SET Secure Electronic Transaction Specification

Book 3: Formal Protocol Definition

Version 1.0
May 31, 1997
Preface

Introduction
The development of electronic commerce is at a critical juncture.

- Consumer demand for secure access to electronic shopping and other services is very high.
- Merchants want simple, cost-effective methods for conducting electronic transactions.
- Financial institutions want a level playing field for software suppliers to ensure quality products at competitive prices.
- Payment card brands must be able to differentiate electronic commerce transactions without significant impact to the existing infrastructure.

The next step toward achieving secure, cost-effective, on-line transactions that will satisfy market demand is the development of a single, open industry specification.

Secure Electronic Transaction protocol
Visa and MasterCard have jointly developed the SET Secure Electronic Transaction protocol as a method to secure payment card transactions over open networks. SET is being published as an open specification for the industry. This specification is available to be applied to any payment card service and may be used by software vendors to develop applications.

Advice and assistance in the development of these specifications have been provided by GTE, IBM, Microsoft, Netscape, RSA, SAIC, Terisa, and Verisign.

Cardholder and merchant software
This document contains the formal protocol definition for the SET protocol. It is primarily intended for use by:

- cryptographers analyzing security,
- writers producing programming guides, and
- system programmers developing cryptographic and messaging primitives.

Payment gateway and certificate authority software
While this specification provides the interface to the Payment Gateway and the certificate authority, it does not provide all necessary information for a software vendor to create these systems. Specifically, the specification does not address the interface between the Payment Gateway and the existing financial system and it does not address the mechanism for the processing of certificate requests, which depend on payment card brand policy.

Continued on next page
Preface, continued

Necessary background

Many vendors will have developed software that either interfaces with payment systems or uses public-key cryptography, but few will have done both. This document does not attempt to provide detailed information on these subjects. Book 1 contains introductory material that provides a primer on these topics. Readers are encouraged to study additional background material in these areas. (See “Related Documentation” on next page.)

Related documentation

The following articles and books contain additional background material. Readers are encouraged to consult these references for more information.


(http://www.rsa.com/rsa/prodspec/bsafe/rsa_bsafe.htm)


“The HMAC Construction,” Mihir Bellare, Ran Canetti, and Hugo Krawczyk, *CryptoBytes*, volume 2, number 1, Spring 1996

*HTML Sourcebook*, Ian S. Graham, John Wiley & Sons, Inc., 1995


*Optimal Asymmetric Encryption*, M. Bellare and P. Rogaway, Eurocrypt 94.
(http://www.cse.ucsd.edu/users/mihir/papers/oae.ps.gz)


## Table of Contents

Preface ........................................................................................................................................... i

Part I Formal Protocol Definition ...................................................................................................... 1
  Introduction ................................................................................................................................. 1
  Notation ..................................................................................................................................... 3

Chapter 1 Cryptography .................................................................................................................. 4
  Overview ..................................................................................................................................... 4
  Entities ......................................................................................................................................... 5
  Hashing and Hash-based operators ................................................................................................. 6
  Signature Primitives ....................................................................................................................... 8
  Encryption Primitives .................................................................................................................... 9
  Encapsulation Operations ............................................................................................................. 12
  Optimal Asymmetric Encryption Padding (OAEP) ................................................................. 15

Chapter 2 Message Encapsulation .................................................................................................... 24
  MessageWrapper ........................................................................................................................ 25
  Error Message ............................................................................................................................ 30

Chapter 3 Payment Message Components ..................................................................................... 32
  Overview ..................................................................................................................................... 32
  TransIDs ...................................................................................................................................... 33
  RRTags ......................................................................................................................................... 34
  PI (Payment Instruction) .............................................................................................................. 35
  InstallRecurData .......................................................................................................................... 42
  AcqCardMsg ............................................................................................................................... 43
  CapToken ..................................................................................................................................... 44
  PANData ..................................................................................................................................... 45
  PANToken .................................................................................................................................... 46
  BatchStatus ................................................................................................................................. 47
  Location ....................................................................................................................................... 52
  SaleDetail ..................................................................................................................................... 53

Chapter 4 Payment Messages .......................................................................................................... 71
  Overview ..................................................................................................................................... 71
  Purchase Initialization Pair ......................................................................................................... 72
  Purchase Pair ............................................................................................................................... 76
  Purchase Inquiry Pair ..................................................................................................................... 90
  Authorization Pair ....................................................................................................................... 92
  Authorization Reversal Pair ........................................................................................................ 106
  Capture Pair ............................................................................................................................... 112
  Capture Reversal Or Credit ........................................................................................................ 121
  Capture Reversal Pair ................................................................................................................ 129
  Credit Pair .................................................................................................................................. 132
  Credit Reversal Pair ................................................................................................................... 135

Chapter 5 Payment Gateway Certificate Request and Batch Administration .................................... 138
  Overview ..................................................................................................................................... 138
  Payment Gateway Certificate Request Pair ............................................................................. 139
  Batch Administration Pair ......................................................................................................... 141

Chapter 6 Certificate Management Payload Components ............................................................ 147
  Overview ..................................................................................................................................... 147
  IDData .......................................................................................................................................... 148
  RequestType ............................................................................................................................... 149
  End Entity and CA Types .......................................................................................................... 150
  BrandCRLIdentifier ................................................................................................................. 151
Table of Tables

Table 1: Notation 3
Table 2: Entity Symbols 5
Table 3: Hash - H 6
Table 4: DigestedData - DD 6
Table 5: Linkage - L 7
Table 6: Keyed hash mechanism - HMAC 7
Table 7: Signature Only - SO 8
Table 8: Signed Message - S 8
Table 9: Asymmetric Encryption - E 9
Table 10: Integrity Encryption - EH 9
Table 11: Extra Encryption - EX 10
Table 12: Extra Encryption with Integrity - EXH 11
Table 13: Symmetric Encryption with Provided Key Data - EK 11
Table 14: Simple Encapsulations with Signature - Enc, EncK 12
Table 15: Extra Encapsulation with Signature - EncX 13
Table 16: Encapsulations with External, Encrypted Baggage - EncB, EncBX 14
Table 17: OAEP Block Format 15
Table 18: MessageWrapper 25
Table 19: Error Message 30
Table 20: TransIDs 33
Table 21: RRTags 34
Table 22: PI Variants 35
Table 23: PI 35
Table 24: PIHead 37
Table 25: AuthToken 40
Table 26: InstallRecurData 40
Table 27: AcqCardMsg 43
Table 28: CapToken 44
Table 29: PANData 45
Table 30: PANToken 46
Table 31: BatchStatus 47
Table 32: TransactionDetail 50
Table 33: SaleDetail 53
Table 34: CommercialCardData 58
Table 35: MarketAutoCap 62
Table 36: MarketHotelCap 66
Table 37: MarketTransportCap 69
Table 38: PInitReq 72
Table 39: PInitRes 74
Table 40: PReq 76
Table 41: PReqDualSigned 77
Table 42: PReqUnsigned 79
Table 43: OIData 81
Table 44: PRes 84
Table 45: PResPayload 86
Table 46: InqReq 90
Table 47: InqRes 91
Table 48: AuthReq 92
Table 49: AuthReqPayload 95
Table 50: AuthRes 99
Table 51: AuthResPayload 101
Table 52: AuthRevReq 107
Part I
Formal Protocol Definition

Introduction

Purpose

The protocol definition that appears in Part I is provided as commentary and direction for the ASN.1 code in Part II and the processing instructions in Book 2, Programmer’s Guide.

In the event of discrepancies between this and any other description of the protocol, the ASN.1 in Part II takes precedence.

Preliminary notes

Signed messages contain all certificates and certificate revocation lists (CRLs) necessary for the recipient to verify their signatures. A request message can use thumbprints to indicate certificates that it has previously validated and cached, so that the corresponding response message does not need to include those certificates. CRLs and signature certificates are implicit in signed message types. As defined by PKCS #7, these are contained in the certificates and crls fields of SignedData.

SET includes key-exchange certificates in SignedData blocks. In other words, these are implicit in the protocol.

Software decides which key-exchange certificate to use for encryption to Payment Gateways or CAs based on a special thumbprint, PEThumb or CAEThumb respectively, which is sent in the same response message as the key-exchange certificate.

Organization

Part I includes the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cryptography</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Message Encapsulation</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>Payment Message Components</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>Payment Messages</td>
<td>71</td>
</tr>
<tr>
<td>5</td>
<td>Payment Gateway Certificate Request and Batch Administration</td>
<td>138</td>
</tr>
<tr>
<td>6</td>
<td>Certificate Management Payload Components</td>
<td>147</td>
</tr>
<tr>
<td>7</td>
<td>Certificate Management Messages</td>
<td>157</td>
</tr>
</tbody>
</table>

The following pages describe notation and terminology used throughout Part I.

Continued on next page
Introduction, continued

Format

Following the description of each signature primitive, encryption primitive, message, data structure and so on, the corresponding ASN.1 is provided. The complete ASN.1, including all these excerpts, is included in Part II.

Terminology

The following terms are used in this book.

Opaque
Data this is not defined in this specification; the format and content are specified outside of this specification. Opaque fields are used for information generated by an end entity then passed through various messages for the benefit of that entity.

Linkage
We say that message 1 is linked to message 2 if message 1 contains a hash of message 2. This does not imply that message 2 is linked to message 1.
Notation

Purpose

The remainder of Part I is written in the abstract notation described below.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuple</td>
<td>{A, B, C}</td>
<td>A grouping of zero or more data elements. This notation means “the tuple containing A, B, and C,” which may, themselves, be tuples.</td>
</tr>
<tr>
<td>Component</td>
<td>T = {A, B, C}</td>
<td>A tuple may be given a name as shown or by including the name in the left hand column of a table; the respective components of T are referred to as T.A, T.B, and T.C. Data elements of a nested tuple may be referenced without all of the intermediate tuples provided the reference is unambiguous.</td>
</tr>
<tr>
<td>Ordered concatenation</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Optional</td>
<td>[A]</td>
<td>This notation means that item A is optional.</td>
</tr>
<tr>
<td>Selection</td>
<td>&lt;A, B, C&gt;</td>
<td>This notation means that exactly one of A, B, and C must appear. This is a selection notation.</td>
</tr>
<tr>
<td>Optional selection</td>
<td>[&lt;A, B, C&gt;]</td>
<td>This notation means that the selection is optional; that is, that either nothing or exactly one of A, B, and C may appear.</td>
</tr>
<tr>
<td>Multiple instances</td>
<td>{A +}</td>
<td>This notation means a tuple containing one or more instances of A. (Order may not be significant; refer to the specific description for details.)</td>
</tr>
<tr>
<td></td>
<td>{A *}</td>
<td>This notation means a tuple containing zero or more instances of A.</td>
</tr>
<tr>
<td></td>
<td>{[A] +}</td>
<td>This notation means a tuple containing one or more instances of A in an ordered array where each instance of A is optional (that is, may be null).</td>
</tr>
<tr>
<td>Exclusive-or</td>
<td>A ⊕ B</td>
<td>This symbol denotes a bit-wise exclusive-or (XOR) operation.</td>
</tr>
</tbody>
</table>

Table 1: Notation
Chapter 1
Cryptography

Overview

Introduction
Chapter 1 provides a brief introduction to the cryptography used in SET.

Organization
Chapter 1 includes the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities</td>
<td>5</td>
</tr>
<tr>
<td>Hashing and Hash-based operators</td>
<td>6</td>
</tr>
<tr>
<td>Signature Primitives</td>
<td>8</td>
</tr>
<tr>
<td>Encryption Primitives</td>
<td>9</td>
</tr>
<tr>
<td>Encapsulation Operations</td>
<td>12</td>
</tr>
<tr>
<td>Optimal Asymmetric Encryption Padding (OAEP)</td>
<td>15</td>
</tr>
</tbody>
</table>
Entities

Definition: Entity

An entity is a person or system that can be identified through certificates.

The SET entities are various CAs, Cardholders, Merchants, and Payment Gateways. These entities are denoted by:

- **CA** for the various CAs,
- **C** for Cardholder,
- **M** for Merchant, and
- **P** for Payment Gateway.

Sometimes it is necessary to distinguish between two Payment Gateways; in this case, **P1** and **P2** are used.

Entity symbols

These symbols denote not only an entity, but a tuple containing the entity’s certificate and all certificates in the signature chain up through the root. Since certificates and their signature chains are physical data inputs to encryption and signature operators, entities are included in the argument lists of the cryptographic functions described in Part I, along with other tuples that denote message texts and parameters.

<table>
<thead>
<tr>
<th>$r, s$</th>
<th>In this chapter:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• $r$ represents a receiver, identified through an encryption or key-exchange certificate; and</td>
</tr>
<tr>
<td></td>
<td>• $s$ represents a sender, identified through a signature certificate.</td>
</tr>
</tbody>
</table>

The symbols $r$ and $s$ in this chapter are variables that can stand for any SET entity.

Table 2: Entity Symbols
Hashing and Hash-based operators

Hash

| \( H(t) \) | 160-bit SHA-1 hash of tuple \( t \); collision-free thumbprint of \( t \). Collision freedom means that it is computationally unfeasible to find two different tuples with the same hash, that is, instances of \( t_1 \) and \( t_2 \) such that \( H(t_1) = H(t_2) \). |

Table 3: Hash - \( H \)

DigestedData

| DD(\( t \)) | 160-bit SHA-1 hash of tuple \( t \) embedded within a PKCS DigestedData sequence. |

Table 4: DigestedData - DD

Continued on next page
Hashing and Hash-based operators, continued

Linkage

<table>
<thead>
<tr>
<th>L(t1, t2)</th>
<th>Shorthand for {t1, DD(t2)}, an augmentation of t1 to provide linkage from t1 to t2.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>More precisely, L(t1, t2) contains a linkage to t2 that is concatenated to t1. Anyone possessing t2 or a trusted hash of t2 can verify the linkage. However, someone not possessing t2 or a trusted hash cannot verify the linkage.</td>
</tr>
<tr>
<td></td>
<td>This treatment is not symmetric: It does not link from t2 to t1.</td>
</tr>
</tbody>
</table>

More precisely, L(t1, t2) contains a linkage to t2 that is concatenated to t1. Anyone possessing t2 or a trusted hash of t2 can verify the linkage. However, someone not possessing t2 or a trusted hash cannot verify the linkage.

This treatment is not symmetric: It does not link from t2 to t1.

Table 5: Linkage - L

```
2821 L { T1, T2 } ::= SEQUENCE {
    -- Linkage from t1 to t2
    2822    t1  T1,
    2823    t2  DD { T2 }                                  -- PKCS#7 DigestedData
    2824 }
```

Keyed hash mechanism

<table>
<thead>
<tr>
<th>HMAC(t, k)</th>
<th>A derivation of HMAC-MD5 using the SHA-1 algorithm.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HMAC(t, k) = H((k ⊕ opad)</td>
</tr>
<tr>
<td></td>
<td>where</td>
</tr>
<tr>
<td></td>
<td>• ipad is the byte 0x36 repeated 64 times, and</td>
</tr>
<tr>
<td></td>
<td>• opad is the byte 0x5c repeated 64 times.</td>
</tr>
<tr>
<td></td>
<td>Note, ⊕ denotes XOR.</td>
</tr>
</tbody>
</table>

Table 6: Keyed hash mechanism - HMAC

```
2840 HMAC { ToBeHashed, Key } ::= Digest
2841   (CONSTRAINED BY { -- HMAC keyed digest of -- ToBeHashed,  |
2842     -- using -- Key })
```
Signature Primitives

Signature only

| $SO(s, t)$ | The signature of entity $s$ on tuple $t$, but not including the plaintext of $t$. $SO$ corresponds to a PKCS #7 detached signature. |

Table 7: Signature Only - SO

```
2856 $SO \{ \text{SIGNER, ToBeSigned } \} ::= \text{SignedData} \quad \text{-- Detached content}
2857 \text{(CONSTRAINED BY \{ SIGNER, -- signs -- ToBeSigned \})}
2858 \text{(WITH COMPONENTS \{ ... , contentInfo}
2859 \text{ \text{(WITH COMPONENTS\{)
2860 \text{ \text{... , content ABSENT } \}) \} \} ^
2861 \text{WITH COMPONENTS \{ ... , signerInfos (SIZE(1..2)) \})}
```

Signed message

| $S(s, t)$ | Shorthand for $\{t, SO(s, t)\}$, the tuple of $t$ and its detached signature by entity $s$. Corresponds to PKCS #7 SignedData. |

Table 8: Signed Message - $S$

```
2849 $S \{ \text{SIGNER, ToBeSigned } \} ::= \text{SignedData}
2850 \text{(CONSTRAINED BY \{ SIGNER, -- signs -- ToBeSigned \})}
2851 \text{(WITH COMPONENTS \{ ... , contentInfo}
2852 \text{ \text{(WITH COMPONENTS \{
2853 \text{ \text{... , content PRESENT } \}) \} ^
2854 \text{WITH COMPONENTS \{ ... , signerInfos (SIZE(1..2)) \})}
```
Encryption Primitives

Asymmetric encryption

See page 15 for a description of OAEP (Optimal Asymmetric Encryption Padding).

<table>
<thead>
<tr>
<th>E(r, t)</th>
<th>Asymmetric Encryption to entity r of tuple t, corresponding to the standard PKCS #7 EnvelopedData.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
<td>Action</td>
</tr>
<tr>
<td>1</td>
<td>Encrypt t with a DES key, k.</td>
</tr>
<tr>
<td>2</td>
<td>Insert k in a PKCS #7 envelope for entity r under OAEP.</td>
</tr>
</tbody>
</table>

Table 9: Asymmetric Encryption - E

Integrity encryption

See page 15 for a description of OAEP (Optimal Asymmetric Encryption Padding).

<table>
<thead>
<tr>
<th>EH(r, t)</th>
<th>Integrity Encryption to entity r of tuple t. Like E except that the PKCS #7 envelope contains a hash of t. Used when a signature is not available. Processing software shall rehash t and check for a match against the hash of t in the PKCS #7 envelope.</th>
</tr>
</thead>
</table>

Table 10: Integrity Encryption - EH

Continued on next page
Encryption Primitives, continued

Extra encryption

See page 15 for a description of OAEP (Optimal Asymmetric Encryption Padding).

<table>
<thead>
<tr>
<th>EX(r, t, p)</th>
<th>r is the receiver, and t and p are the parts of a two-part message:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• t is the tuple to be linked to p and subjected to symmetric encryption.</td>
</tr>
<tr>
<td></td>
<td>• p is a parameter subject to “extra” processing.</td>
</tr>
</tbody>
</table>

In SET’s implementation, p must be small because it is put inside the PKCS #7 envelope and there is limited space in the envelope.

The SET implementation is as follows:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Generate a fresh, 20-byte nonce and place inside the appropriate field of p to foil dictionary attacks. In the descriptions of “PANData” through “PANOnly” on pages 20-23, the nonce is called EXNonce. This nonce is a one-time, throw-away value.</td>
</tr>
<tr>
<td>2</td>
<td>Let m = L(t, p), that is, t linked to p.</td>
</tr>
<tr>
<td>3</td>
<td>Encrypt m with a DES key k and let OAEP((k, p)) be the RSA envelope for entity r.</td>
</tr>
</tbody>
</table>

Table 11: Extra Encryption - EX

```
2926 EX { RECPIENT, ToBeEnveloped, Parameter } ::= E {
2927    RECPIENT,
2928    L { ToBeEnveloped, Parameter }
2929 } (CONSTRAINED BY { Parameter -- data is included in the OAEP block -- })
```
Encryption Primitives, continued

Extra encryption with integrity

See page 15 for a description of OAEP (Optimal Asymmetric Encryption Padding).

<table>
<thead>
<tr>
<th>EXH(r, t, p)</th>
<th>Like EX, except that</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• a hash of ( t ) is included in the PKCS #7 envelope and</td>
</tr>
<tr>
<td></td>
<td>• the processing software shall check the hash of ( t ), as with EH.</td>
</tr>
</tbody>
</table>

Table 12: Extra Encryption with Integrity - EXH

2931 EXH \{ RECIPIENT, ToBeEnveloped, Parameter \} ::= EX \{
2932 RECIPIENT,
2933 ToBeEnveloped,
2934 Parameter
2935 \} (CONSTRAINED BY \{ -- H(ToBeEnveloped) included in the OAEP block -- \})

Symmetric encryption with provided key data

See page 15 for a description of OAEP (Optimal Asymmetric Encryption Padding).

| EK(kd, t) | Symmetric encryption with provided key data, \( kd \) (algorithm and key). |

Table 13: Symmetric Encryption with Provided Key Data - EK

2937 EK \{ KeyData, ToBeEnveloped \} ::= EncryptedData
2938 (CONSTRAINED BY \{ ToBeEnveloped, -- encrypted with -- KeyData \})
2939 (WITH COMPONENTS \{ ..., encryptedContentInfo
2940 (WITH COMPONENTS \{ ..., encryptedContent PRESENT\} \})

Version 1.0
Encapsulation Operations

**Simple encapsulations with signature**

- **Enc**: models signed, then encrypted messages.
- **EncK**: models signed messages encrypted with a secret key provided in an earlier message.

