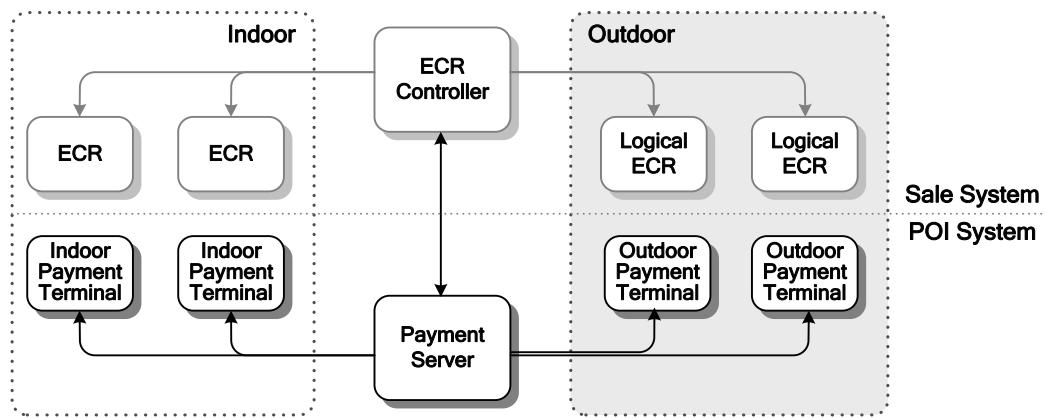


Figure 258: Distributed POI in Petrol Station

Considering the huge investment for a pump manufacturer to develop the payment terminal part of the pump, tendency is to separate these two products. As the unattended outdoor payment terminal offering is quite specific, compared to the pin-pad or standalone payment terminal market, there is a lack of standard to connect an outdoor payment terminal to a payment server.

8.2 Supermarket and Department Store

Supermarket and department store share the following features:

- An important number of lanes or POS,
- The majority of POS (sale terminals) are identical and aligned for supermarket, but some of them are very different and can request specific processing, or can belong to another owner. Some POS can be also semi-attended.

On the other hand, the purchases are processed quite differently:

- For supermarkets, transaction time is quite long, and contains dozen of items per purchase,
- For department store, transaction time is quite short, and contains few items per purchase,

The payment system is more often a Clustered POI System.

Example on Server Client

8.3 Car Park

The car park is equipped with three types of payment terminals:

- Inside the car at the parking barrier, registration of the card during the entry in the parking, and unattended payment with the card at the exit.
- Outside the car with automatic payment machines, unattended payment before get back the car to exit the parking.
- On an attended point of sale, inside the car at the barrier gate, or outside the car in the central supervision of the parking.

All these payment terminals are connected to a payment sever located in the parking or centralised, depending on the parking size.

There is a very simple interface between the Sale software in the various parking machine and the payment terminals to request payment. The payment terminal processes the minimum of functions, card reading and user interface, the payment server manages all the other functions.

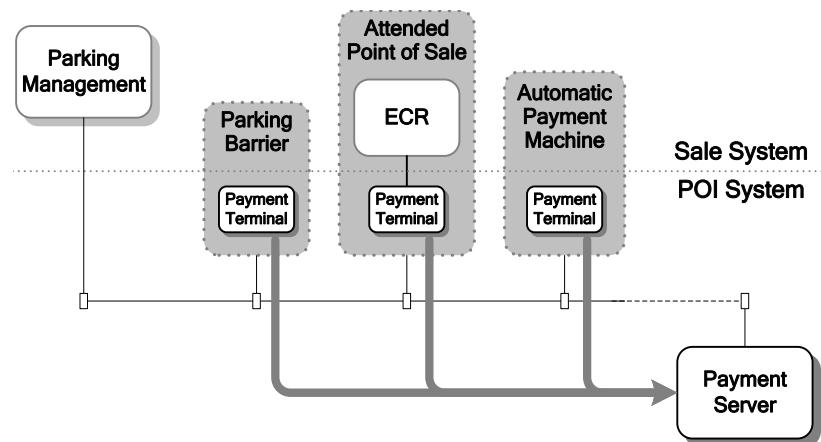


Figure 259: Car Park Architecture

This is the architecture of Clustered POI System, where all the transactions are driven by the POI Terminal and sent to a local or remote POI Server.