<table>
<thead>
<tr>
<th>Enc(s, r, t)</th>
<th>Simple Encapsulation with Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shorthand for $E(r, S(s, t))$. Corresponds to an instance of PKCS #7 SignedData encapsulated in EnvelopedData.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EncK(kd, s, t)</th>
<th>Simple Encapsulation with Signature and Provided Key Data (algorithm and key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$EncK(kd, s, t) = EK(kd, S(s, t))$.</td>
<td></td>
</tr>
</tbody>
</table>

Table 14: Simple Encapsulations with Signature - Enc, EncK

```
2869  Enc { SIGNER, RECIPIENT, T } ::= E {  
2870      RECIPIENT, 
2871      S { SIGNER, T } 
2872 } 
2877  EncK { KeyData, SIGNER, T } ::= EK {  
2878      KeyData, 
2879      S { SIGNER, T } 
2880 } 
```

Continued on next page
Encapsulation Operations, continued

This operator models two-part messages encrypted with the first part of the message in the symmetric encryption slot of EX and the second part of the message in the OAEP (extra) slot of EX.

EncX(s, r, t, p)  

| r is the receiver, and t and p are the components of a two-part message:  
| t is the part subject to symmetric encryption.  
| p is the parameter subject to “extra” processing as described in “Extra encryption” on page 10. p is always in one of the formats defined in “Encoding of DB” on page 19.  

<table>
<thead>
<tr>
<th>p is processed in two distinct ways:</th>
<th>...which require two different formats for p:</th>
<th>...and are indicated as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is incorporated into the OAEP data.</td>
<td>See “Extra encryption” on page 10.</td>
<td>OAEP(p)</td>
</tr>
<tr>
<td>It is included in the SO signature described below.</td>
<td>The SO signature is computed over the DER-encoded form of p.</td>
<td>DER(p)</td>
</tr>
</tbody>
</table>

As described in “Encoding of DB” on page 19, p shall include a fresh random nonce called EXNonce. The purpose of this nonce is to foil dictionary attacks against p via the hash implicitly included in the SO signature.

To produce EncX:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>That is,</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Place t and SO(s, {t, DER(p)}) in the DES-protected portion of the message.</td>
<td>Let the clear text of the message be defined as m = {t, SO(s, {t, DER(p)})}. Encrypt m with a DES key k.</td>
</tr>
<tr>
<td>2</td>
<td>Place OAEP(p) in the RSA-protected portion of the message.</td>
<td>Encrypt OAEP((k, p)) using the public key of entity r to create the RSA envelope.</td>
</tr>
</tbody>
</table>

Table 15: Extra Encapsulation with Signature - EncX

Continued on next page
Encapsulation Operations, continued

Encapsulations with external, encrypted baggage

These avoid double encryption for cases where a message must be linked to a previously encrypted tuple such as a PI or a CapToken.

- **EncB** models signed, encrypted messages with external baggage.
- **EncBX** models signed, EX-encrypted, two-part messages with baggage.

SET does not use unsigned, EX-encrypted, two-part messages with baggage.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Link the baggage to the main message.</td>
</tr>
<tr>
<td>2</td>
<td>Sign and encrypt the linked object.</td>
</tr>
<tr>
<td>3</td>
<td>Append the baggage to the end of the encrypted message.</td>
</tr>
</tbody>
</table>

**EncB(s, r, t, b)**  
*Simple Encapsulation with Signature and Baggage.*  
EncB(s, r, t, b) = {Enc(s, r, L(t, b)), b}

**EncBX(s, r, t, b, p)**  
*Extra Encapsulation with Signature and Baggage.*  
EncBX(s, r, t, b, p) = {EncX(s, r, L(t, b), p), b}

<table>
<thead>
<tr>
<th>Encapsulation with External, Encrypted Baggage - EncB, EncBX</th>
</tr>
</thead>
</table>

2897 EncB { SIGNER, RECIPIENT, T, Baggage } ::= SEQUENCE {
2898    enc Enc { SIGNER, RECIPIENT, L { T, Baggage } },
2899    baggage Baggage
2900 }

2905 EncBX { SIGNER, RECIPIENT, T, Baggage, Parameter } ::= SEQUENCE {
2906    encX EncX { SIGNER, RECIPIENT, L { T, Baggage }, Parameter },
2907    baggage Baggage
2908 }
Optimal Asymmetric Encryption Padding (OAEP)

The E, EH, EX, and EXH encryption primitives combine RSA encryption and Bellare-Rogaway Optimal Asymmetric Encryption Padding (OAEP). The format of the RSA block and the OAEP processing are defined here.

### OAEP block format

The plain text block before RSA encryption. The block consists of a leading byte containing I, followed by a padded data block PDB, as follows:

\[ R = I | PDB \]

The leading I ensures that the encryption block, considered as an integer, is less than the modulus.

- **R**
  - The initial byte is a single, non-zero byte with the high-order bit set to zero. The low-order 7 bits should be a fresh, random, non-zero value.
  - Length: 1

- **PDB**
  - The Padded Data Block, the concatenation of two parts: A and B.
  - Length: 127

- **A**
  - The XOR of the H1 hash of E-Salt and of the ultimate data block to be encrypted, DB:
  \[ A = H1(E-Salt) \oplus DB \]
  - Length: 111

- **H1(t)**
  - The length of H1 is the same as the length of DB, as described later in this table. It is constructed by extracting the leading bytes from the string formed from the following expression:
  \[ H(x | 00) | H(x | 01) | H(x | 02) | \ldots | H(x | 05) \]
  - where
    - H(x | n) is generated as many times as needed to produce the required bytes (in this case, six times),
    - n is a single byte counter (in this case with values from 00 to 05),
    - H is SHA-1, which produces a 20-byte hash, and
    - t is the parameter to H1, that is, E-Salt.
  - Length: 111

- **E-Salt**
  - Fresh, 16-byte random salt.
  - Length: 16

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>The plain text block before RSA encryption. The block consists of a leading byte containing I, followed by a padded data block PDB, as follows: [ R = I</td>
<td>PDB ] The leading I ensures that the encryption block, considered as an integer, is less than the modulus.</td>
</tr>
<tr>
<td>I</td>
<td>The initial byte is a single, non-zero byte with the high-order bit set to zero. The low-order 7 bits should be a fresh, random, non-zero value.</td>
<td>1</td>
</tr>
<tr>
<td>PDB</td>
<td>The Padded Data Block, the concatenation of two parts: A and B. [ PDB = A</td>
<td>B ]</td>
</tr>
<tr>
<td>A</td>
<td>The XOR of the H1 hash of E-Salt and of the ultimate data block to be encrypted, DB: [ A = H1(E-Salt) \oplus DB ]</td>
<td>111</td>
</tr>
</tbody>
</table>
| H1(t) | The length of H1 is the same as the length of DB, as described later in this table. It is constructed by extracting the leading bytes from the string formed from the following expression: \[ H(x | 00) | H(x | 01) | H(x | 02) | \ldots | H(x | 05) \] where
  - H(x | n) is generated as many times as needed to produce the required bytes (in this case, six times),
  - n is a single byte counter (in this case with values from 00 to 05),
  - H is SHA-1, which produces a 20-byte hash, and
  - t is the parameter to H1, that is, E-Salt. | 111 |
| E-Salt | Fresh, 16-byte random salt. | 16 |

Table 17: OAEP Block Format

*Continued on next page*
Optimal Asymmetric Encryption Padding (OAEP), continued

OAEP block format (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
</table>
| DB   | The data block, DB, consists of:  
|      | • the Actual Data Block, ADB, concatenated to  
|      | • a Block Type byte, BT;  
|      | • a Block Contents byte, BC, and  
|      | • the verification string, V:  
|      | $DB = BT | BC | V | ADB$ | 111 |
| BT   | A single byte containing the fixed constant x'03'. The purpose of this byte is to identify the block format. | 1 |
| BC   | Block contents byte, indicating what data is carried in the Actual Data Block, ADB. The high-order bit of this byte is:  
|      | • one if the ADB contains HD (as described later in this table);  
|      | • otherwise, zero.  
|      | The remaining bits indicate the format of any extra data, X (values when the bit indicating the presence of HD is set are given in parentheses):  
|      | 00 (80) no extra data, i.e., X is not present  
|      | 01 (81) PANData (see page 20)  
|      | 02 (82) PANData0 (see page 21)  
|      | 03 (83) PANToken (see page 22)  
|      | 04 (84) PANOnly (see page 23)  
|      | 05 (85) AcctData (see page 23) | 1 |
| V    | A 7-byte string of zeroes. The purpose of V is to verify correct decryption of the RSA block. (Note: the combination of BT and V gives 8 fixed-value bytes that verify correct decryption.) | 7 |

Table 17: OAEP Block Format, continued

Continued on next page
Optimal Asymmetric Encryption Padding (OAEP), continued

### OAEP block format (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADB</strong></td>
<td>The Actual Data Block, containing one or more of the fields DEK, HD, and X (depending on the encryption primitive, as indicated by the block content byte, BC), left-justified:</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td><strong>Value of BC</strong></td>
<td><strong>Encryption primitive</strong></td>
</tr>
<tr>
<td>0 (zero)</td>
<td>E</td>
<td>DEK</td>
</tr>
<tr>
<td>&gt; 0 and &lt; 80 hex</td>
<td>EX</td>
<td>DEK</td>
</tr>
<tr>
<td>80 hex</td>
<td>EH</td>
<td>DEK</td>
</tr>
<tr>
<td>&gt; 81 hex</td>
<td>EXH</td>
<td>DEK</td>
</tr>
<tr>
<td><strong>DEK</strong></td>
<td>An 8-byte DES encryption key stored at the start of ADB. This key is used to encrypt D.</td>
<td>8</td>
</tr>
<tr>
<td><strong>HD</strong></td>
<td>A 20-byte SHA-1 hash of the data prior to encryption: H(D).</td>
<td>20</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>The data that will be symmetrically encrypted under the DES key DEK.</td>
<td>varies</td>
</tr>
<tr>
<td><strong>X</strong></td>
<td>“Extra-encrypted” data contained within the OAEP-processed and RSA-encrypted block. The format of this data is described below under “Encoding of DB” on page 19.</td>
<td>varies</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>B is the XOR of E-Salt with the H2 hash of A: B = E-Salt ⊕ H2(A) B is the same length as E-Salt.</td>
<td>16</td>
</tr>
<tr>
<td><strong>H2(t)</strong></td>
<td>H2 returns the trailing 16 bytes of the SHA-1 hash of its argument, t.</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 17: OAEP Block Format, continued

Continued on next page
## Optimal Asymmetric Encryption Padding (OAEP), continued

<table>
<thead>
<tr>
<th>Field lengths</th>
<th>SET fixes the lengths of the salt (E-Salt), verification field (V), and data block (DB) fields. The length of extra data (X) can be derived from the block contents (BC) byte.</th>
</tr>
</thead>
</table>
| I field       | The SET format is differentiated from existing PKCS #7 block formats by setting the first byte (I) non-zero.  
|               | • Force the high-order bit of I to zero to ensure that the arithmetic value of the block is less than the RSA modulus.  
|               | • Set the remaining 7 bits of I to a fresh, random, non-zero value.                                                                                                                                  |
| RSA modulus   | The PKCS #7 block format requires that the RSA modulus, when expressed as an OCTET STRING, have the first bit set. That is, a 1024-bit modulus must be in the range of $2^{1023}$ to $2^{1024} - 1$. Moduli in this range must necessarily be greater than the arithmetic value of the block (prior to RSA encryption) since the first bit of the block is required to be zero. This avoids ambiguity in the RSA decryption process. |
| BT field      | The Block Type byte is provided to identify the SET block format, and allow future variations.                                                                                                          |

*Continued on next page*
Optimal Asymmetric Encryption Padding (OAEP), continued

Space available for extra encryption

The maximum length of the “extra” data, $X$, is a function of the size of the RSA block, and whether $\text{EncX}$, $\text{EX}$, or $\text{EXH}$ encryption is used. The RSA block is 128 bytes. The following table shows the net amount of space available for extra-encrypted data for each encryption type with this RSA block size.

<table>
<thead>
<tr>
<th>Extra Encryption Type</th>
<th>Space Available with 128 Byte RSA Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>EncX</td>
<td>94 bytes</td>
</tr>
<tr>
<td>EX</td>
<td>94 bytes</td>
</tr>
<tr>
<td>EXH</td>
<td>74 bytes</td>
</tr>
</tbody>
</table>

Encoding of DB

Data present in data block (DB) fields are not formatted with the usual DER encoding method, in order to save space. The format used for the DB is defined here to support interoperability among implementations.

For all of the definitions, all fields shall be present.

Only fields from the ASN.1 definition are present in DB. Each element is encoded within DB in the canonical form used by DER encoding, but without tag and length indicators. When transferring data from DER-encoded format to DB, add pad characters to the end of the data; when transferring from DB to DER-encoded format, strip all pad characters from the end of the data.

To understand the format of a DB field, examine the ASN.1 used to define the field for signature purposes. Determine the corresponding ASN.1 type, and store the field in DB according to the following table, which summarizes the DER format of field types used in SET extra-encrypted data:

<table>
<thead>
<tr>
<th>ASN.1 Type</th>
<th>DB Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>VisibleString</td>
<td>ASCII string, first character in lowest-numbered position, padded with blanks (0x20).</td>
</tr>
<tr>
<td>NumericString</td>
<td>ASCII string, first character in lowest-numbered position, padded with blanks (0x20).</td>
</tr>
<tr>
<td>OCTET STRING</td>
<td>Binary byte string in lowest-numbered position, padded with bytes of zero (0x00).</td>
</tr>
</tbody>
</table>

Continued on next page
Optimal Asymmetric Encryption Padding (OAEP), continued

**DB fields**

The OAEP block contains several standard fields, which are formatted as follows:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>A</td>
<td>111</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>B</td>
<td>16</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>BT</td>
<td>1</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>BC</td>
<td>1</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>V</td>
<td>7</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>DEK</td>
<td>8</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>HD</td>
<td>20</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>X</td>
<td>varies</td>
<td>depends upon content</td>
</tr>
</tbody>
</table>

**PANData**

PANData is carried in the signed form of the purchase request (PReq) message. PANData is 65 bytes and contains four fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN</td>
<td>19</td>
<td>Numeric String</td>
</tr>
<tr>
<td>CardExpiry</td>
<td>6</td>
<td>NumericString - YYYYMM</td>
</tr>
<tr>
<td>PANSecret</td>
<td>20</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>EXNonce</td>
<td>20</td>
<td>OCTET STRING</td>
</tr>
</tbody>
</table>

When a signature is calculated that includes PANData, the following ASN.1 is used.

300 PANData ::= SEQUENCE {
301    pan         PAN,
302    cardExpiry  CardExpiry,
303    panSecret   Secret,
304    exNonce     Nonce
305 }

298 PAN ::= NumericString (SIZE(1..19))

252 CardExpiry ::= NumericString (SIZE(6)) -- YYYYMM expiration date of card

296 Nonce ::= OCTET STRING (SIZE(20))

Continued on next page
Optimal Asymmetric Encryption Padding (OAEP), continued

**PANData0**

PANData0 is carried in the Certificate Request (CertReq) message. It is like PANData, except that CardSecret substitutes for PANSecret. PANData0 is 65 bytes and contains four fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN</td>
<td>19</td>
<td>Numeric String</td>
</tr>
<tr>
<td>CardExpiry</td>
<td>6</td>
<td>NumericString - YYYYMM</td>
</tr>
<tr>
<td>CardSecret</td>
<td>20</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>EXNonce</td>
<td>20</td>
<td>OCTET STRING</td>
</tr>
</tbody>
</table>

When a signature is calculated that includes PANData0, the following ASN.1 is used.

```
307 PANData0 ::= SEQUENCE {
308   pan         PAN,
309   cardExpiry  CardExpiry,
310   cardSecret  Secret,
311   exNonce     Nonce
312 }

298 PAN ::= NumericString (SIZE(1..19))

252 CardExpiry ::= NumericString (SIZE(6)) -- YYYYMM expiration date of card

296 Nonce ::= OCTET STRING (SIZE(20))
```

*Continued on next page*
Optimal Asymmetric Encryption Padding (OAEP), continued

**PANToken**

**PANToken** is carried in the unsigned form of the purchase request (**PReq**) message as well as optionally carried in a number of the messages transmitted between an Payment Gateway and a Merchant. **PANToken** is 45 bytes and contains three fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN</td>
<td>19</td>
<td>Numeric String</td>
</tr>
<tr>
<td>CardExpiry</td>
<td>6</td>
<td>NumericString - YYYYMM</td>
</tr>
<tr>
<td>EXNonce</td>
<td>20</td>
<td>OCTET STRING</td>
</tr>
</tbody>
</table>

When a signature is calculated that includes **PANToken**, the following ASN.1 is used.

314 PANToken ::= SEQUENCE {
315   pan      PAN,
316   cardExpiry CardExpiry,
317   exNonce  Nonce
318 }

298 PAN ::= NumericString (SIZE(1..19))

252 CardExpiry ::= NumericString (SIZE(6)) -- YYYYMM expiration date of card

296 Nonce ::= OCTET STRING (SIZE(20))

*Continued on next page*
Optimal Asymmetric Encryption Padding (OAEP), continued

**PANOnly**

The **PAN** is carried on its own in the **RegFormReq** message.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN</td>
<td>19</td>
<td>Numeric String</td>
</tr>
<tr>
<td>EXNonce</td>
<td>20</td>
<td>OCTET STRING</td>
</tr>
</tbody>
</table>

When a signature is calculated that includes **PANOnly**, the following ASN.1 is used.

```
552 PANOnly ::= SEQUENCE {
    PAN             NumericString (SIZE(1..19)),
    exNonce         OCTET STRING (SIZE(20))
}
```

**AcctData**

**AcctData** contains identification information about a Merchant or a Payment Gateway in a **CertReq** message. **AcctData** contains two fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcctIdentification</td>
<td>74</td>
<td>VisibleString</td>
</tr>
<tr>
<td>EXNonce</td>
<td>20</td>
<td>OCTET STRING</td>
</tr>
</tbody>
</table>

When a signature is calculated that includes **AcctData**, the following ASN.1 is used.

```
397 AcctData ::= SEQUENCE {
    AcctIdentification AcctIdentification,  
    exNonce             OCTET STRING (SIZE(20))
}
```

Version 1.0
Chapter 2
Message Encapsulation

Organization

This chapter describes:

• MessageWrapper
• Error Message
MessageWrapper

<table>
<thead>
<tr>
<th>MessageWrapper</th>
<th>{MessageHeader, Message, [MWExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageHeader</td>
<td>{Version, Revision, Date, [MessageIDs], [RRPID], SWIdent}</td>
</tr>
<tr>
<td>MWExtensions</td>
<td>Appropriate where:</td>
</tr>
<tr>
<td></td>
<td>• the data in the extension is general purpose information about SET messages, or</td>
</tr>
<tr>
<td></td>
<td>• the contents of the message are encrypted and the extension contains non-financial data that does not require confidentiality.</td>
</tr>
<tr>
<td></td>
<td>Note: The message wrapper is not encrypted so this extension must not contain confidential information.</td>
</tr>
</tbody>
</table>

Table 18: MessageWrapper

Continued on next page
**MessageWrapper**, continued

---

**MessageWrapper (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Version of SET message</td>
</tr>
<tr>
<td>Revision</td>
<td>Minor revision of SET message</td>
</tr>
<tr>
<td>Date</td>
<td>Date and time of message generation</td>
</tr>
<tr>
<td>MessageIDs</td>
<td>{[LID-C], [LID-M], [XID]}</td>
</tr>
<tr>
<td>RRPID</td>
<td>Request/response pair ID for this cycle</td>
</tr>
<tr>
<td>SWIdent</td>
<td>String identifying the software (vendor and version) initiating the request.</td>
</tr>
<tr>
<td>LID-C</td>
<td>Local ID; convenience label generated by and for Cardholder system</td>
</tr>
<tr>
<td>LID-M</td>
<td>Local ID; convenience label generated by and for Merchant system</td>
</tr>
<tr>
<td>XID</td>
<td>Globally unique ID generated by Merchant in <strong>PInitRes</strong> or by Cardholder in <strong>PReq</strong></td>
</tr>
</tbody>
</table>

Table 18: **MessageWrapper**, continued

---

*Continued on next page*
MessageWrapper, continued

MessageWrapper (continued)

43 MessageWrapper ::= SEQUENCE {
44   messageHeader  MessageHeader,
45   message        [0] EXPLICIT MESSAGE.&Type (Message),
46   mwExtensions   [1] MsgExtensions {{MWExtensionsIOS}} OPTIONAL
47 }

58 MessageHeader ::= SEQUENCE {
59   version     INTEGER { setVer1(1) } (setVer1),
60   revision    INTEGER (0) DEFAULT 0,   -- This is version 1.0
61   date        Date,
62   messageIDs  [0] MessageIDs OPTIONAL,
63   rrpid       [1] RRID OPTIONAL,
64   swIdent     SWIdent
65 }

Continued on next page
MessageWrapper, continued

MessageWrapper (continued)

75 Message ::= CHOICE {
76   purchaseInitRequest [ 0] EXPLICIT PInitReq,
77   purchaseInitResponse [ 1] EXPLICIT PInitRes,
78   purchaseRequest [ 2] EXPLICIT PReq,
79   purchaseResponse [ 3] EXPLICIT PRes,
80   inquiryRequest [ 4] EXPLICIT InqReq,
81   inquiryResponse [ 5] EXPLICIT InqRes,
82   authorizationRequest [ 6] EXPLICIT AuthReq,
83   authorizationResponse [ 7] EXPLICIT AuthRes,
84   authReversalRequest [ 8] EXPLICIT AuthRevReq,
85   authReversalResponse [ 9] EXPLICIT AuthRevRes,
86   captureRequest [10] EXPLICIT CapReq,
88   captureReversalRequest [12] EXPLICIT CapRevReq,
90   creditRequest [14] EXPLICIT CredReq,
91   creditResponse [15] EXPLICIT CredRes,
92   creditReversalRequest [16] EXPLICIT CredRevReq,
93   creditReversalResponse [17] EXPLICIT CredRevRes,
94   pCertificateRequest [18] EXPLICIT PCertReq,
95   pCertificateResponse [19] EXPLICIT PCertRes,
96   batchAdministrationRequest [20] EXPLICIT BatchAdminReq,
97   batchAdministrationResponse [21] EXPLICIT BatchAdminRes,
98   cardholderCInitRequest [22] EXPLICIT CardCInitReq,
99   cardholderCInitResponse [23] EXPLICIT CardCInitRes,
100  meAqCInitRequest [24] EXPLICIT Me-AqCInitReq,
101  meAqCInitResponse [25] EXPLICIT Me-AqCInitRes,
102  registrationFormRequest [26] EXPLICIT RegFormReq,
103  registrationFormResponse [27] EXPLICIT RegFormRes,
104  certificateRequest [28] EXPLICIT CertReq,
105  certificateResponse [29] EXPLICIT CertRes,
106  certificateInquiryRequest [30] EXPLICIT CertInqReq,
107  certificateInquiryResponse [31] EXPLICIT CertInqRes,
108  error [999] EXPLICIT Error
126 }

Continued on next page
MessageWrapper, continued

MessageWrapper (continued)

265 Date ::= GeneralizedTime

67 MessageIDs ::= SEQUENCE {
68    lid-C [0] LocalID OPTIONAL,
69    lid-M [1] LocalID OPTIONAL,
70    xID [2] XID OPTIONAL
71 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

328 SWIdent ::= VisibleString (SIZE(1..ub-SWIdent)) -- Software identification

348 XID ::= OCTET STRING (SIZE(20))
Error Message

Error message

<table>
<thead>
<tr>
<th>Error</th>
<th>&lt; SignedError, UnsignedError &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignedError</td>
<td>S(EE, ErrorTBS)</td>
</tr>
<tr>
<td>UnsignedError</td>
<td>ErrorTBS</td>
</tr>
<tr>
<td>ErrorTBS</td>
<td>{ErrorCode, ErrorNonce, [ErrorOID], [ErrorThumb],ErrorMsg}</td>
</tr>
</tbody>
</table>

The unsigned version of **Error** shall only be used if the entity does not have a valid signature certificate or is temporarily unable to generate signatures (such as when there is a cryptographic hardware failure).

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>Enumerated code.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorNonce</td>
<td>A nonce to ensure the signature is generated over unpredictable data.</td>
</tr>
<tr>
<td>ErrorOID</td>
<td>The object identifier of an object (extension, content type, attribute, etc.) that caused the error.</td>
</tr>
<tr>
<td>ErrorThumb</td>
<td>The thumbprint of the certificate, CRL or BrandCRLIdentifier that caused the error.</td>
</tr>
<tr>
<td>ErrorMsg</td>
<td>&lt;MessageHeader, BadWrapper&gt;</td>
</tr>
<tr>
<td>MessageHeader</td>
<td>The message header of the message that produced the error.</td>
</tr>
<tr>
<td>BadWrapper</td>
<td>The message wrapper of the message that produced the error (up to 20,000 bytes).</td>
</tr>
</tbody>
</table>

Table 19: Error Message

```
144 Error ::= CHOICE {
  145   signedError [0] EXPLICIT SignedError,
  146   unsignedError [1] EXPLICIT ErrorTBS
147 }

149 SignedError ::= S {EE, ErrorTBS}

151 ErrorTBS ::= SEQUENCE {
  152   errorCode    ErrorCode,
  153   errorNonce   Nonce,
  154   errorOID     [0] OBJECT IDENTIFIER OPTIONAL,
  155   errorThumb   [1] EXPLICIT CertThumb OPTIONAL,
  156   errorMsg     [2] EXPLICIT ErrorMsg
157 }
```

Continued on next page
Error Message, continued

Error message (continued)

164 ErrorCode ::= ENUMERATED {
    unspecifiedFailure       (1),
    messageNotSupported      (2),
    decodingFailure          (3),
    invalidCertificate       (4),
    expiredCertificate       (5),
    revokedCertificate       (6),
    missingCertificate       (7),
    signatureFailure         (8),
    badMessageHeader         (9),
    wrapperMsgMismatch       (10),
    versionTooOld            (11),
    versionTooNew            (12),
    unrecognizedExtension    (13),
    messageTooBig            (14),
    signatureRequired        (15),
    messageTooOld            (16),
    messageTooNew            (17),
    thumbsMismatch           (18),
    unknownRRPID             (19),
    unknownXID               (20),
    unknownLID               (21),
    challengeMismatch        (22)
}

159 ErrorMsg ::= CHOICE {
    messageHeader  [0] EXPLICIT MessageHeader, -- Either the
    badWrapper     [1] OCTET STRING (SIZE(1..20000)) -- MessageHeader or a

58 MessageHeader ::= SEQUENCE {
    version     INTEGER { setVer1(1) } (setVer1),
    revision    INTEGER (0) DEFAULT 0, -- This is version 1.0
    date        Date,
    messageIDs  [0] MessageIDs OPTIONAL,
    rrpid       [1] RRPID OPTIONAL,
    swIdent     SWIdent
}

59 version INTEGER { setVer1(1) } (setVer1),
60 revision INTEGER (0) DEFAULT 0, -- This is version 1.0
61 date Date,
62 messageIDs [0] MessageIDs OPTIONAL,
63 rrpid [1] RRPID OPTIONAL,
64 swIdent SWIdent
65 }
Chapter 3
Payment Message Components

Overview

Introduction
Chapter 3 defines the protocol components TransIDs and RRTags, plus various payload components included in payment messages that are described in Chapter 4.

Notes
1. Comments are in italics.
2. Sub-definitions appear in depth-first order following first use.

Organization
This chapter includes the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Components</td>
<td></td>
</tr>
<tr>
<td>TransIDs</td>
<td>33</td>
</tr>
<tr>
<td>RRTags</td>
<td>34</td>
</tr>
<tr>
<td>Payload Components</td>
<td></td>
</tr>
<tr>
<td>PI (Payment Instruction)</td>
<td>35</td>
</tr>
<tr>
<td>PIHead</td>
<td>37</td>
</tr>
<tr>
<td>AuthToken</td>
<td>40</td>
</tr>
<tr>
<td>InstallRecurData</td>
<td>42</td>
</tr>
<tr>
<td>AcqCardMsg</td>
<td>43</td>
</tr>
<tr>
<td>CapToken</td>
<td>44</td>
</tr>
<tr>
<td>PANData</td>
<td>45</td>
</tr>
<tr>
<td>PANToken</td>
<td>46</td>
</tr>
<tr>
<td>BatchStatus</td>
<td>47</td>
</tr>
<tr>
<td>TransactionDetail</td>
<td>50</td>
</tr>
<tr>
<td>Location</td>
<td>52</td>
</tr>
<tr>
<td>SaleDetail</td>
<td>53</td>
</tr>
<tr>
<td>CommercialCardData</td>
<td>58</td>
</tr>
<tr>
<td>MarketAutoCap</td>
<td>62</td>
</tr>
<tr>
<td>MarketHotelCap</td>
<td>66</td>
</tr>
<tr>
<td>MarketTransportCap</td>
<td>69</td>
</tr>
</tbody>
</table>
TransIDs

<table>
<thead>
<tr>
<th>TransIDs</th>
<th>{LID-C, [LID-M], XID, PReqDate, [PaySysID], Language}</th>
</tr>
</thead>
<tbody>
<tr>
<td>LID-C</td>
<td>Local ID; convenience label generated by and for Cardholder system. This field has the same value as in the MessageWrapper; see page 25.</td>
</tr>
<tr>
<td>LID-M</td>
<td>Local ID; convenience label generated by and for Merchant system. This field has the same value as in the MessageWrapper; see page 25.</td>
</tr>
<tr>
<td>XID</td>
<td>Globally unique ID. This field has the same value as in the MessageWrapper; see page 25.</td>
</tr>
<tr>
<td>PReqDate</td>
<td>Date of purchase request; generated by Merchant in \texttt{PlnitRes} or by Cardholder in \texttt{PReq}.</td>
</tr>
<tr>
<td>PaySysID</td>
<td>Used by some payment card brands to label transaction from time of authorization onward</td>
</tr>
<tr>
<td>Language</td>
<td>Cardholder’s natural language</td>
</tr>
</tbody>
</table>

Table 20: TransIDs

337 TransIDs ::= SEQUENCE {
338   lid-C LocalID,
339   lid-M [0] LocalID OPTIONAL,
340   xid XID,
341   pReqDate Date,
342   paySysID [1] PaySysID OPTIONAL,
343   language Language -- Cardholder requested session language
344 }

348 XID ::= OCTET STRING (SIZE(20))

320 PaySysID ::= VisibleString (SIZE(1..ub-paySysID))

282 Language ::= VisibleString (SIZE(1..ub-RFC1766-language))
### RRTags

<table>
<thead>
<tr>
<th>RRTags</th>
<th>{RRPID, MerTermIDs, Date}</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRPID</td>
<td>Fresh request/response pair ID</td>
</tr>
<tr>
<td>MerTermIDs</td>
<td>{MerchantID, [TerminalID], [AgentNum], [ChainNum], [StoreNum]}</td>
</tr>
<tr>
<td>Date</td>
<td>Current date for aging logs</td>
</tr>
<tr>
<td>MerchantID</td>
<td>Cardholder inserts this data in PIHead. It is copied from MerID in the Merchant signature certificate.</td>
</tr>
<tr>
<td>TerminalID</td>
<td>Merchant inserts this data in AuthReq</td>
</tr>
<tr>
<td>AgentNum</td>
<td>Merchant inserts this data in AuthReq</td>
</tr>
<tr>
<td>ChainNum</td>
<td>Merchant inserts this data in AuthReq</td>
</tr>
<tr>
<td>StoreNum</td>
<td>Merchant inserts this data in AuthReq</td>
</tr>
</tbody>
</table>

#### Table 21: RRTags

```
1914  RRTags ::= SEQUENCE {
1915    rrpid        RRPID,
1916    merTermIDs  MerTermIDs,
1917    currentDate  Date
1918  }

324  RRPID ::= OCTET STRING(SIZE(20))  -- Request response pair identification

1906 MerTermIDs ::= SEQUENCE {
1907    merchantID  MerchantID,
1908    terminalID  VisibleString (SIZE(1..ub-terminalID)) OPTIONAL,
1909    agentNum    INTEGER (0..MAX) OPTIONAL,
1910    chainNum    [0] INTEGER (0..MAX) OPTIONAL,
1911    storeNum    [1] INTEGER (0..MAX) OPTIONAL
1912  }

265  Date ::= GeneralizedTime

294  MerchantID ::= SETString { ub-MerchantID }
```
PI (Payment Instruction)

There are three alternative forms of PI:

<table>
<thead>
<tr>
<th>PI form</th>
<th>Created by...</th>
<th>For...</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIUnsigned</td>
<td>Cardholder</td>
<td>Sending PReqUnsigned message</td>
</tr>
<tr>
<td>PIDualSigned</td>
<td>Cardholder</td>
<td>Sending PReqDualSigned message</td>
</tr>
<tr>
<td>AuthToken</td>
<td>Payment Gateway</td>
<td>Merchants to redeem in subsequent AuthReqs</td>
</tr>
</tbody>
</table>

Table 22: PI Variants

<table>
<thead>
<tr>
<th>PI</th>
<th>&lt; PIUnsigned, PIDualSigned, AuthToken &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cardholder creates PIUnsigned or PIDualSigned.</td>
</tr>
<tr>
<td></td>
<td>Payment gateway creates AuthToken to support split shipments or installment/recurring payments.</td>
</tr>
<tr>
<td></td>
<td>Merchant shall retain the PI for later incorporation into AuthReq.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PI Unsigned</th>
<th>EXH(P, PI-OILink, PANToken)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See page 46 for PANToken.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PI DualSigned</th>
<th>{PISignature, EX(P, PI-OILink, PANData)}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See page 45 for PANData.</td>
</tr>
</tbody>
</table>

| AuthToken      | See page 40.                           |

<table>
<thead>
<tr>
<th>PI-OILink</th>
<th>L(PIHead, OIdata)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See page 37 for PIHead. See page 81 for OIdata.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PI Signature</th>
<th>SO(C, PI-TBS)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PI-TBS</th>
<th>{HPIData, HOIData}</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>HPIData</th>
<th>DD(PIData)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>HOIData</th>
<th>DD(OIData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See page 81 for OIdata.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PIData</th>
<th>{PIHead, PANData}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See page 37 for PIHead.</td>
</tr>
<tr>
<td></td>
<td>See page 45 for PANData.</td>
</tr>
</tbody>
</table>

Table 23: PI

Continued on next page
PI (Payment Instruction), continued

PI (Payment Instruction) (continued)

822 PI ::= CHOICE {
823     piUnsigned         [0] EXPLICIT PIUnsigned,
824     piDualSigned       [1] EXPLICIT PIDualSigned,
825     authToken          [2] EXPLICIT AuthToken
826 }

898 PIUnsigned ::= EXH { P, PI-OILink, PANToken }

799 PIDualSigned ::= SEQUENCE {
800     piSignature        PISignature,
801     exPIData          EX { P, PI-OILink, PANData }
802 }

1787 AuthToken ::= EncX { P1, P2, AuthTokenData, PANToken }

807 PI-OILink ::= L { PIHead, OIData }

811 PISignature ::= SO { C, PI-TBS }

813 PI-TBS ::= SEQUENCE {
814     hPIData            HPIData,
815     hOIData           HOIData
816 }

818 HPIData ::= DD { PIData } -- PKCS#7 DigestedData
820 HOIData ::= DD { OIData } -- PKCS#7 DigestedData

828 PIData ::= SEQUENCE {
829     piHead             PIHead,
830     panData           PANData
831 }

Continued on next page
**PI (Payment Instruction), continued**

**PIHead**

<table>
<thead>
<tr>
<th>PIHead</th>
<th>{TransIDs, Inputs, MerchantID, [InstallRecurData], TransStain, SWIdent, [AcqBackKeyData], [PIExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransIDs</td>
<td>See page 33.</td>
</tr>
<tr>
<td>Inputs</td>
<td>{HOD, PurchAmt}</td>
</tr>
<tr>
<td>MerchantID</td>
<td>Copied from Merchant signature certificate</td>
</tr>
<tr>
<td>InstallRecurData</td>
<td>See page 42.</td>
</tr>
<tr>
<td>TransStain</td>
<td>HMAC(XID, CardSecret)</td>
</tr>
<tr>
<td>SWIdent</td>
<td>String identifying the software (vendor and version) initiating the request. It is specified in the PI so the Payment Gateway knows the software of the Cardholder.</td>
</tr>
<tr>
<td>AcqBackKeyData</td>
<td>{AcqBackAlg, AcqBackKey}</td>
</tr>
<tr>
<td>PIExtensions</td>
<td>The data in an extension to the payment instructions must be financial and should be important for the processing of an authorization by the Payment Gateway, the financial network, or the issuer.</td>
</tr>
</tbody>
</table>

*Table 24: PIHead*
**PI (Payment Instruction), continued**

---

**PIHead (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOD</td>
<td>The same value as placed in <strong>OIData</strong>. See “<strong>OIData</strong>” on page 81</td>
</tr>
<tr>
<td>PurchAmt</td>
<td>The amount of the transaction as specified by the Cardholder</td>
</tr>
<tr>
<td>XID</td>
<td>Copied from <strong>TransIDs</strong>; see page 33</td>
</tr>
<tr>
<td>CardSecret</td>
<td>See “<strong>PANData0</strong>” on page 21.</td>
</tr>
<tr>
<td>AcqBackAlg</td>
<td>Selected from Encryption IDs in Payment Gateway certificate.</td>
</tr>
<tr>
<td>AcqBackKey</td>
<td>Key for <strong>AcqCardMsg</strong> of an appropriate length for <strong>AcqBackAlg</strong></td>
</tr>
</tbody>
</table>

---

**Table 24: PIHead, continued**

833 **PIHead** ::= SEQUENCE {
834  transIDs TransIDs,
835  inputs Inputs,
836  merchantID MerchantID,
837  installRecurData [0] InstallRecurData OPTIONAL,
838  transStain TransStain,
839  swIdent SWIdent,
840  acqBackKeyData [1] EXPLICIT BackKeyData OPTIONAL,
842 }

337 **TransIDs** ::= SEQUENCE {
338  lid-C LocalID,
339  lid-M [0] LocalID OPTIONAL,
340  xid XID,
341  pReqDate Date,
342  paySysID [1] PaySysID OPTIONAL,
343  language Language -- Cardholder requested session language
344 }

---

*Continued on next page*
PI (Payment Instruction), continued

PIHead (continued)

846 Inputs ::= SEQUENCE {
847    hod       HOD,
848    purchAmt  CurrencyAmount
849 }

294 MerchantID ::= SETString { ub-MerchantID }

1945 InstallRecurData ::= SEQUENCE {
1946    installRecurInd  InstallRecurInd,
1947    irExtensions     [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL
1948 }

851 TransStain ::= HMAC { XID, Secret }

328 SWIdent ::= VisibleString (SIZE(1..ub-SWIdent))  -- Software identification

870 HOD ::= DD { HODInput }

348 XID ::= OCTET STRING (SIZE(20))

1102 AcqBackKey ::= BackKeyData

Continued on next page
PI (Payment Instruction), continued

AuthToken

Sent by Payment Gateway to Merchant as a proxy for the Cardholder PI for use in subsequent authorizations that occur as a result of split shipments or installment/recurring payments.

<table>
<thead>
<tr>
<th>AuthToken</th>
<th>EncX(P1, P2, AuthTokenData, PANToken)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthTokenData</td>
<td>{TransIDs, PurchAmt, MerchantID, [AcqBackKeyData], [InstallRecurData], [RecurringCount], PrevAuthDateTime, TotalAuthAmount, AuthTokenOpaque}</td>
</tr>
<tr>
<td>PANToken</td>
<td></td>
</tr>
<tr>
<td>TransIDs</td>
<td></td>
</tr>
<tr>
<td>PurchAmt</td>
<td>Fields copied from Cardholder-produced PIHead. See page 37.</td>
</tr>
<tr>
<td>MerchantID</td>
<td></td>
</tr>
<tr>
<td>AcqBackKeyData</td>
<td></td>
</tr>
<tr>
<td>InstallRecurData</td>
<td></td>
</tr>
<tr>
<td>RecurringCount</td>
<td>Number of recurring Authorizations performed so far</td>
</tr>
<tr>
<td>PrevAuthDateTime</td>
<td>Date and time of Merchant’s last Authorization in a sequence of recurring Authorizations</td>
</tr>
<tr>
<td>TotalAuthAmount</td>
<td>The total amount authorized so far by all Authorizations for this XID</td>
</tr>
<tr>
<td>AuthTokenOpaque</td>
<td>Opaque data defined by the generating Payment Gateway</td>
</tr>
</tbody>
</table>

Table 25: AuthToken

1787 AuthToken ::= EncX { P1, P2, AuthTokenData, PANToken }
1800 AuthTokenData ::= SEQUENCE {
    transIDs          TransIDs,  
    purchAmt          CurrencyAmount,  
    merchantID        MerchantID,  
    acqBackKeyData    BackKeyData OPTIONAL,  
    installRecurData  [0] InstallRecurData OPTIONAL,  
    recurringCount    [1] INTEGER (1..MAX) OPTIONAL,  
    prevAuthDateTime  Date,  
    totalAuthAmount   [2] CurrencyAmount OPTIONAL,  
    authTokenOpaque   [3] EXPLICIT TokenOpaque OPTIONAL  
}

Continued on next page
PI (Payment Instruction), continued

AuthToken (continued)

314 PANToken ::= SEQUENCE {
315     pan      PAN,
316     cardExpiry CardExpiry,
317     exNonce   Nonce
318 }

337 TransIDs ::= SEQUENCE {
338     lid-C   LocalID,
339     lid-M   [0] LocalID OPTIONAL,
340     xid     XID,
341     pReqDate Date,
342     paySysID [1] PaySysID OPTIONAL,
343     language Language -- Cardholder requested session language
344 }

294 MerchantID ::= SETString { ub-MerchantID }

1945 InstallRecurData ::= SEQUENCE {
1946     installRecurInd InstallRecurInd,
1947     irExtensions  [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL
1948 }
InstallRecurData

Specifies information about installment or recurring payments.