8.4 Vending Machine

We differentiate two kinds of vending machines:

1. Those accepting standard card scheme, which include a payment terminal and are on-line (e.g. ticketing).
2. Those accepting mainly cash and pre-paid cards in a closed and off-line environment (e.g. laundry, game, universities, prison, cafeteria...).

These two categories are in a public or private location. They use a communication gateway to send sale information and for the monitoring of the vending machine.

In case of type 1, a payment server can be present to manage several vending machines in the same area, and to reach an Acquirer Host.

In case of type 2, on-line payment solutions start to be installed using the communication gateway for this purpose, with magnetic stripe or contact-less card readers.

8.5 e-Commerce

Payment at home on open networks, realises more and more transactions each year, and are accomplished as MOTO (Mail Order Telephone Order) payment transactions, without a card reader.

Typical architecture for these payments includes:

- A *merchant Web server*, which corresponds to the store and the Sale System, and contains the product catalogue with sophisticated functionalities and the shopping card management.
- A *payment Web server* which corresponds to the POI System, and process payment part of the purchases,
- The *Cardholder* at home, which interacts with these two Web servers with a browser.

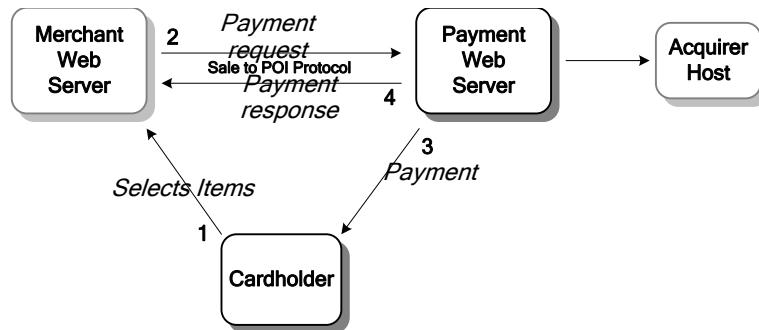


Figure 260: e-Commerce Architecture

The sequence flow of a transaction is the following:

1. The Cardholder chooses products in the catalogue, fills up shopping cart and decides to pay.
2. The merchant Web server prepares and sends a payment request to the payment Web server.
3. The payment Web server processes the payment transaction with the Cardholder and the Acquirer Host.
4. The payment Web server ends the payment and sends the result to the merchant Web server, which finishes the purchase transaction.

Even if the interface between the Cardholder and the payment Web server is very specific, the interface between the merchant Web server and the payment Web server can follow the Sale to POI protocol.

In some cases, a POI component can process the payment at the Cardholder site with a plug-in or a card reader.

Other architectures also propose a centralised role for the Cardholder, where after getting the payment data (1), he is directed to his Issuer host via a card scheme directory (2 and 2'), authenticated and debited by the Issuer (3), and bring the proof of the payment at the Issuer to the Acquirer (4 and 4').

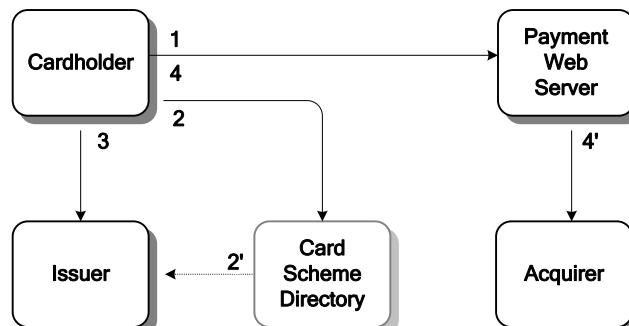


Figure 261: Cardholder Centralised Architecture