<table>
<thead>
<tr>
<th>InstallRecurData</th>
<th>{InstallRecurInd, [IRExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstallRecurInd</td>
<td>&lt; InstallTotalTrans, Recurring &gt;</td>
</tr>
<tr>
<td>IRExtensions</td>
<td>The data in an extension to installment or recurring data must be financial and should relate to the processing of subsequent authorizations by the Merchant and the Payment Gateway. Note: The installment/recurring data is not transmitted to the issuer.</td>
</tr>
<tr>
<td>InstallTotalTrans</td>
<td>Cardholder specifies a maximum number of permitted Authorizations for installment payments.</td>
</tr>
<tr>
<td>Recurring</td>
<td>{RecurringFrequency, RecurringExpiry}</td>
</tr>
<tr>
<td>RecurringFrequency</td>
<td>The minimum number of days between Authorizations (a frequency of monthly is indicated by a value of 28), and...</td>
</tr>
<tr>
<td>RecurringExpiry</td>
<td>a final date, after which no further Authorizations are permitted.</td>
</tr>
</tbody>
</table>

Table 26: InstallRecurData

1945 installRecurData ::= SEQUENCE {
1946    installRecurInd  InstallRecurInd,
1947    irExtensions      [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL
1948 }

1952 installRecurInd ::= CHOICE {
1953    installTotalTrans  [0] INTEGER (2..MAX),
1954    recurring          [1] Recurring
1955 }

1957 Recurring ::= SEQUENCE {
1958    recurringFrequency INTEGER (1..ub-recurringFrequency),
1959    recurringExpiry   Date
1960 }
AcqCardMsg

This is tunneled from the Payment Gateway to the Cardholder through the Merchant. The Cardholder sends the symmetric key needed to decrypt it to the Payment Gateway in the **PI**. The Merchant receives it in **AuthRes** and is required to copy it to any subsequent **PRes** and **InqRes** messages generated.

<table>
<thead>
<tr>
<th>AcqCardMsg</th>
<th>EncK(AcqBackKeyData, P, AcqCardCodeMsg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcqBackKeyData</td>
<td><em>is supplied by the Cardholder in the PI. The encrypted message is destined to the Cardholder.</em></td>
</tr>
<tr>
<td>AcqCardCodeMsg</td>
<td>{AcqCardCode, AcqCardMsgData}</td>
</tr>
<tr>
<td>AcqCardCode</td>
<td><em>Enumerated code</em></td>
</tr>
<tr>
<td>AcqCardMsgData</td>
<td><em>Copied from PIHead.AcqBackKeyData; see page 37.</em></td>
</tr>
<tr>
<td>AcqCardText</td>
<td><em>Textual message to be displayed to Cardholder</em></td>
</tr>
<tr>
<td>AcqCardURL</td>
<td><em>URL referencing HTML message to be displayed to Cardholder</em></td>
</tr>
<tr>
<td>AcqCardPhone</td>
<td><em>Phone number to be presented to the Cardholder</em></td>
</tr>
</tbody>
</table>

| Table 27: AcqCardMsg |

1104 AcqCardMsg ::= EncK { AcqBackKey, P, AcqCardCodeMsg }

1109 AcqCardCodeMsg ::= SEQUENCE {
1110    acqCardCode AcqCardCode,
1111    acqCardMsgData AcqCardMsgData
1112 }

1114 AcqCardCode ::= ENUMERATED {
1115    messageOfDay (0),
1116    accountInfo (1),
1117    callCustomerService (2)
1118 }

1120 AcqCardMsgData ::= SEQUENCE {
1121    acqCardText [0] EXPLICIT SETString { ub-acqCardText } OPTIONAL,
1122    acqCardURL [1] URL OPTIONAL,
1123    acqCardPhone [2] EXPLICIT SETString { ub-acqCardPhone } OPTIONAL
1124 }

Version 1.0
CapToken

CapToken

Included in payment messages for the use of the payment gateway; inclusion in responses is at the option of the payment gateway.

<table>
<thead>
<tr>
<th>CapToken</th>
<th>&lt; Enc(P1, P2, CapTokenData), EncX(P1, P2, CapTokenData, PANToken )</th>
<th>P1 and P2 denote Payment Gateways: P1 is the sender. P2 is the receiver. In this version of SET, P1 and P2 are always the same Payment Gateway.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapTokenData</td>
<td>{AuthRRPID, AuthAmt, TokenOpaque}</td>
<td></td>
</tr>
<tr>
<td>PANToken</td>
<td>See page 46.</td>
<td></td>
</tr>
<tr>
<td>AuthRRPID</td>
<td>The RRPID that appeared in the corresponding AuthReq or AuthRevReq</td>
<td></td>
</tr>
<tr>
<td>AuthAmt</td>
<td>Actual amount authorized, which may differ from Cardholder's PurchAmt</td>
<td></td>
</tr>
<tr>
<td>TokenOpaque</td>
<td>Opaque data defined by the generating Payment Gateway</td>
<td></td>
</tr>
</tbody>
</table>

Table 28: CapToken

1816 CapToken ::= CHOICE {
  1817   encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
  1818   enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
  1819   null  [2] EXPLICIT NULL
1820 }

1835 CapTokenData ::= SEQUENCE { authRRPID RRPID,
  1836   authAmt CurrencyAmount,
  1837   tokenOpaque TokenOpaque
1839 }

314 PANToken ::= SEQUENCE { pan PAN,
  315   cardExpiry CardExpiry,
  316   exNonce Nonce
318 }

1962 TokenOpaque ::= TYPE-IDENTIFIER.&Type -- Gateway-defined data
### PANData

**PANData**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always in the extra (OAEP) slot of an encapsulation operator</td>
<td></td>
</tr>
<tr>
<td>Primary Account Number; typically, the account number on the card</td>
<td></td>
</tr>
<tr>
<td>Expiration date on the card</td>
<td></td>
</tr>
<tr>
<td>Secret value shared among Cardholder, Payment Gateway, and Cardholder CA: prevents guessing attacks on PAN in the Cardholder certificate.</td>
<td></td>
</tr>
<tr>
<td>A fresh nonce to foil dictionary attacks on PANData</td>
<td></td>
</tr>
</tbody>
</table>

**Table 29: PANData**

```plaintext
300 PANData ::= SEQUENCE {
301    pan         PAN,
302    cardExpiry  CardExpiry,
303    panSecret   Secret,
304    exNonce     Nonce
305 }

298 PAN ::= NumericString (SIZE(1..19))

252 CardExpiry ::= NumericString (SIZE(6)) -- YYYYMM expiration date of card

296 Nonce ::= OCTET STRING (SIZE(20))
```
PANToken

<table>
<thead>
<tr>
<th>PANToken</th>
<th>{PAN, CardExpiry, EXNonce}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always in the extra (OAEP) slot of an encapsulation operator</td>
<td></td>
</tr>
<tr>
<td>PAN</td>
<td>Primary Account Number; typically, the account number on the card</td>
</tr>
<tr>
<td>CardExpiry</td>
<td>Expiration date on the card</td>
</tr>
<tr>
<td>EXNonce</td>
<td>A fresh nonce to foil dictionary attacks on PANToken</td>
</tr>
</tbody>
</table>

Table 30: PANToken

```
314 PANToken ::= SEQUENCE {
315    pan           PAN,
316    cardExpiry    CardExpiry,
317    exNonce       Nonce
318 }

298 PAN ::= NumericString (SIZE(1..19))

252 CardExpiry ::= NumericString (SIZE(6)) -- YYYYMM expiration date of card
```
### BatchStatus

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenDateTime</strong></td>
<td><em>The date and time the batch was opened</em></td>
</tr>
<tr>
<td><strong>ClosedWhen</strong></td>
<td>{CloseStatus, CloseDateTime}</td>
</tr>
<tr>
<td><strong>BatchDetails</strong></td>
<td>{BatchTotals, [BrandBatchDetailsSeq]}</td>
</tr>
<tr>
<td><strong>BatchExtensions</strong></td>
<td><em>The data in an extension to the batch administration message must be financial and should be important for the processing of the batch administration request.</em></td>
</tr>
<tr>
<td><strong>CloseStatus</strong></td>
<td><em>Enumerated code indicating status of batch close</em></td>
</tr>
<tr>
<td><strong>CloseDateTime</strong></td>
<td><em>The date and time the batch was closed</em></td>
</tr>
<tr>
<td><strong>BatchTotals</strong></td>
<td>{TransactionCountCredit, TransactionTotalAmtCredit, TransactionCountDebit, TransactionTotalAmtDebit, [BatchTotalExtensions]}</td>
</tr>
<tr>
<td><strong>BrandBatchDetailsSeq</strong></td>
<td><em>{BrandBatchDetails +}</em></td>
</tr>
<tr>
<td><strong>TransactionCountCredit</strong></td>
<td><em>The number of transactions that resulted in a credit to the Merchant's account</em></td>
</tr>
<tr>
<td><strong>TransactionTotalAmtCredit</strong></td>
<td><em>The total amount credited to the Merchant's account</em></td>
</tr>
<tr>
<td><strong>TransactionCountDebit</strong></td>
<td><em>The number of transactions that resulted in a debit to the Merchant's account</em></td>
</tr>
<tr>
<td><strong>TransactionTotalAmtDebit</strong></td>
<td><em>The total amount debited from the Merchant's account</em></td>
</tr>
</tbody>
</table>

Table 31: BatchStatus

*Continued on next page*
**BatchStatus**, continued

**BatchStatus** (continued)

<table>
<thead>
<tr>
<th>BatchTotalExtensions</th>
<th>The data in an extension to the batch administration response message must be financial and should be important for the processing of the batch administration request. Note: Information regarding the processing of the request itself should appear in an extension to <strong>BatchAdminResData</strong>; information regarding the status of a batch should appear in an extension to <strong>BatchStatus</strong>; information regarding detail for an item within the capture batch should appear in an extension to <strong>TransactionDetail</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrandBatchDetails</td>
<td>{BrandID, BatchTotals}</td>
</tr>
<tr>
<td>BrandID</td>
<td>Payment card brand (without product type)</td>
</tr>
</tbody>
</table>

| 1718 BatchStatus ::= SEQUENCE { |
| 1719   openDateTime     Date, |
| 1720   closedWhen       [0] ClosedWhen OPTIONAL, |
| 1721   batchDetails     BatchDetails, |
| 1722   batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL |
| 1723 } |

| 1706 ClosedWhen ::= SEQUENCE { |
| 1707   closeStatus    CloseStatus, |
| 1708   closeDateTime  Date|
| 1709 } |

| 1727 BatchDetails ::= SEQUENCE { |
| 1728   batchTotals     BatchTotals, |
| 1729   brandBatchDetailsSeq  BrandBatchDetailsSeq OPTIONAL |
| 1730 } |

| 1711 CloseStatus ::= ENUMERATED { |
| 1712   closedbyMerchant (0), |
| 1713   closedbyAcquirer (1) |
| 1714 } |

| 1739 BatchTotals ::= SEQUENCE { |
| 1740   transactionCountCredit     INTEGER (0..MAX), |
| 1741   transactionTotalAmtCredit  CurrencyAmount, |
| 1742   transactionCountDebit      INTEGER (0..MAX), |
| 1743   transactionTotalAmtDebit   CurrencyAmount, |
| 1744   batchTotalExtensions       [0] MsgExtensions {{BTExtensionsIOS}} OPTIONAL |
| 1745 } |

Table 31: **BatchStatus**, continued

Continued on next page
**BatchStatus**, continued

**BatchStatus** (continued)

1732 BrandBatchDetailsSeq ::= SEQUENCE SIZE(1..MAX) OF BrandBatchDetails

1734 BrandBatchDetails ::= SEQUENCE {
1735    brandID       BrandID,
1736    batchTotals  BatchTotals
1737 }

232 BrandID ::= SETString { ub-BrandID }

*Continued on next page*
BatchStatus, continued

TransactionDetail

<table>
<thead>
<tr>
<th>TransactionDetail</th>
<th>{TransIDs, AuthRRPID, BrandID, BatchSequenceNum, [ReimbursementID], TransactionAmt, TransactionAmtType, [TransactionStatus], [TransExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransIDs</td>
<td>The transaction identifiers from the authorization/capture processing of the item.</td>
</tr>
<tr>
<td>AuthRRPID</td>
<td>The RRPID that appeared in the corresponding AuthReq or AuthRevReq.</td>
</tr>
<tr>
<td>BrandID</td>
<td>Payment card brand (without product type).</td>
</tr>
<tr>
<td>BatchSequenceNum</td>
<td>The sequence number of this item within the batch.</td>
</tr>
<tr>
<td>ReimbursementID</td>
<td>Enumerated code indicating the type of reimbursement for the item.</td>
</tr>
<tr>
<td>TransactionAmt</td>
<td>The amount for the item of the type indicated by TransactionAmtType. The amount is always specified as a positive value.</td>
</tr>
<tr>
<td>TransactionAmtType</td>
<td>Enumerated code indicating the type of amount (credit or debit).</td>
</tr>
<tr>
<td>TransactionStatus</td>
<td>Enumerated code indicating the result of passing the transaction to the next upstream system.</td>
</tr>
</tbody>
</table>
| TransExtensions   | The data in an extension to the batch administration response message must be financial and should be important for the processing of the batch administration request. 

Note: Information regarding the processing of the request itself should appear in an extension to BatchAdminResData; information regarding the status of a batch should appear in an extension to BatchStatus; information regarding detail for an item within the capture batch should appear in an extension to TransactionDetail.

Table 32: TransactionDetail

Continued on next page
BatchStatus, continued

TransactionDetail (continued)

1751 TransactionDetail ::= SEQUENCE {
1752    transIDs             TransIDs,
1753    authRRPID           RRPID,
1754    brandID             BrandID,
1755    batchSequenceNum   BatchSequenceNum,
1756    reimbursementID    ReimbursementID OPTIONAL,
1757    transactionAmt     CurrencyAmount,
1758    transactionAmtType AmountType,
1759    transactionStatus  [0] TransactionStatus OPTIONAL,
1761 }

337 TransIDs ::= SEQUENCE {
338    lid-C    LocalID,
339    lid-M   [0] LocalID OPTIONAL,
340    xid      XID,
341    pReqDate Date,
342    paySysID [1] PaySysID OPTIONAL,
343    language Language -- Cardholder requested session language
344 }

232 BrandID ::= SETString { ub-BrandID }

1814 BatchSequenceNum ::= INTEGER (1..MAX)

1775 ReimbursementID ::= ENUMERATED {
1776    unspecified    (0),
1777    standard       (1),
1778    keyEntered     (2),
1779    electronic     (3),
1780    additionalData (4),
1781    enhancedData   (5),
1782    marketSpecific (6)
1783 }

1770 TransactionStatus ::= ENUMERATED {
1771    success         (0),
1772    unspecifiedFailure (1)
1773 }

Version 1.0
Location

<table>
<thead>
<tr>
<th>Location</th>
<th>{CountryCode, [City], [StateProvince], [PostalCode], [LocationID]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountryCode</td>
<td>The ISO 3166 country code for the location.</td>
</tr>
<tr>
<td>City</td>
<td>The city name of the location.</td>
</tr>
<tr>
<td>StateProvince</td>
<td>The name or abbreviation of the state or province.</td>
</tr>
<tr>
<td>PostalCode</td>
<td>The postal code of the location.</td>
</tr>
<tr>
<td>LocationID</td>
<td>An identifier that the Merchant uses to specify one of its locations</td>
</tr>
</tbody>
</table>

Location (continued)

```plaintext
286 Location ::= SEQUENCE {
287    countryCode  CountryCode,
288    city          [0] EXPLICIT SETString { ub-cityName } OPTIONAL,
289    stateProvince [1] EXPLICIT SETString { ub-stateProvince } OPTIONAL,
290    postalCode   [2] EXPLICIT SETString { ub-postalCode } OPTIONAL,
291    locationID   [3] EXPLICIT SETString { ub-locationID } OPTIONAL
292 }

261 CountryCode ::= INTEGER (1..999) -- ISO-3166 country code
```
### SaleDetail

**SaleDetail**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchID</td>
<td>Identification of the settlement batch for merchant-acquirer accounting</td>
</tr>
<tr>
<td>BatchSequenceNum</td>
<td>The sequence number of this item within the batch</td>
</tr>
<tr>
<td>PayRecurlnd</td>
<td>Enumerated transaction type</td>
</tr>
<tr>
<td>MerOrderNum</td>
<td>Merchant order number</td>
</tr>
<tr>
<td>AuthCharInd</td>
<td>Copied from AuthResPayload; see page 101</td>
</tr>
<tr>
<td>MarketSpecSaleData</td>
<td>([MarketSpecDataID], [MarketSpecCapData])</td>
</tr>
<tr>
<td>CommercialCardData</td>
<td>Description of items for this capture; see page 58. Typically, this information is only included for commercial card products under special arrangement between the merchant and the customer.</td>
</tr>
<tr>
<td>OrderSummary</td>
<td>A summary description of the order.</td>
</tr>
<tr>
<td>CustomerReferenceNumber</td>
<td>A reference number assigned to the order by the Cardholder.</td>
</tr>
<tr>
<td>CustomerServicePhone</td>
<td>The Merchant’s customer service telephone number</td>
</tr>
<tr>
<td>OKtoPrintPhoneInd</td>
<td>A Boolean value indicating if the Issuer may print the customer service telephone number on the Cardholder’s statement.</td>
</tr>
</tbody>
</table>

*Note: This field may appear in an AuthReq with CaptureNow set to TRUE or in the capture-related messages; when appearing in AuthReq, the fields noted as originating from AuthResPayload are not present.*

Table 33: SaleDetail

*Continued on next page*
SaleDetail, continued

SaleDetail (continued)

<table>
<thead>
<tr>
<th>SaleExtensions</th>
<th>The data in an extension to the sale detail must be financial and should be important for the processing of a capture request by the Payment Gateway, the financial network, or the issuer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MarketSpecDataID</td>
<td>Copied from AuthResPayload; see page 101</td>
</tr>
<tr>
<td>MarketSpecCapData</td>
<td>&lt; MarketAutoCap, MarketHotelCap, MarketTransportCap &gt;</td>
</tr>
<tr>
<td>MarketAutoCap</td>
<td>Market-specific capture data</td>
</tr>
<tr>
<td>MarketHotelCap</td>
<td>Automobile rental charge description. See page 62.</td>
</tr>
<tr>
<td>MarketTransportCap</td>
<td>Hotel charge description. See page 66.</td>
</tr>
<tr>
<td></td>
<td>Passenger transport data. See page 69.</td>
</tr>
</tbody>
</table>

Table 33: SaleDetail, continued

1920 SaleDetail ::= SEQUENCE {
1921     batchID [ 0] BatchID  OPTIONAL,
1922     batchSequenceNum [ 1] BatchSequenceNum  OPTIONAL,
1923     payRecurInd [ 2] PayRecurInd  OPTIONAL,
1924     merOrderNum [ 3] MerOrderNum  OPTIONAL,
1925     authCharInd [ 4] AuthCharInd  OPTIONAL,
1927     commercialCardData [ 6] CommercialCardData  OPTIONAL,
1928     orderSummary [ 7] EXPLICIT SETString { ub-summary }  OPTIONAL,
1929     customerReferenceNumber [ 8] EXPLICIT SETString { ub-reference }  OPTIONAL,
1930     customerServicePhone [ 9] EXPLICIT Phone  OPTIONAL,
1931     okToPrintPhoneInd [10] BOOLEAN DEFAULT TRUE,
1933    }

Continued on next page
SaleDetail, continued

SaleDetail (continued)

1920 SaleDetail ::= SEQUENCE {
1921    batchID                  [ 0] BatchID  OPTIONAL,
1922    batchSequenceNum         [ 1] BatchSequenceNum  OPTIONAL,
1923    payRecurInd              [ 2] PayRecurInd  OPTIONAL,
1924    merOrderNum              [ 3] MerOrderNum  OPTIONAL,
1925    authCharInd              [ 4] AuthCharInd  OPTIONAL,
1927    commercialCardData       [ 6] CommercialCardData  OPTIONAL,
1928    orderSummary             [ 7] EXPLICIT SETString { ub-summary }  OPTIONAL,
1929    customerReferenceNumber  [ 8] EXPLICIT SETString { ub-reference }  OPTIONAL,
1930    customerServicePhone     [ 9] EXPLICIT Phone  OPTIONAL,
1931    okToPrintPhoneInd        [10] BOOLEAN DEFAULT TRUE,
1933 }

1812 BatchID ::= INTEGER (0..MAX)
1814 BatchSequenceNum ::= INTEGER (1..MAX)
1937 PayRecurInd ::= ENUMERATED {
1938    unknown               (0),
1939    singleTransaction     (1),
1940    recurringTransaction  (2),
1941    installmentPayment    (3),
1942    otherMailOrder        (4)
1943 }
1904 MerOrderNum ::= VisibleString (SIZE(1..ub-merOrderNum))
1217 AuthCharInd ::= ENUMERATED {
1218    directMarketing      (0),
1219    recurringPayment     (1),
1220    addressVerification  (2),
1221    preferredCustomer    (3),
1222    incrementalAuth      (4)
1223 }
1890 MarketSpecSaleData ::= SEQUENCE {
1891    marketSpecDataID   MarketSpecDataID OPTIONAL,
1892    marketSpecCapData  MarketSpecCapData OPTIONAL
1893 }
3167 CommercialCardData ::= SEQUENCE {
3168    chargeInfo        [0] ChargeInfo  OPTIONAL,
3169    merchantLocation  [1] Location  OPTIONAL,
3170    shipFrom          [2] Location  OPTIONAL,
3171    shipTo            [3] Location  OPTIONAL,
3172    itemSeq           [4] ItemSeq  OPTIONAL
3173 }

Continued on next page


SaleDetail, continued

SaleDetail (continued)

1897 MarketSpecDataID ::= ENUMERATED {
1898   failedEdit  (0),
1899   auto        (1),
1900   hotel       (2),
1901   transport   (3)
1902 }

1884 MarketSpecCapData ::= CHOICE {
1885    auto-rental  [0] MarketAutoCap,
1886    hotel        [1] MarketHotelCap,
1888 }

3210 MarketAutoCap ::= SEQUENCE {
3211   renterName             [0] EXPLICIT SETString { ub-renterName }  OPTIONAL,
3212   rentalLocation         [1] Location  OPTIONAL,
3213   rentalDateTime         DateTime,
3214   autoNoShow             [2] AutoNoShow OPTIONAL,
3215   rentalAgreementNumber  [3] EXPLICIT SETString { ub-rentalnum }  OPTIONAL,
3216   referenceNumber        [4] EXPLICIT SETString { ub-rentalRefNum }  OPTIONAL,
3217   insuranceType          [5] EXPLICIT SETString { ub-insuranceType }  OPTIONAL,
3218   autoRateInfo           [6] AutoRateInfo OPTIONAL,
3219   returnLocation         [7] Location  OPTIONAL,
3220   returnDateTime         DateTime,
3221   autoCharges            AutoCharges
3222 }

3261 MarketHotelCap ::= SEQUENCE {
3262   arrivalDate           Date,
3263   hotelNoShow           [0] HotelNoShow  OPTIONAL,
3264   departureDate         Date,
3265   durationOfStay        [1] INTEGER (0..99)  OPTIONAL,
3266   folioNumber           [2] EXPLICIT SETString { ub-hotelFolio }  OPTIONAL,
3267   propertyPhone         [3] Phone  OPTIONAL,
3268   customerServicePhone  [4] Phone  OPTIONAL,
3269   programCode           [5] EXPLICIT SETString { ub-programCode }  OPTIONAL,
3270   hotelRateInfo         [6] HotelRateInfo OPTIONAL,
3271   hotelCharges          HotelCharges
3272 }

Continued on next page
SaleDetail, continued

SaleDetail (continued)

3303 MarketTransportCap ::= SEQUENCE {
3304    passengerName SETString { ub-passName },
3305    departureDate Date,
3306    origCityAirport SETString { ub-airportCode },
3307    tripLegSeq [0] TripLegSeq OPTIONAL,
3308    ticketNumber [1] EXPLICIT SETString { ub-ticketNum } OPTIONAL,
3309    travelAgencyCode [2] EXPLICIT SETString { ub-taCode } OPTIONAL,
3310    travelAgencyName [3] EXPLICIT SETString { ub-taName } OPTIONAL,
3311    restrictions [4] Restrictions OPTIONAL
3312 }

Continued on next page
### CommercialCardData

This data structure is included in “SaleDetail,” described on page 53.

<table>
<thead>
<tr>
<th>CommercialCardData</th>
<th>{{[ChargeInfo, [MerchantLocation], [ShipFrom], [ShipTo], [ItemSeq]]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChargeInfo</td>
<td>{{[TotalFreightShippingAmount], [TotalDutyTariffAmount], [DutyTariffReference], [TotalNationalTaxAmount], [TotalLocalTaxAmount], [TotalOtherTaxAmount], [TotalTaxAmount], [MerchantTaxID], [MerchantDutyTariffRef], [CustomerDutyTariffRef], [SummaryCommodityCode], [MerchantType]}}</td>
</tr>
<tr>
<td>MerchantLocation</td>
<td>Location: see page 52</td>
</tr>
<tr>
<td>ShipFrom</td>
<td>Location: see page 52</td>
</tr>
<tr>
<td>ShipTo</td>
<td>Location: see page 52</td>
</tr>
<tr>
<td>ItemSeq</td>
<td>{{Item +}}</td>
</tr>
<tr>
<td>ItemSeq</td>
<td>1 to 999 item level detail records</td>
</tr>
<tr>
<td>TotalFreightShippingAmount</td>
<td>The total amount added to the order for shipping and handling.</td>
</tr>
<tr>
<td>TotalDutyTariffAmount</td>
<td>The total amount of duties or tariff for the order.</td>
</tr>
<tr>
<td>DutyTariffReference</td>
<td>The reference number assigned to the duties or tariff for the order.</td>
</tr>
<tr>
<td>TotalNationalTaxAmount</td>
<td>The total amount of national tax (sales or VAT) applied to the order.</td>
</tr>
<tr>
<td>TotalLocalTaxAmount</td>
<td>The total amount of local tax applied to the order.</td>
</tr>
<tr>
<td>TotalOtherTaxAmount</td>
<td>The total amount of other taxes applied to the order.</td>
</tr>
<tr>
<td>TotalTaxAmount</td>
<td>The total amount of taxes applied to the order.</td>
</tr>
<tr>
<td>MerchantTaxID</td>
<td>The tax identification number of the Merchant.</td>
</tr>
</tbody>
</table>

| Table 34: CommercialCardData |

Continued on next page
**SaleDetail, continued**

**CommercialCardData (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantDutyTariffRef</td>
<td>The duty or tariff reference number assigned to the Merchant.</td>
</tr>
<tr>
<td>CustomerDutyTariffRef</td>
<td>The duty or tariff reference number assigned to the Cardholder.</td>
</tr>
<tr>
<td>SummaryCommodityCode</td>
<td>The commodity code that applies to the entire order.</td>
</tr>
<tr>
<td>MerchantType</td>
<td>The type of merchant.</td>
</tr>
<tr>
<td>Item</td>
<td>{Quantity, [UnitOfMeasureCode], Descriptor, [CommodityCode], [ProductCode], [UnitCost], [NetCost], DiscountInd, [DiscountAmount], [NationalTaxAmount], [NationalTaxRate], [NationalTaxType], [LocalTaxAmount], [OtherTaxAmount], ItemTotalCost}</td>
</tr>
<tr>
<td>Quantity</td>
<td>The quantity for the line item.</td>
</tr>
<tr>
<td>UnitOfMeasureCode</td>
<td>The unit of measure for the line item.</td>
</tr>
<tr>
<td>Descriptor</td>
<td>A description of the line item.</td>
</tr>
<tr>
<td>CommodityCode</td>
<td>The commodity code for the line item.</td>
</tr>
<tr>
<td>ProductCode</td>
<td>The product code for the line item.</td>
</tr>
<tr>
<td>UnitCost</td>
<td>The unit cost of the line item.</td>
</tr>
<tr>
<td>NetCost</td>
<td>The net cost per unit of the line item.</td>
</tr>
<tr>
<td>DiscountInd</td>
<td>Indicates if a discount was applied.</td>
</tr>
<tr>
<td>DiscountAmount</td>
<td>The amount of discount applied to the line item.</td>
</tr>
<tr>
<td>NationalTaxAmount</td>
<td>The amount of national tax (sales or VAT) applied to the line item.</td>
</tr>
<tr>
<td>NationalTaxRate</td>
<td>The national tax (sales or VAT) rate applied to the line item.</td>
</tr>
<tr>
<td>NationalTaxType</td>
<td>The type of national tax applied to the line item.</td>
</tr>
<tr>
<td>LocalTaxAmount</td>
<td>The amount of local tax applied to the line item.</td>
</tr>
<tr>
<td>OtherTaxAmount</td>
<td>The amount of other taxes applied to the line item.</td>
</tr>
<tr>
<td>ItemTotalCost</td>
<td>The total cost of the line item.</td>
</tr>
</tbody>
</table>

Table 34: CommercialCardData, continued

Continued on next page
Continued on next page
SaleDetail, continued

CommercialCardData (continued)

3192 Item ::= SEQUENCE {
3193     quantity INTEGER (1..MAX) DEFAULT 1,
3194     unitOfMeasureCode [ 0] EXPLICIT SETString { ub-unitMeasure } OPTIONAL,
3195     descriptor SETString { ub-description },
3196     commodityCode [ 1] EXPLICIT SETString { ub-commCode } OPTIONAL,
3198     unitCost [ 3] CurrencyAmount OPTIONAL,
3199     netCost [ 4] CurrencyAmount OPTIONAL,
3200     discountInd BOOLEAN DEFAULT FALSE,
3201     discountAmount [ 5] CurrencyAmount OPTIONAL,
3202     nationalTaxAmount [ 6] CurrencyAmount OPTIONAL,
3203     nationalTaxRate [ 7] FloatingPoint OPTIONAL,
3204     nationalTaxType [ 8] EXPLICIT SETString { ub-taxType } OPTIONAL,
3205     localTaxAmount [ 9] CurrencyAmount OPTIONAL,
3206     otherTaxAmount [10] CurrencyAmount OPTIONAL,
3207     itemTotalCost CurrencyAmount
3208 }

3192 Item ::= SEQUENCE {
3193     quantity INTEGER (1..MAX) DEFAULT 1,
3194     unitOfMeasureCode [ 0] EXPLICIT SETString { ub-unitMeasure } OPTIONAL,
3195     descriptor SETString { ub-description },
3196     commodityCode [ 1] EXPLICIT SETString { ub-commCode } OPTIONAL,
3198     unitCost [ 3] CurrencyAmount OPTIONAL,
3199     netCost [ 4] CurrencyAmount OPTIONAL,
3200     discountInd BOOLEAN DEFAULT FALSE,
3201     discountAmount [ 5] CurrencyAmount OPTIONAL,
3202     nationalTaxAmount [ 6] CurrencyAmount OPTIONAL,
3203     nationalTaxRate [ 7] FloatingPoint OPTIONAL,
3204     nationalTaxType [ 8] EXPLICIT SETString { ub-taxType } OPTIONAL,
3205     localTaxAmount [ 9] CurrencyAmount OPTIONAL,
3206     otherTaxAmount [10] CurrencyAmount OPTIONAL,
3207     itemTotalCost CurrencyAmount
3208 }

Continued on next page
**Table 35: MarketAutoCap**

This data describes an automobile rental, and is included in “SaleDetail,” described on page 53.

<table>
<thead>
<tr>
<th>MarketAutoCap</th>
<th>([RenterName], [RentalLocation], RentalDateTime, [AutoNoShow], [RentalAgreementNumber], [ReferenceNumber], [InsuranceType], [AutoRateInfo], [ReturnLocation], ReturnDateTime, AutoCharges)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RenterName</td>
<td>The name of the person renting the vehicle.</td>
</tr>
<tr>
<td>RentalLocation</td>
<td>Location: see page 52</td>
</tr>
<tr>
<td>RentalDateTime</td>
<td>The date (and optionally time) the vehicle was rented.</td>
</tr>
<tr>
<td>AutoNoShow</td>
<td>Enumerated code indicating that the customer failed to show up to rent the vehicle as scheduled.</td>
</tr>
<tr>
<td>RentalAgreementNumber</td>
<td>The rental agreement number.</td>
</tr>
<tr>
<td>ReferenceNumber</td>
<td>The rental reference number.</td>
</tr>
<tr>
<td>InsuranceType</td>
<td>The type of insurance selected by the renter.</td>
</tr>
<tr>
<td>AutoRateInfo</td>
<td>([AutoApplicableRate, [LateReturnHourlyRate], [DistanceRate], [FreeDistance], [VehicleClassCode], [CorporateID]])</td>
</tr>
<tr>
<td>ReturnLocation</td>
<td>Location: see page 52</td>
</tr>
<tr>
<td>ReturnDateTime</td>
<td>The date (and optionally time) the vehicle was returned.</td>
</tr>
<tr>
<td>AutoCharges</td>
<td>([RegularDistanceCharges, [LateReturnCharges], [TotalDistance], [ExtraDistanceCharges], [InsuranceCharges], [FuelCharges], [AutoTowingCharges], [OneWayDropOffCharges], [TelephoneCharges], [ViolationsCharges], [DeliveryCharges], [ParkingCharges], [OtherCharges], [TotalTaxAmount], [AuditAdjustment]])</td>
</tr>
<tr>
<td>AutoApplicableRate</td>
<td>&lt;DailyRentalRate, WeeklyRentalRate&gt;</td>
</tr>
<tr>
<td>LateReturnHourlyRate</td>
<td>The hourly charge for late returns.</td>
</tr>
<tr>
<td>DistanceRate</td>
<td>The rate charged per mile in excess of any free distance allowance.</td>
</tr>
<tr>
<td>FreeDistance</td>
<td>The distance the vehicle can travel per day without incurring an additional charge.</td>
</tr>
</tbody>
</table>

Continued on next page
### MarketAutoCap (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VehicleClassCode</td>
<td>The class of vehicle rented.</td>
</tr>
<tr>
<td>CorporateID</td>
<td>The corporate identification number that applies to the rental rate.</td>
</tr>
<tr>
<td>RegularDistanceCharges</td>
<td>The amount of charges for the rental (excluding extras classified below).</td>
</tr>
<tr>
<td>LateReturnCharges</td>
<td>The amount of charges for returning the vehicle after the date and time due back.</td>
</tr>
<tr>
<td>TotalDistance</td>
<td>The total distance the vehicle was driven.</td>
</tr>
<tr>
<td>ExtraDistanceCharges</td>
<td>The amount of the charges resulting from exceeding the free distance allowance.</td>
</tr>
<tr>
<td>InsuranceCharges</td>
<td>The amount of charges resulting from insurance.</td>
</tr>
<tr>
<td>FuelCharges</td>
<td>The amount of refueling charges.</td>
</tr>
<tr>
<td>AutoTowingCharges</td>
<td>The amount of charges resulting from towing.</td>
</tr>
<tr>
<td>OneWayDropOffCharges</td>
<td>The amount of the drop-off charges resulting from a one-way rental.</td>
</tr>
<tr>
<td>TelephoneCharges</td>
<td>The amount of charges resulting from the use of the rental vehicle telephone.</td>
</tr>
<tr>
<td>ViolationsCharges</td>
<td>The amount of charges resulting from violations assessed during the rental period.</td>
</tr>
<tr>
<td>DeliveryCharges</td>
<td>The amount of charges resulting from the delivery of the rental vehicle.</td>
</tr>
<tr>
<td>ParkingCharges</td>
<td>The amount of charges resulting from parking the rental vehicle.</td>
</tr>
<tr>
<td>OtherCharges</td>
<td>The amount of other charges not classified elsewhere.</td>
</tr>
<tr>
<td>TotalTaxAmount</td>
<td>The total amount of taxes applied to the rental.</td>
</tr>
<tr>
<td>AuditAdjustment</td>
<td>The amount the transaction was adjusted as a result of auditing by the rental company.</td>
</tr>
<tr>
<td>DailyRentalRate</td>
<td>The daily rental rate.</td>
</tr>
<tr>
<td>WeeklyRentalRate</td>
<td>The weekly rental rate.</td>
</tr>
</tbody>
</table>

Table 35: MarketAutoCap, continued

Continued on next page
SaleDetail, continued

MarketAutoCap (continued)

3210 MarketAutoCap ::= SEQUENCE {
3211   renterName             [0] EXPLICIT SETString { ub-renterName } OPTIONAL,
3212   rentalLocation         [1] Location OPTIONAL,
3213   rentalDateTime         DateTime,
3214   autoNoShow             [2] AutoNoShow OPTIONAL,
3215   rentalAgreementNumber  [3] EXPLICIT SETString { ub-rentalNum } OPTIONAL,
3216   referenceNumber        [4] EXPLICIT SETString { ub-rentalRefNum } OPTIONAL,
3217   insuranceType          [5] EXPLICIT SETString { ub-insuranceType } OPTIONAL,
3218   autoRateInfo           [6] AutoRateInfo OPTIONAL,
3219   returnLocation         [7] Location OPTIONAL,
3220   returnDateTime         DateTime,
3221   autoCharges            AutoCharges
3222 }

3224 AutoNoShow ::= ENUMERATED {
3225   normalVehicle   (0),
3226   specialVehicle  (1)
3227 }

3229 AutoRateInfo ::= SEQUENCE {
3230   autoApplicableRate    AutoApplicableRate,
3231   lateReturnHourlyRate  [0] CurrencyAmount OPTIONAL,
3232   distanceRate          [1] CurrencyAmount OPTIONAL,
3233   freeDistance          [2] Distance OPTIONAL,
3234   vehicleClassCode      [3] EXPLICIT SETString { ub-vehicleClass } OPTIONAL,
3235   corporateID           [4] EXPLICIT SETString { ub-corpID } OPTIONAL
3236 }

3243 AutoCharges ::= SEQUENCE {
3244   regularDistanceCharges  CurrencyAmount,
3245   lateReturnCharges       [0] CurrencyAmount OPTIONAL,
3246   totalDistance           [1] Distance OPTIONAL,
3247   extraDistanceCharges    [2] CurrencyAmount OPTIONAL,
3248   insuranceCharges        [3] CurrencyAmount OPTIONAL,
3249   fuelCharges             [4] CurrencyAmount OPTIONAL,
3250   autoTowingCharges       [5] CurrencyAmount OPTIONAL,
3251   oneWayDropOffCharges    [6] CurrencyAmount OPTIONAL,
3252   telephoneCharges        [7] CurrencyAmount OPTIONAL,
3253   violationsCharges       [8] CurrencyAmount OPTIONAL,
3254   deliveryCharges         [9] CurrencyAmount OPTIONAL,
3255   parkingCharges          [10] CurrencyAmount OPTIONAL,
3256   otherCharges            [11] CurrencyAmount OPTIONAL,
3257   totalTaxAmount          [12] CurrencyAmount OPTIONAL,
3258   auditAdjustment         [13] CurrencyAmount OPTIONAL
3259 }

Continued on next page
SaleDetail, continued

MarketAutoCap (continued)

3238 AutoApplicableRate ::= CHOICE {
3239   dailyRentalRate  [0] CurrencyAmount,
3240   weeklyRentalRate [1] CurrencyAmount
3241 }

261 CountryCode ::= INTEGER (1..999)   -- ISO-3166 country code

Continued on next page
SaleDetail, continued

MarketHotelCap

This data describes a hotel stay, and is included in “SaleDetail,” described on page 53.

<table>
<thead>
<tr>
<th>MarketHotelCap</th>
<th>{ArrivalDate, [HotelNoShow], DepartureDate, [DurationOfStay], [FolioNumber], [PropertyPhone], [CustomerServicePhone], [ProgramCode], [HotelRateInfo], HotelCharges}</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrivalDate</td>
<td>The date the Cardholder checked in (or was scheduled to check in) to the hotel.</td>
</tr>
<tr>
<td>HotelNoShow</td>
<td>Enumerated code indicating that the customer failed to check in to the hotel as scheduled.</td>
</tr>
<tr>
<td>DepartureDate</td>
<td>The date the Cardholder checked out of the hotel.</td>
</tr>
<tr>
<td>DurationOfStay</td>
<td>The number of days the Cardholder stayed in the hotel.</td>
</tr>
<tr>
<td>FolioNumber</td>
<td>The folio number.</td>
</tr>
<tr>
<td>PropertyPhone</td>
<td>The telephone number of the hotel.</td>
</tr>
<tr>
<td>CustomerServicePhone</td>
<td>The customer service telephone number (of the hotel or the hotel chain).</td>
</tr>
<tr>
<td>ProgramCode</td>
<td>A code indicating the type of special program that applies to the stay.</td>
</tr>
<tr>
<td>HotelRateInfo</td>
<td>{DailyRoomRate, [DailyTaxRate]}</td>
</tr>
<tr>
<td>HotelCharges</td>
<td>{RoomCharges, [RoomTax], [PrepaidExpenses], [FoodBeverageCharges], [RoomServiceCharges], [MiniBarCharges], [LaundryCharges], [TelephoneCharges], [BusinessCenterCharges], [ParkingCharges], [MovieCharges], [HealthClubCharges], [GiftShopPurchases], [FolioCashAdvances], [OtherCharges], [TotalTaxAmount], [AuditAdjustment]}</td>
</tr>
<tr>
<td>DailyRoomRate</td>
<td>The daily room rate. This value includes applicable taxes unless the DailyTaxRate is specified.</td>
</tr>
<tr>
<td>DailyTaxRate</td>
<td>The amount of taxes applied to the daily room rate</td>
</tr>
</tbody>
</table>

Table 36: MarketHotelCap

Continued on next page
**SaleDetail, continued**

**MarketHotelCap (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoomCharges</td>
<td>The total amount charged for the room (excluding extras classified below).</td>
</tr>
<tr>
<td>RoomTax</td>
<td>The amount of tax applied to the RoomCharges.</td>
</tr>
<tr>
<td>PrepaidExpenses</td>
<td>The total amount of pre-paid expenses.</td>
</tr>
<tr>
<td>FoodBeverageCharges</td>
<td>The total amount of food and beverage charges.</td>
</tr>
<tr>
<td>RoomServiceCharges</td>
<td>The total amount of room service charges.</td>
</tr>
<tr>
<td>MiniBarCharges</td>
<td>The total amount of mini bar charges.</td>
</tr>
<tr>
<td>LaundryCharges</td>
<td>The total amount of laundry charges.</td>
</tr>
<tr>
<td>TelephoneCharges</td>
<td>The total amount of telephone charges.</td>
</tr>
<tr>
<td>BusinessCenterCharges</td>
<td>The total amount of business center charges.</td>
</tr>
<tr>
<td>ParkingCharges</td>
<td>The total amount of parking charges.</td>
</tr>
<tr>
<td>MovieCharges</td>
<td>The total amount of in-room movie charges.</td>
</tr>
<tr>
<td>HealthClubCharges</td>
<td>The total amount of health club charges.</td>
</tr>
<tr>
<td>GiftShopPurchases</td>
<td>The total amount of gift shop purchase charges.</td>
</tr>
<tr>
<td>FolioCashAdvances</td>
<td>The total amount of cash advances applied to the room.</td>
</tr>
<tr>
<td>OtherCharges</td>
<td>The total amount of other charges (not classified above).</td>
</tr>
<tr>
<td>TotalTaxAmount</td>
<td>The total amount of taxes applied to the bill.</td>
</tr>
<tr>
<td>Audit Adjustment</td>
<td>The amount the transaction was adjusted as a result of auditing by the hotel.</td>
</tr>
</tbody>
</table>

Table 36: MarketHotelCap, continued

```plaintext
3261 MarketHotelCap ::= SEQUENCE {
3262    arrivalDate       Date,              
3263    hotelNoShow       [0] HotelNoShow OPTIONAL, 
3264    departureDate     Date,              
3265    durationOfStay    [1] INTEGER (0..99) OPTIONAL, 
3266    folioNumber       [2] EXPLICIT SETString { ub-hotelFolio } OPTIONAL, 
3267    propertyPhone     [3] Phone OPTIONAL,  
3268    customerServicePhone [4] Phone OPTIONAL, 
3269    programCode       [5] EXPLICIT SETString { ub-programCode } OPTIONAL, 
3270    hotelRateInfo      [6] HotelRateInfo OPTIONAL,  
3271    hotelCharges       HotelCharges,      
3272 }

3274 HotelNoShow ::= ENUMERATED { 
3275    guaranteedLateArrival (0)  
3276 }

3278 HotelRateInfo ::= SEQUENCE { 
3279    dailyRoomRate  CurrencyAmount,      
3280    dailyTaxRate  CurrencyAmount OPTIONAL
3281 }

Continued on next page
SaleDetail, continued

MarketHotelCap (continued)

3283 HotelCharges ::= SEQUENCE {
3284   roomCharges            CurrencyAmount,
3285   roomTax                [ 0] CurrencyAmount  OPTIONAL,
3286   prepaidExpenses       [ 1] CurrencyAmount  OPTIONAL,
3287   foodBeverageCharges   [ 2] CurrencyAmount  OPTIONAL,
3288   roomServiceCharges    [ 3] CurrencyAmount  OPTIONAL,
3289   miniBarCharges         [ 4] CurrencyAmount  OPTIONAL,
3290   laundryCharges        [ 5] CurrencyAmount  OPTIONAL,
3291   telephoneCharges      [ 6] CurrencyAmount  OPTIONAL,
3292   businessCenterCharges [ 7] CurrencyAmount  OPTIONAL,
3293   parkingCharges        [ 8] CurrencyAmount  OPTIONAL,
3294   movieCharges          [ 9] CurrencyAmount  OPTIONAL,
3295   healthClubCharges     [10] CurrencyAmount  OPTIONAL,
3296   giftShopPurchases     [11] CurrencyAmount  OPTIONAL,
3297   folioCashAdvances     [12] CurrencyAmount  OPTIONAL,
3298   otherCharges          [13] CurrencyAmount  OPTIONAL,
3299   totalTaxAmount        [14] CurrencyAmount  OPTIONAL,
3300   auditAdjustment       [15] CurrencyAmount  OPTIONAL,
3301 }

Continued on next page
**MarketTransportCap**

This data describes passenger transport transaction, and is included in “SaleDetail,” described on page 53.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PassengerName</td>
<td>The name of the passenger to whom the tickets were issued.</td>
</tr>
<tr>
<td>DepartureDate</td>
<td>The departure date.</td>
</tr>
<tr>
<td>OrigCityAirport</td>
<td>The city of origin for the trip.</td>
</tr>
<tr>
<td>TripLegSeq</td>
<td>(TripLeg +) 1 to 16 TripLeg records</td>
</tr>
<tr>
<td>TicketNumber</td>
<td>The ticket number.</td>
</tr>
<tr>
<td>TravelAgencyCode</td>
<td>The travel agency code.</td>
</tr>
<tr>
<td>TravelAgencyName</td>
<td>The travel agency name.</td>
</tr>
<tr>
<td>Restrictions</td>
<td>Enumerated code indicating restrictions on refunds or changes.</td>
</tr>
<tr>
<td>DateOfTravel</td>
<td>The date of travel for this trip leg.</td>
</tr>
<tr>
<td>CarrierCode</td>
<td>The carrier code for this trip leg.</td>
</tr>
<tr>
<td>ServiceClass</td>
<td>The class of service for this trip leg.</td>
</tr>
<tr>
<td>StopOverCode</td>
<td>Enumerated code indicating whether stopovers are permitted for this trip leg.</td>
</tr>
<tr>
<td>DestCityAirport</td>
<td>The destination city for this trip leg.</td>
</tr>
<tr>
<td>FareBasisCode</td>
<td>The fare basis code for this trip leg.</td>
</tr>
<tr>
<td>DepartureTax</td>
<td>The departure tax for this trip leg.</td>
</tr>
</tbody>
</table>

### Table 37: MarketTransportCap

```
3303 MarketTransportCap ::= SEQUENCE {
3304   passengerName SETString { ub-passName },
3305   departureDate Date,
3306   origCityAirport SETString { ub-airportCode },
3307   tripLegSeq [0] TripLegSeq OPTIONAL,
3308   ticketNumber [1] EXPLICIT SETString { ub-ticketNum } OPTIONAL,
3309   travelAgencyCode [2] EXPLICIT SETString { ub-taCode } OPTIONAL,
3310   travelAgencyName [3] EXPLICIT SETString { ub-taName } OPTIONAL,
3311   restrictions [4] Restrictions OPTIONAL
3312 }
```

Continued on next page
SaleDetail, continued

MarketTransportCap (continued)

3314 TripLegSeq ::= SEQUENCE SIZE(1..16) OF TripLeg

3316 TripLeg ::= SEQUENCE {
    dateOfTravel    Date,
    carrierCode     SETString { ub-carrierCode },
    serviceClass    SETString { ub-serviceClass },
    stopOverCode    StopOverCode,
    destCityAirport SETString { ub-airportCode },
    fareBasisCode   [0] SETString { ub-fareBasis }  OPTIONAL,
    departureTax    [1] CurrencyAmount  OPTIONAL
}

3326 StopOverCode ::= ENUMERATED {
    noStopOverPermitted  (0),
    stopOverPermitted    (1)
}

Version 1.0
Chapter 4
Payment Messages

Overview

Introduction
Chapter 4 outlines the contents of all payment messages. Certain protocol and payload components were defined in Chapter 3.

Organization
This chapter includes the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase Initialization Pair</td>
<td>72</td>
</tr>
<tr>
<td>Purchase Pair</td>
<td>76</td>
</tr>
<tr>
<td>Purchase Inquiry Pair</td>
<td>90</td>
</tr>
<tr>
<td>Authorization Pair</td>
<td>92</td>
</tr>
<tr>
<td>Authorization Reversal Pair</td>
<td>106</td>
</tr>
<tr>
<td>Capture Pair</td>
<td>112</td>
</tr>
<tr>
<td>Capture Reversal Or Credit</td>
<td>121</td>
</tr>
<tr>
<td>Capture Reversal Pair</td>
<td>129</td>
</tr>
<tr>
<td>Credit Pair</td>
<td>132</td>
</tr>
<tr>
<td>Credit Reversal Pair</td>
<td>135</td>
</tr>
</tbody>
</table>
Purchase Initialization Pair

The purchase initialization pair is optional. Its main purpose is:

- to provide the Cardholder with necessary Merchant and Payment Gateway certificates, and
- to allow the Merchant system to generate `XID` and `PReqDate`.

If the purchase initialization pair is not present, then the Cardholder system must obtain the certificates out of band to the protocol and generate `XID` and `PReqDate`.

**PlInitReq**

The purchase initialization pair is optional. Its main purpose is:

- to provide the Cardholder with necessary Merchant and Payment Gateway certificates, and
- to allow the Merchant system to generate `XID` and `PReqDate`.

If the purchase initialization pair is not present, then the Cardholder system must obtain the certificates out of band to the protocol and generate `XID` and `PReqDate`.

<table>
<thead>
<tr>
<th>PlInitReq</th>
<th>(RRPID, Language, LID-C, [LID-M], Chall-C, BrandID, BIN, [Thumbs], [PIRqExtensions])</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>Language</td>
<td>Cardholder’s natural language</td>
</tr>
<tr>
<td>LID-C</td>
<td>Local ID; convenience label generated by and for the Cardholder system</td>
</tr>
<tr>
<td>LID-M</td>
<td>Copied from SET initiation messages (if present) described in the External Interface Guide.</td>
</tr>
<tr>
<td>Chall-C</td>
<td>Cardholder’s challenge to Merchant’s signature freshness</td>
</tr>
<tr>
<td>BrandID</td>
<td>Cardholder’s chosen payment card brand</td>
</tr>
<tr>
<td>BIN</td>
<td>Bank Identification Number from the cardholder’s account number (first six digits)</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Lists of Certificate, CRL, and BrandCRLIdentifier thumbprints in Cardholder’s cache</td>
</tr>
<tr>
<td>PIRqExtensions</td>
<td>Note: The purchase initialization request is not encrypted, so this extension must not contain confidential information.</td>
</tr>
</tbody>
</table>

Table 38: PlInitReq

756 PlInitReq ::= SEQUENCE { -- Purchase Initialization Request
757     rpid           RRPID,
758     language        Language,
759     localID-C       LocalID,
760     localID-M       [0] LocalID OPTIONAL,
761     chall-C         Challenge,
762     brandID         BrandID,
763     bin             BIN,
764     thumbs          [1] EXPLICIT Thumbs OPTIONAL,
765     piRqExtensions  [2] MsgExtensions {{PIRqExtensionsIOS}} OPTIONAL
766 }
Purchase Initialization Pair, continued

PInitReq (continued)

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

282 Language ::= VisibleString (SIZE(1..ub-RFC1766-language))

232 BrandID ::= SETString { ub-BrandID }

250 BIN ::= NumericString (SIZE(6)) -- Bank identification number

330 Thumbs ::= SEQUENCE {
  331   digestAlgorithm AlgorithmIdentifier {{DigestAlgorithms}},
  332   certThumbs [0] EXPLICIT Digests OPTIONAL,
  333   crlThumbs [1] EXPLICIT Digests OPTIONAL,
  334   brandCRLIdThumbs [2] EXPLICIT Digests OPTIONAL
  335 }
Purchase Initialization Pair, continued

PInitRes

<table>
<thead>
<tr>
<th>PInitRes</th>
<th>S(M, PInitResData)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PInitResData</td>
<td>{TransIDs, RRPID, Chall-C, Chall-M, [BrandCRLIdentifier], PEThumb, [Thumbs], [PIRsExtensions]}</td>
</tr>
<tr>
<td>TransIDs</td>
<td>See page 33.</td>
</tr>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>Chall-C</td>
<td>Copied from PInitReq</td>
</tr>
<tr>
<td>Chall-M</td>
<td>Merchant’s challenge to Cardholder’s signature freshness</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>List of current CRLs for all CAs under a Brand CA. See page 151.</td>
</tr>
<tr>
<td>PETHumb</td>
<td>Thumbprint of Payment Gateway key-exchange certificate</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Copied from PInitReq.</td>
</tr>
<tr>
<td>PIRsExtensions</td>
<td>Note: The purchase initialization response is not encrypted, so this extension must not contain confidential information.</td>
</tr>
</tbody>
</table>

Table 39: PInitRes

770 PInitRes ::= S { M, PInitResData }  
772 PInitResData ::= SEQUENCE { 
773     transIDs TransIDs, 
774     rrpid RRPID, 
775     chall-C Challenge, 
776     chall-M Challenge, 
777     brandCRLIdentifier [0] EXPLICIT BrandCRLIdentifier OPTIONAL, 
778     pethumb [1] EXPLICIT CertThumb, 
779     thumbs [2] EXPLICIT Thumbs OPTIONAL, 
780     pirsExtensions [3] MsgExtensions {{PIRsExtensionsIOS}} OPTIONAL 
781 }  

Continued on next page
Purchase Initialization Pair, continued

\textbf{PlInitRes} (continued)

\begin{verbatim}
337 TransIDs ::= SEQUENCE {
338     lid-C      LocalID,
339     lid-M      [0] LocalID OPTIONAL,
340     xid        XID,
341     pReqDate   Date,
342     paySysID   [1] PaySysID OPTIONAL,
343     language   Language  -- Cardholder requested session language
344 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

191 BrandCRLIdentifier ::= SIGNED {
192     EncodedBrandCRLID
193 } { CONSTRANIED BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } }

330 Thumbs ::= SEQUENCE {
331     digestAlgorithm AlgorithmIdentifier {{DigestAlgorithms}},
332     certThumbs        [0] EXPLICIT Digests OPTIONAL,
333     crlThumbs         [1] EXPLICIT Digests OPTIONAL,
334     brandCRLIdThumbs  [2] EXPLICIT Digests OPTIONAL
335 }
\end{verbatim}
Purchase Pair

Signed and unsigned versions of this message are provided:

- **PReqDualSigned** includes a “dual signature” and is used by Cardholders with certificates;
- **PReqUnsigned** uses the EXH operator and is used by Cardholders without certificates.

<table>
<thead>
<tr>
<th>PReq</th>
<th>&lt; PReqDualSigned, PReqUnsigned &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PReqDualSigned</td>
<td>See page 77.</td>
</tr>
<tr>
<td>PReqUnsigned</td>
<td>See page 79.</td>
</tr>
</tbody>
</table>

**Table 40: PReq**

```plaintext
787  PReq ::= CHOICE {
788      pReqDualSigned  [0] EXPLICIT PReqDualSigned,
789      pReqUnsigned    [1] EXPLICIT PReqUnsigned
790  }
794  PReqDualSigned ::= SEQUENCE {
795      pidDualSigned  PIDualSigned,
796      oidDualSigned  OIDualSigned
797  }
886  PReqUnsigned ::= SEQUENCE { -- Sent by cardholders without certificates
887      pidUnsigned   PIUnsigned,
888      oidUnsigned   OIUnsigned
889  }
```

Continued on next page
Purchase Pair, continued

**PReqDualSigned**

**PReqDualSigned** is the signed form of the **PReq** message, sent by Cardholders with certificates.

The Cardholder’s signature is contained in the **PISignature** field within **PIDualSigned**. As stated in “PI (Payment Instruction)” on page 35, the Cardholder’s signature is computed over the sequence \( \{ DD(PIData), DD(OIData) \} \).

The Merchant verifies the Cardholder’s signature by using the \( DD(PIData) \) implicit in the linkage contained in **OIDualSigned**, and by generating \( DD(OIData) \).

The Payment Gateway verifies the Cardholder’s signature by generating \( DD(PIData) \), and by using \( HOIData \) provided by the Merchant in **AuthReqData**.

<table>
<thead>
<tr>
<th><strong>PReqDualSigned</strong></th>
<th>{PIDualSigned, OIDualSigned}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PIDualSigned</strong></td>
<td>See “PI (Payment Instruction)” on page 35.</td>
</tr>
<tr>
<td><strong>OIDualSigned</strong></td>
<td>( L(OIData, PIData) )</td>
</tr>
<tr>
<td><strong>OIData</strong></td>
<td>See page 81.</td>
</tr>
<tr>
<td><strong>PIData</strong></td>
<td>{PIHead, PANData}</td>
</tr>
<tr>
<td></td>
<td>See page 37 for <strong>PIHead</strong>.</td>
</tr>
<tr>
<td></td>
<td>See page 45 for <strong>PANData</strong>.</td>
</tr>
</tbody>
</table>

Table 41: PReqDualSigned

794 PReqDualSigned ::= SEQUENCE {
  795   piDualSigned  PIDualSigned,
  796   oiDualSigned  OIDualSigned
}

799 PIDualSigned ::= SEQUENCE {
  800   piSignature  PISignature,
  801   exPIData     EX \{ P, PI-OILink, PANData \}
}

809 OIDualSigned ::= L \{ OIData, PIData \}
Purchase Pair, continued

\textbf{PReqDualSigned} (continued)

\begin{verbatim}
853 OIData ::= SEQUENCE { -- Order Information Data
  854   transIDs    TransIDs,  
  855   rrpid       RRPID, 
  856   chall-C     Challenge, 
  857   hod         HOD, 
  858   odSalt      Nonce, 
  859   chall-M     Challenge OPTIONAL, 
  860   brandID     BrandID, 
  861   bin         BIN, 
  862   odExtOIDs   [0] OIDList OPTIONAL, 
  863   oiExtensions [1] MsgExtensions {{OIExtensionsIOS}} OPTIONAL 
  864 }

828 PIData ::= SEQUENCE {  
  829   piHead     PIHead,  
  830   panData    PANData  
  831 }
\end{verbatim}

\textit{Continued on next page}
Purchase Pair, continued

**PReqUnsigned** Sent by Cardholders without certificates.

<table>
<thead>
<tr>
<th>PReqUnsigned</th>
<th>{PIUnsigned, OIUnsigned}</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIUnsigned</td>
<td>See “PI (Payment Instruction)” on page 35.</td>
</tr>
<tr>
<td>OIUnsigned</td>
<td>L(OIData, PIDataUnsigned)</td>
</tr>
<tr>
<td>OIData</td>
<td>See page 81.</td>
</tr>
<tr>
<td>PIDataUnsigned</td>
<td>{PIHead, PANToken}</td>
</tr>
<tr>
<td></td>
<td>See page 37 for PIHead.</td>
</tr>
<tr>
<td></td>
<td>See page 46 for PANToken.</td>
</tr>
</tbody>
</table>

Table 42: PReqUnsigned

886 PReqUnsigned ::= SEQUENCE { -- Sent by cardholders without certificates
887    piUnsigned  PIUnsigned,
888    oiUnsigned  OIUnsigned
889 }

898 PIUnsigned ::= EXH { P, PI-OILink, PANToken }

891 OIUnsigned ::= L { OIData, PIDataUnsigned }

853 OIData ::= SEQUENCE { -- Order Information Data
854    transIDs      TransIDs,
855    rpipid        RRPID,
856    chall-C       Challenge,
857    hod           HOD,
858    odSalt        Nonce,
859    chall-M       Challenge OPTIONAL,
860    brandID       BrandID,
861    bin           BIN,
862    odExtOIDs     [0] OIDList OPTIONAL,
863    oiExtensions  [1] MagExtensions {{OIExtensionsIOS}} OPTIONAL
864 }

893 PIDataUnsigned ::= SEQUENCE {
894    piHead    PIHead,
895    panToken  PANToken
896 }

---

*Continued on next page*
Purchase Pair, continued

PReqUnsigned (continued)

833 PIHead ::= SEQUENCE {
834   transIDs           TransIDs,
835   inputs            Inputs,
836   merchantID        MerchantID,
837   installRecurData  [0] InstallRecurData  OPTIONAL,
838   transStain        TransStain,
839   swIdent           SWIdent,
840   acqBackKeyData    [1] EXPLICIT BackKeyData  OPTIONAL,
842 }

314 PANToken ::= SEQUENCE {
315   pan           PAN,
316   cardExpiry    CardExpiry,
317   exNonce       Nonce
318 }

Continued on next page
**Purchase Pair, continued**

<table>
<thead>
<tr>
<th>OIData</th>
<th>{TransIDs, RRPID, Chall-C, HOD, ODSalt, [Chall-M], BrandID, BIN, [ODExtOIDs], [OIExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransID</td>
<td>Copied from PInitRes, if present; see page 33</td>
</tr>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>Chall-C</td>
<td>Copied from corresponding PInitReq; see page 72</td>
</tr>
<tr>
<td>HOD</td>
<td>DD(HODInput)</td>
</tr>
<tr>
<td></td>
<td>Links OIData to PurchAmt without copying PurchAmt into OIData, which would create confidentiality problems.</td>
</tr>
<tr>
<td>ODSalt</td>
<td>Copied from HODInput</td>
</tr>
<tr>
<td>Chall-M</td>
<td>Merchant’s challenge to Cardholder’s signature freshness</td>
</tr>
<tr>
<td>BrandID</td>
<td>Cardholder’s chosen payment card brand</td>
</tr>
<tr>
<td>BIN</td>
<td>Bank Identification Number from the cardholder’s account number (first six digits)</td>
</tr>
<tr>
<td>ODExtOIDs</td>
<td>List of object identifiers from ODExtensions in the same order as the extensions appeared in ODExtensions</td>
</tr>
<tr>
<td>OIExtensions</td>
<td>The data in an extension to the OI should relate to the Merchant’s processing of the order.</td>
</tr>
<tr>
<td></td>
<td>Note: The order information is not encrypted so this extension must not contain confidential information.</td>
</tr>
</tbody>
</table>

Table 43: OIData

Continued on next page
Purchase Pair, continued

OIData (continued)

<table>
<thead>
<tr>
<th>HODInput</th>
<th>{OD, PurchAmt, ODSalt, [InstallRecurData], [ODExtensions]}</th>
</tr>
</thead>
</table>

**OD**
The Order Description. This information is exchanged between the Cardholder and the Merchant out-of-band to SET. The contents, which are determined by the Merchant’s processing requirements, will include information such as the description of the items ordered (including quantity, size, price, etc.), the shipping address, and the Cardholder’s billing address (if required).

**PurchAmt**
The amount of the transaction as specified by the Cardholder; this must match the value in PIHead on page 37.

**ODSalt**
Fresh Nonce generated by Cardholder to prevent dictionary attacks on HOD

**InstallRecurData**
See page 42

**ODExtensions**
The data in an extension to the OD should relate to the Merchant’s processing of the order.
The information in these extensions must be independently known to both the Cardholder and Merchant.

Table 43: OIData, continued

853 OIData ::= SEQUENCE { -- Order Information Data
  854   transIDs TransIDs,
  855   rrpid RRPID,
  856   chall-C Challenge,
  857   hod HOD,
  858   odSalt Nonce,
  859   chall-M Challenge OPTIONAL,
  860   brandID BrandID,
  861   bin BIN,
  862   odExtOIDs [0] OIDList OPTIONAL,
  863   oiExtensions [1] MsgExtensions {{OIExtensionsIOS}} OPTIONAL
  864 }
Purchase Pair, continued

OIData (continued)

337 TransIDs ::= SEQUENCE {
338   lid-C  LocalID,  
339   lid-M  [0] LocalID  OPTIONAL,  
340   xid   XID,  
341   pReqDate  Date,  
342   paySysID  [1] PaySysID  OPTIONAL,  
343   language  Language          -- Cardholder requested session language  
344 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

870 HOD ::= DD { HODInput }  

232 BrandID ::= SETString { ub-BrandID }  

250 BIN ::= NumericString (SIZE(6))            -- Bank identification number

872 HODInput ::= SEQUENCE {
873   od   OD,  
874   purchAmt  CurrencyAmount,  
875   odSalt  Nonce,  
876   installRecurData  [0] InstallRecurData  OPTIONAL,  
877   odExtensions  [1] MsgExtensions {{ODExtensionsIOS}} OPTIONAL  
878 }

882 OD ::= OCTET STRING  

1945 InstallRecurData ::= SEQUENCE {
1946   installRecurInd  InstallRecurInd,  
1947   irExtensions  [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL  
1948 }

Continued on next page
Purchase Pair, continued

PRes

<table>
<thead>
<tr>
<th>PRes</th>
<th>S(M, PResData)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PResData</td>
<td>{TransIDs, RRPID, Chall-C, [BrandCRLIdentifier], PResPayloadSeq}</td>
</tr>
</tbody>
</table>

TransIDs

Copied from PReq: see page 33

RRPID

Request/response pair ID

Chall-C

Copied from corresponding PInitReq: see page 72

BrandCRLIdentifier

List of current CRLs for all CAs under a Brand CA. See page 151.

PResPayloadSeq

{PResPayload +}

One entry per Authorization performed. Note: a reversal removes the data from PResPayload.

If no authorizations have been performed, a single entry with the appropriate status appears.

PResPayload

See page 86.

Table 44: PRes

903 PRes ::= S { M, PResData }

905 PResData ::= SEQUENCE {
  906   transIDs             TransIDs,
  907   rrpid                RRPID,
  908   chall-C              Challenge,
  909   brandCRLIdentifier   [0] EXPLICIT BrandCRLIdentifier OPTIONAL,
  910   pResPayloadSeq       PResPayloadSeq
}

337 TransIDs ::= SEQUENCE {
  338   lid-C                LocalID,
  339   lid-M                [0] LocalID OPTIONAL,
  340   xid                  XID,
  341   pReqDate             Date,
  342   paySysID             [1] PaySysID OPTIONAL,
  343   language             Language          -- Cardholder requested session language
}

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

191 BrandCRLIdentifier ::= SIGNED {
  192   EncodedBrandCRLID
}

193 } { CONSTRAINED BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } }

Continued on next page
Purchase Pair, continued

PRes

913 PResPayloadSeq ::= SEQUENCE SIZE(1..MAX) OF PResPayload

915 PResPayload ::= SEQUENCE {
916    completionCode  CompletionCode,
917    results         Results  OPTIONAL,
918    pRsExtensions   [0]  MsgExtensions {{PRsExtensionsIOS}} OPTIONAL
919  }

Continued on next page
### Purchase Pair, continued

<table>
<thead>
<tr>
<th>PResPayload</th>
<th>{CompletionCode, [Results], [PRsExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompletionCode</td>
<td>Enumerated code indicating completion status of transaction.</td>
</tr>
<tr>
<td>Results</td>
<td>{[AcqCardMsg], [AuthStatus], [CapStatus], [CredStatusSeq]}</td>
</tr>
<tr>
<td>PRsExtensions</td>
<td>Note: The purchase response is not encrypted so this extension must not contain confidential information.</td>
</tr>
<tr>
<td>AcqCardMsg</td>
<td>Copied from AuthRes. See page 43.</td>
</tr>
<tr>
<td>AuthStatus</td>
<td>{AuthDate, AuthCode, AuthRatio, [CurrConv]}</td>
</tr>
<tr>
<td>CapStatus</td>
<td>{CapDate, CapCode, CapRatio}</td>
</tr>
<tr>
<td></td>
<td>Data only appears if CapReq corresponding to the Authorization has been performed. Note: a CapRevReq removes the data.</td>
</tr>
<tr>
<td>CredStatusSeq</td>
<td>{CreditStatus +}</td>
</tr>
<tr>
<td></td>
<td>Data only appears if CredReq corresponding to the Authorization has been performed. Note: a CredRevReq removes the data.</td>
</tr>
<tr>
<td>AuthDate</td>
<td>Date of authorization; copied from AuthRTags.Date (see page 92)</td>
</tr>
<tr>
<td>AuthCode</td>
<td>Enumerated code indicating outcome of payment authorization processing; copied from AuthResPayload (see page 101)</td>
</tr>
<tr>
<td>AuthRatio</td>
<td>AuthReqAmt ÷ PurchAmt</td>
</tr>
<tr>
<td></td>
<td>For AuthReqAmt, see “AuthReqPayload” on page 95 or AuthNewAmt, see “AuthRevReq” on page 106.</td>
</tr>
<tr>
<td></td>
<td>For PurchAmt, see “OIData” on page 81. After a partial reversal, the new amount replaces the original amount.</td>
</tr>
<tr>
<td>CurrConv</td>
<td>{CurrConvRate, CardCurr}</td>
</tr>
<tr>
<td></td>
<td>Currency conversion information; copied from AuthResPayload (see page 101)</td>
</tr>
</tbody>
</table>

Table 45: PResPayload

---

**Continued on next page**
Purchase Pair, continued

PResPayload (continued)

<table>
<thead>
<tr>
<th>CapDate</th>
<th>Date of capture; copied from CapPayload (see page 115)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapCode</td>
<td>Enumerated code indicating status of capture; copied from CapResPayload (see page 119)</td>
</tr>
<tr>
<td>CapRatio</td>
<td>CapReqAmt ÷ PurchAmt</td>
</tr>
<tr>
<td></td>
<td>For CapReqAmt, see “CapPayload” on page 115. For PurchAmt, see “OIData” on page 81.</td>
</tr>
<tr>
<td>CreditStatus</td>
<td>{CreditDate, CreditCode, CreditRatio}</td>
</tr>
<tr>
<td></td>
<td>Data only appears if corresponding CreditReq has been performed. Note: A CredRevReq removes the data.</td>
</tr>
<tr>
<td>CreditDate</td>
<td>Date of credit; copied from CapRevOrCredReqData. CapRevOrCredReqDate (see page 132)</td>
</tr>
<tr>
<td>CreditCode</td>
<td>Enumerated code indicating status of credit; copied from CapRevOrCredResPayload.CapRevOrCredCode (see page 127)</td>
</tr>
<tr>
<td>CreditRatio</td>
<td>CapRevOrCredReqAmt ÷ PurchAmt</td>
</tr>
<tr>
<td></td>
<td>For CapRevOrCredReqAmt, see “CapRevOrCredReqData” on page 122.</td>
</tr>
<tr>
<td></td>
<td>For PurchAmt, see “OIData” on page 81.</td>
</tr>
</tbody>
</table>

Table 45: PResPayload, continued

915 PResPayload ::= SEQUENCE {
916    completionCode  CompletionCode,
917    results         Results OPTIONAL,
918    prsExtensions   [0] MsgExtensions {{PRsExtensionsIOS}} OPTIONAL
919  }

Continued on next page
Purchase Pair, continued

**PResPayload (continued)**

923 CompletionCode ::= ENUMERATED {
924    meaninglessRatio        (0),  -- PurchAmt = 0; ratio cannot be computed
925    orderRejected           (1),  -- Merchant cannot process order
926    orderReceived           (2),  -- No processing to report
927    orderNotReceived        (3),  -- InqReq received without PReq
928    authorizationPerformed  (4),  -- See AuthStatus for details
929    capturePerformed        (5),  -- See CapStatus for details
930    creditPerformed         (6)   -- See CreditStatus for details
931 }

933 Results ::= SEQUENCE {
934    acqCardMsg     [0] EXPLICIT AcqCardMsg  OPTIONAL,
935    authStatus     [1] AuthStatus  OPTIONAL,
936    capStatus      [2] CapStatus  OPTIONAL,
937    credStatusSeq  [3] CreditStatusSeq  OPTIONAL
938 }

1104 AcqCardMsg ::= EncK { AcqBackKey, P, AcqCardCodeMsg }

940 AuthStatus ::= SEQUENCE {
941    authDate   Date,
942    authCode   AuthCode,
943    authRatio  FloatingPoint,
944    currConv   [0] CurrConv  OPTIONAL
945 }

947 CapStatus ::= SEQUENCE {
948    capDate   Date,
949    capCode   CapCode,
950    capRatio  FloatingPoint
951 }

1142 AuthCode ::= ENUMERATED {
1143    approved                ( 0),
1144    unspecifiedFailure      ( 1),
1145    declined                ( 2),
1146    noReply                 ( 3),
1147    callIssuer              ( 4),
1148    amountError             ( 5),
1149    expiredCard             ( 6),
1150    invalidTransaction      ( 7),
1151    systemError             ( 8),
1152    piPreviouslyUsed        ( 9),
1153    recurringTooSoon        (10),
1154    recurringExpired        (11),
1155    piAuthMismatch          (12),
1156    installRecurrMismatch   (13),
1157    captureNotSupported     (14),
1158    signatureRequired       (15),
1159    cardMerchBrandMismatch  (16)
1160 }

Continued on next page
Purchase Pair, continued

PResPayload (continued)

955 CreditStatus ::= SEQUENCE {
956    creditDate   Date,
957    creditCode   CapRevOrCredCode,
958    creditRatio  FloatingPoint
959 }

1394 CapCode ::= ENUMERATED {
1395    success             (0),
1396    unspecifiedFailure  (1),
1397    duplicateRequest    (2),
1398    authExpired         (3),
1399    authDataMissing     (4),
1400    invalidAuthData     (5),
1401    capTokenMissing     (6),
1402    invalidCapToken     (7),
1403    batchUnknown        (8),
1404    batchClosed         (9),
1405    unknownXID         (10),
1406    unknownLID         (11)
1407 }
Purchase Inquiry Pair

Table 46: InqReq

<table>
<thead>
<tr>
<th>InqReq</th>
<th>&lt; InqReqSigned, InqReqData &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>InqReqSigned</td>
<td>S(C, InqReqData)</td>
</tr>
<tr>
<td>InqReqData</td>
<td>{TransIDs, RRPID, Chall-C2, [InqRqExtensions]}</td>
</tr>
</tbody>
</table>

| TransIDs  | Copied from the most recent of the following: PReq (see page 76), PRes (see page 84), InqRes (see page 91) |
| RRPID     | Request/response pair ID |
| Chall-C2  | Fresh Cardholder challenge to Merchant’s signature |
| InqRqExtensions | Note: The inquiry request is not encrypted so this extension must not contain confidential information. |

963 InqReq ::= CHOICE {
  964   inqReqSigned    [0] EXPLICIT InqReqSigned, 
  965   inqReqUnsigned  [1] EXPLICIT InqReqData 
966 }

968 InqReqSigned ::= S { C, InqReqData }

970 InqReqData ::= SEQUENCE { -- Signed by cardholder, if signed
  971   transIDs         TransIDs, 
  972   rrpid            RRPID, 
  973   chall-C2         Challenge, 
  974   inqRqExtensions  [0] MsgExtensions {(InqRqExtensionsIOS)}  OPTIONAL 
975 }

337 TransIDs ::= SEQUENCE {
  338   lid-C      LocalID, 
  339   lid-M      [0] LocalID OPTIONAL, 
  340   xid        XID, 
  341   pReqDate   Date, 
  342   paySysID   [1] PaySysID OPTIONAL, 
  343   language   Language -- Cardholder requested session language 
344 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

Continued on next page
Purchase Inquiry Pair, continued

InqRes

| InqRes | Identical to PRes; see page 84. |

Table 47: InqRes

979 InqRes ::= PRes
Authorization Pair

AuthReq

<table>
<thead>
<tr>
<th>AuthReq</th>
<th>EncB(M, P, AuthReqData, PI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthReqData</td>
<td>{AuthReqItem, [MThumbs], CaptureNow, [SaleDetail]}</td>
</tr>
<tr>
<td>PI</td>
<td>See page 35.</td>
</tr>
<tr>
<td>AuthReqItem</td>
<td>{AuthTags, [CheckDigests], AuthReqPayload}</td>
</tr>
<tr>
<td>MThumbs</td>
<td>Thumbprints of certificates, CRLs, and Brand CRL Identifiers currently held in Merchant's cache</td>
</tr>
<tr>
<td>CaptureNow</td>
<td>Boolean indicating that capture should be performed if authorization is approved.</td>
</tr>
<tr>
<td>SaleDetail</td>
<td>See page 53.</td>
</tr>
<tr>
<td>AuthTags</td>
<td>{AuthRRTags, TransIDs, [AuthRetNum]}</td>
</tr>
<tr>
<td>CheckDigests</td>
<td>{HOIData, HOD2} Used by Payment Gateway to authenticate PI. Omit if PI is an AuthToken.</td>
</tr>
<tr>
<td>AuthReqPayload</td>
<td>See page 95.</td>
</tr>
<tr>
<td>AuthRRTags</td>
<td>RRTags, see page 34. Note: RRPIID is needed because there may be more than one authorization cycle per PReq.</td>
</tr>
<tr>
<td>TransIDs</td>
<td>Copied from corresponding OIData: see page 76</td>
</tr>
<tr>
<td>AuthRetNum</td>
<td>Identification of the authorization request used within the financial network</td>
</tr>
</tbody>
</table>

Table 48: AuthReq

Continued on next page
Authorization Pair, continued

AuthReq (continued)

<table>
<thead>
<tr>
<th>HOIData</th>
<th>DD(OIData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See page 81 for the definition of OIData.</td>
</tr>
<tr>
<td></td>
<td>An independent hash computed by Merchant. Payment Gateway compares with Cardholder-produced copy in PI to verify linkage from PI to OIData.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HOD2</th>
<th>DD(HODInput)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See “OIData” on page 81 for definition of HODInput.</td>
</tr>
<tr>
<td></td>
<td>Independent computation by Merchant. Payment Gateway compares to Cardholder-produced copy in PI to verify out-of-band receipt by Merchant of relevant data. See “OIData” on page 81.</td>
</tr>
</tbody>
</table>

Table 48: AuthReq, continued

983 AuthReq ::= EncB { M, P, AuthReqData, PI }

990 AuthReqData ::= SEQUENCE {
  991    authReqItem AuthReqItem,  
  992    mThumbs [0] EXPLICIT Thumbs OPTIONAL,  
  993    captureNow BOOLEAN DEFAULT FALSE,  
  994    saleDetail [1] SaleDetail OPTIONAL  
995 ) } ( WITH COMPONENTS { ..., captureNow (TRUE) } |  
996     WITH COMPONENTS { ..., captureNow (FALSE), saleDetail ABSENT } )

822 PI ::= CHOICE {
  823   piUnsigned [0] EXPLICIT PIUnsigned,  
  824   piDualSigned [1] EXPLICIT PIDualSigned,  
  825   authToken [2] EXPLICIT AuthToken  
826 }  

998 AuthReqItem ::= SEQUENCE {
  999    authTags AuthTags,  
  1000   checkDigests [0] CheckDigests OPTIONAL,  
  1001   authReqPayload AuthReqPayload  
1002 )

Continued on next page
Authorization Pair, continued

AuthReq (continued)

1920 SaleDetail ::= SEQUENCE {
1921    batchID [0] BatchID OPTIONAL,
1922    batchSequenceNum [1] BatchSequenceNum OPTIONAL,
1923    payRecurInd [2] PayRecurInd OPTIONAL,
1924    merOrderNum [3] MerOrderNum OPTIONAL,
1925    authCharInd [4] AuthCharInd OPTIONAL,
1927    commercialCardData [6] CommercialCardData OPTIONAL,
1928    orderSummary [7] EXPLICIT SETString { ub-summary } OPTIONAL,
1929    customerReferenceNumber [8] EXPLICIT SETString { ub-reference } OPTIONAL,
1930    customerServicePhone [9] EXPLICIT Phone OPTIONAL,
1931    okToPrintPhoneInd [10] BOOLEAN DEFAULT TRUE,
1933 }

1004 AuthTags ::= SEQUENCE {
1005    authRRTags RRTags,
1006    transIDs TransIDs,
1007    authRetNum AuthRetNum OPTIONAL
1008 }

1010 CheckDigests ::= SEQUENCE {
1011    hOIData HOIData,
1012    hod2 HOD
1013 }

1015 AuthReqPayload ::= SEQUENCE {
1016    subsequentAuthInd BOOLEAN DEFAULT FALSE,
1017    authReqAmt CurrencyAmount, -- May differ from PurchAmt
1018    avsData [0] AVSData OPTIONAL,
1019    specialProcessing [1] SpecialProcessing OPTIONAL,
1020    cardSuspect [2] CardSuspect OPTIONAL,
1021    requestCardTypeID BOOLEAN DEFAULT FALSE,
1022    installRecurData [3] InstallRecurData OPTIONAL,
1023    marketSpecAuthData [4] EXPLICIT MarketSpecAuthData OPTIONAL,
1024    merchData MerchData,
1025    aRqExtensions [5] MsgExtensions {{ARqExtensionsIOS}} OPTIONAL
1026 }

337 TransIDs ::= SEQUENCE {
338    lid-C LocalID,
339    lid-M [0] LocalID OPTIONAL,
340    xid XID,
341    pReqDate Date,
342    paySysID [1] PaySysID OPTIONAL,
343    language Language -- Cardholder requested session language
344 }

1259 AuthRetNum ::= INTEGER (0..MAX)

820 HOIData ::= DD { OIData } -- PKCS#7 DigestedData

Continued on next page
Authorization Pair, continued

**AuthReqPayload**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthReqPayload</td>
<td>{SubsequentAuthInd, AuthReqAmt, [AVSData], [SpecialProcessing], [CardSuspect], RequestCardTypeInd, [InstallRecurData], [MarketSpecAuthData], MerchData, [ARqExtensions]}</td>
</tr>
<tr>
<td>SubsequentAuthInd</td>
<td>Boolean indicating Merchant requests an additional authorization because of a split shipment</td>
</tr>
<tr>
<td>AuthReqAmt</td>
<td>May differ from PurchAmt; acquirer policy may place limitations on the permissible difference</td>
</tr>
<tr>
<td>AVSData</td>
<td>{[StreetAddress], Location}</td>
</tr>
<tr>
<td></td>
<td>Cardholder billing address; contents are received from cardholder using an out-of-band mechanism</td>
</tr>
<tr>
<td></td>
<td>See page 52 for definition of Location.</td>
</tr>
<tr>
<td>SpecialProcessing</td>
<td>Enumerated field indicating the type of special processing requested.</td>
</tr>
<tr>
<td>CardSuspect</td>
<td>Enumerated code indicating that Merchant is suspicious of the Cardholder and the reason for the suspicion</td>
</tr>
<tr>
<td>RequestCardTypeInd</td>
<td>Indicates that the type of card should be returned in CardType in the response; if the information is not available, the value unavailable(0) is returned.</td>
</tr>
<tr>
<td>InstallRecurData</td>
<td>See page 42.</td>
</tr>
<tr>
<td>MarketSpecAuthData</td>
<td>&lt; MarketAutoAuth, MarketHotelAuth, MarketTransportAuth &gt;</td>
</tr>
<tr>
<td></td>
<td>Market-specific authorization data</td>
</tr>
<tr>
<td>MerchData</td>
<td>{ [MerchCatCode], [MerchGroup]}</td>
</tr>
<tr>
<td>ARqExtensions</td>
<td>The data in an extension to the authorization request must be financial and should be related to the processing of an authorization (or subsequent capture) by the Payment Gateway, the financial network, or the issuer.</td>
</tr>
<tr>
<td>StreetAddress</td>
<td>The street address of the cardholder</td>
</tr>
</tbody>
</table>

Table 49: AuthReqPayload

Continued on next page
Authorization Pair, continued

AuthReqPayload (continued)

<table>
<thead>
<tr>
<th>Authorization Pair</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MarketAutoAuth</td>
<td>{Duration}</td>
</tr>
<tr>
<td>MarketHotelAuth</td>
<td>{Duration, [Prestige]}</td>
</tr>
<tr>
<td>MarketTransportAuth</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td>There is currently no authorization data for this market segment.</td>
</tr>
<tr>
<td>MerchCatCode</td>
<td>Four-byte code (defined in ANSI X9.10) describing Merchant’s type of business, product, or service</td>
</tr>
<tr>
<td>MerchGroup</td>
<td>Enumerated code identifying the general category of the merchant</td>
</tr>
<tr>
<td>Duration</td>
<td>The anticipated duration of the transaction (in days). This information assists the issuer by indicating how much time is likely to elapse between the authorization and the capture.</td>
</tr>
<tr>
<td>Prestige</td>
<td>Enumerated type of prestigious property; the meaning of the various levels are defined by the payment card brand</td>
</tr>
</tbody>
</table>

Table 49: AuthReqPayload, continued

1015 AuthReqPayload ::= SEQUENCE {
1016    subsequentAuthInd  BOOLEAN DEFAULT FALSE,
1017    authReqAmt          CurrencyAmount, -- May differ from PurchAmt
1018    avsData             [0] AVSData  OPTIONAL,
1019    specialProcessing   [1] SpecialProcessing  OPTIONAL,
1020    cardSuspect         [2] CardSuspect  OPTIONAL,
1021    requestCardTypeInd  BOOLEAN DEFAULT FALSE,
1022    installRecurData    [3] InstallRecurData  OPTIONAL,
1023    marketSpecAuthData  [4] EXPLICIT MarketSpecAuthData  OPTIONAL,
1024    merchData           MerchData,
1025    aRqExtensions       [5] MsgExtensions {{ARqExtensionsIOS}} OPTIONAL
1026 }
Authorization Pair, continued

AuthReqPayload (continued)

1030 AVSData ::= SEQUENCE {
1031    streetAddress SETString { ub-AVSData } OPTIONAL,
1032    location Location
1033 }

1035 SpecialProcessing ::= ENUMERATED {
1036    directMarketing (0),
1037    preferredCustomer (1),
1038 }

1040 CardSuspect ::= ENUMERATED {   -- Indicates merchant suspects cardholder
1041    -- Specific values indicate why the merchant is suspicious
1042    -- Either the merchant does not differentiate
1043    -- reasons for suspicion, or the specific
1044    unspecifiedReason (0) -- Either the merchant does not differentiate
1045    -- reason does not appear in the list
1046    -- reason does not appear in the list
1047 }

1945 InstallRecurData ::= SEQUENCE {
1946    installRecurInd InstallRecurInd,
1947    irExtensions [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL
1948 }

1878 MarketSpecAuthData ::= CHOICE {
1879    auto-rental [0] MarketAutoAuth,
1880    hotel [1] MarketHotelAuth,
1882 }

1049 MerchData ::= SEQUENCE {
1050    merchCatCode MerchCatCode OPTIONAL,
1051    merchGroup MerchGroup OPTIONAL
1052 }

1860 MarketAutoAuth ::= SEQUENCE {
1861    duration Duration
1862 }

Continued on next page
Authorization Pair, continued

AuthReqPayload (continued)

1864 MarketHotelAuth ::= SEQUENCE {
1865   duration  Duration,
1866   prestige Prestige  OPTIONAL
1867 }
1895 MarketTransportAuth ::= NULL

1054 MerchCatCode ::= NumericString (SIZE(ub-merType))  -- ANSI X9.10
1055     -- Merchant Category Code (MCCs) are assigned by acquirer to
1056     -- describe the merchant's product, service or type of business

1058 MerchGroup ::= ENUMERATED {
1059       commercialTravel  (1),
1060       lodging           (2),
1061       automobileRental (3),
1062       restaurant        (4),
1063       medical           (5),
1064       mailOrPhoneOrder (6),
1065       riskyPurchase     (7),
1066       other             (8)
1067 }

286 Location ::= SEQUENCE {
287     countryCode  CountryCode,
288     city         [0] EXPLICIT SETString { ub-cityName }  OPTIONAL,
289     stateProvince [1] EXPLICIT SETString { ub-stateProvince }  OPTIONAL,
290     postalCode   [2] EXPLICIT SETString { ub-postalCode }  OPTIONAL,
291     locationID   [3] EXPLICIT SETString { ub-locationID }  OPTIONAL
292 }
1869 Duration ::= INTEGER (1..99)                          -- Number of days

1871 Prestige ::= ENUMERATED {
1872       unknown   (0),
1873       level-1   (1),  -- Transaction floor limits for each level are
1874       level-2   (2),  -- defined by brand policy and may vary between
1875       level-3   (3)   -- national markets.
1876 }

261 CountryCode ::= INTEGER (1..999)  -- ISO-3166 country code

Continued on next page
Authorization Pair, continued

Table 50: AuthRes

1069 AuthRes ::= CHOICE {
1072 }

Continued on next page
Authorization Pair, continued

AuthRes (continued)

1089 AuthResData ::= SEQUENCE {
1090   authTags    AuthTags,
1091   brandCRLIdentifier [0] EXPLICIT BrandCRLIdentifier OPTIONAL,
1092   peThumb    [1] EXPLICIT CertThumb OPTIONAL,
1093   authResPayload AuthResPayload
1094 }

1096 AuthResBaggage ::= SEQUENCE {
1097   capToken [0] EXPLICIT CapToken OPTIONAL,
1098   acqCardMsg [1] EXPLICIT AcqCardMsg OPTIONAL,
1099   authToken [2] EXPLICIT AuthToken OPTIONAL
1100 }

314 PANToken ::= SEQUENCE {
315   pan    PAN,
316   cardExpiry CardExpiry,
317   exNonce Nonce
318 }

1004 AuthTags ::= SEQUENCE {
1005   authRRTags RRTags,
1006   transIDs TransIDs,
1007   authRetNum AuthRetNum OPTIONAL
1008 }

191 BrandCRLIdentifier ::= SIGNED {
192   EncodedBrandCRLID
193 } (CONSTRAINED BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } )

1126 AuthResPayload ::= SEQUENCE {
1127   authHeader AuthHeader,
1128   capResPayload CapResPayload OPTIONAL,
1129   arsExtensions [0] MsgExtensions {{ARsExtensionsIOS}} OPTIONAL
1130 }

1816 CapToken ::= CHOICE {
1817   encX [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
1818   enc [1] EXPLICIT Enc { P1, P2, CapTokenData },
1819   null [2] EXPLICIT NULL
1820 }

1104 AcqCardMsg ::= EncK { AcqBackKey, P, AcqCardCodeMsg }

1787 AuthToken ::= EncX { P1, P2, AuthTokenData, PANToken }

Continued on next page
Authorization Pair, continued

AuthResPayload

<table>
<thead>
<tr>
<th>AuthResPayload</th>
<th>{AuthHeader, [CapResPayload], [ARsExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthHeader</td>
<td>{AuthAmt, AuthCode, ResponseData, [BatchStatus], [CurrConv]}</td>
</tr>
<tr>
<td>CapResPayload</td>
<td>See page 119. Returned if <strong>CaptureNow</strong> had a value of <strong>TRUE</strong> in <strong>AuthReq</strong>.</td>
</tr>
<tr>
<td>ARsExtensions</td>
<td>The data in an extension to the authorization response must be financial and should be important for the processing of the authorization response or a subsequent authorization reversal or capture request by the Payment Gateway, the financial network, or the issuer.</td>
</tr>
<tr>
<td>AuthAmt</td>
<td>Copied from AuthReqPayload.AuthReqAmt</td>
</tr>
<tr>
<td>AuthCode</td>
<td>Enumerated code indicating outcome of payment authorization processing</td>
</tr>
<tr>
<td>ResponseData</td>
<td>{[AuthValCodes], [RespReason], [CardType], [AVSResult], [LogRefID]}</td>
</tr>
<tr>
<td>BatchStatus</td>
<td>See page 47.</td>
</tr>
<tr>
<td>CurrConv</td>
<td>{CurrConvRate, CardCurr}</td>
</tr>
<tr>
<td>AuthValCodes</td>
<td>{[ApprovalCode], [AuthCharInd], [ValidationCode], [MarketSpecDataID]}</td>
</tr>
<tr>
<td>RespReason</td>
<td>Enumerated code that indicates authorization service entity and (if appropriate) reason for decline</td>
</tr>
<tr>
<td>CardType</td>
<td>Enumerated code indicating the type of card used for the transaction</td>
</tr>
<tr>
<td>AVSResult</td>
<td>Enumerated Address Verification Service response code</td>
</tr>
<tr>
<td>LogRefID</td>
<td>Alphanumeric data assigned to the authorization transaction (used for matching to reversals)</td>
</tr>
</tbody>
</table>

Table 51: AuthResPayload

Continued on next page
Authorization Pair, continued

**AuthResPayload (continued)**

<table>
<thead>
<tr>
<th>CurrConvRate</th>
<th>Currency Conversion Rate: value with which to multiply AuthReqAmt to provide an amount in the Cardholder's currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>CardCurr</td>
<td>ISO 4217 currency code of Cardholder</td>
</tr>
<tr>
<td>ApprovalCode</td>
<td>Approval code assigned to the transaction by the Issuer.</td>
</tr>
<tr>
<td>AuthCharInd</td>
<td>Enumerated value that indicates the conditions present when the authorization was performed</td>
</tr>
<tr>
<td>ValidationCode</td>
<td>Four-byte alphanumeric code calculated to ensure that required fields in the authorization messages are also present in their respective clearing messages.</td>
</tr>
<tr>
<td>MarketSpecDataID</td>
<td>Enumerated code that identifies the type of market-specific data supplied on the authorization (as determined by the financial network)</td>
</tr>
</tbody>
</table>

Table 51: AuthResPayload, continued

```
1126 AuthResPayload ::= SEQUENCE {
1127   authHeader     AuthHeader,                       
1128   capResPayload CapResPayload OPTIONAL,           
1129   aRsExtensions [0] MsgExtensions {{ARsExtensionsIOS}} OPTIONAL
1130 }

1134 AuthHeader ::= SEQUENCE {
1135   authAmt       CurrencyAmount,                     
1136   authCode      AuthCode,                         
1137   responseData ResponseData,                     
1138   batchStatus  [0] BatchStatus OPTIONAL,          
1139   currConv      CurrConv OPTIONAL -- Merchant to cardholder 
1140 }

1384 CapResPayload ::= SEQUENCE {
1385   capCode CapCode,                                 
1386   capAmt CurrencyAmount,                          
1387   batchID  [0] BatchID OPTIONAL,                   
1388   batchSequenceNum [1] BatchSequenceNum OPTIONAL,  
1390 }
```

Continued on next page
Authorization Pair, continued

AuthResPayload (continued)

1142 AuthCode ::= ENumerated {
1143    approved                ( 0),
1144    unspecifiedFailure      ( 1),
1145    declined                ( 2),
1146    noReply                 ( 3),
1147    callIssuer              ( 4),
1148    amountError             ( 5),
1149    expiredCard             ( 6),
1150    invalidTransaction      ( 7),
1151    systemError             ( 8),
1152    piPreviouslyUsed        ( 9),
1153    recurringTooSoon        (10),
1154    recurringExpired        (11),
1155    piAuthMismatch          (12),
1156    installRecurMismatch    (13),
1157    captureNotSupported     (14),
1158    signatureRequired       (15),
1159    cardMerchBrandMismatch  (16)
1160 }

1162 responseData ::= SEQUENCE {
1163    authValCodes  [0] AuthValCodes  OPTIONAL,
1164    respReason    [1] RespReason  OPTIONAL,
1165    cardType      CardType  OPTIONAL,
1166    avsResult     [2] AVSResult  OPTIONAL,
1167    logRefID      LogRefID  OPTIONAL
1168 }

1718 BatchStatus ::= SEQUENCE {
1719    openDateTime     Date,
1720    closedWhen       [0] ClosedWhen  OPTIONAL,
1721    batchDetails     BatchDetails,
1722    batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL
1723 }

1853 CurrConv ::= SEQUENCE {
1854    currConvRate  FloatingPoint,
1855    cardCurr      Currency
1856 }

1170 AuthValCodes ::= SEQUENCE {
1171    approvalCode    [0] ApprovalCode  OPTIONAL,
1172    authCharInd     [1] AuthCharInd  OPTIONAL,
1173    validationCode  [2] ValidationCode  OPTIONAL,
1174    marketSpec      MarketSpecDataID  OPTIONAL
1175 }

Continued on next page
Authorization Pair, continued

AuthResPayload (continued)

1177 RespReason ::= ENUMERATED {  
1178     issuer (0),  
1179     standInTimeOut (1),  
1180     standInFloorLimit (2),  
1181     standInSuppressInquiries (3),  
1182     standInIssuerUnavailable (4),  
1183     standInIssuerRequest (5)  
1184 }

1186 CardType ::= ENUMERATED {  
1187     unavailable (0),  
1188     classic (1),  
1189     gold (2),  
1190     platinum (3),  
1191     premier (4),  
1192     debit (5),  
1193     pinBasedDebit (6),  
1194     atm (7),  
1195     electronicOnly (8),  
1196     unspecifiedConsumer (9),  
1197     corporateTravel (10),  
1198     purchasing (11),  
1199     business (12),  
1200     unspecifiedCommercial (13),  
1201     privateLabel (14),  
1202     proprietary (15)  
1203 }

1205 AVSResult ::= ENUMERATED {  
1206     resultUnavailable (0),  
1207     noMatch (1),  
1208     addressMatchOnly (2),  
1209     postalCodeMatchOnly (3),  
1210     fullMatch (4)  
1211 }

1213 LogRefID ::= NumericString (SIZE(1..ub-logRefID))

1215 ApprovalCode ::= VisibleString (SIZE(ub-approvalCode))

1217 AuthCharInd ::= ENUMERATED {  
1218     directMarketing (0),  
1219     recurringPayment (1),  
1220     addressVerification (2),  
1221     preferredCustomer (3),  
1222     incrementalAuth (4)  
1223 }

Continued on next page
Authorization Pair, continued

AuthResPayload (continued)

1225 ValidationCode ::= VisibleString (SIZE(ub-validationCode))

1897 MarketSpecDataID ::= ENUMERATED {
1898   failedEdit   (0),
1899   auto         (1),
1900   hotel        (2),
1901   transport    (3)
1902 }

Version 1.0
**Authorization Reversal Pair**

**AuthRevReq**
Merchant uses this to cancel an authorization or to reduce the amount of the authorization.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthRevReqData</td>
<td>{AuthRevTags, [MThumbs], [AuthReqData], [AuthResPayload], AuthNewAmt, [ARvRqExtensions]}</td>
</tr>
<tr>
<td>AuthRevReqBaggage</td>
<td>{Pl, [CapToken]}</td>
</tr>
<tr>
<td>AuthRevTags</td>
<td>{AuthRevRRTags, [AuthRetNum]}</td>
</tr>
<tr>
<td>MThumbs</td>
<td>Thumbprints of certificates, CRLs, and Brand CRL Identifiers currently held in Merchant’s cache</td>
</tr>
<tr>
<td>AuthReqData</td>
<td>Copied from prior, corresponding <strong>AuthReq</strong>. Not required in message if <strong>CapToken</strong> generated by Payment Gateway contains all relevant data.</td>
</tr>
<tr>
<td>AuthResPayload</td>
<td>Copied from prior, corresponding <strong>AuthRes</strong>. Not required in message if <strong>CapToken</strong> generated by Payment Gateway contains all relevant data.</td>
</tr>
<tr>
<td>AuthNewAmt</td>
<td>New authorization amount requested. A value of zero indicates that the entire Authorization should be reversed; any other value less than the original authorized amount indicates a partial reversal. Full or partial reversals are used by Issuers to adjust the Cardholder’s open to buy.</td>
</tr>
<tr>
<td>ARvRqExtensions</td>
<td>The data in an extension to the authorization reversal request must be financial and should be related to the processing of an authorization reversal (or subsequent capture) by the Payment Gateway, the financial network, or the issuer.</td>
</tr>
</tbody>
</table>

*Continued on next page*
Authorization Reversal Pair, continued

AuthRevReq (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>Copied from prior, corresponding AuthReq</td>
</tr>
<tr>
<td>CapToken</td>
<td>Copied from prior, corresponding AuthRes</td>
</tr>
<tr>
<td>AuthRevRRTags</td>
<td>RRTags, see page 34.</td>
</tr>
<tr>
<td>AuthRetNum</td>
<td>Identification of the authorization request used within the financial network</td>
</tr>
</tbody>
</table>

Table 52: AuthRevReq

Continued on next page
Authorization Reversal Pair, continued

AuthRevReq (continued)

1229 AuthRevReq ::= EncB { M, P, AuthRevReqData, AuthRevReqBaggage }

1236 AuthRevReqData ::= SEQUENCE {
1237   authRevTags     AuthRevTags,
1238   mThumbs         [0] EXPLICIT Thumbs OPTIONAL,
1239   authReqData     [1] AuthReqData OPTIONAL,
1241   authNewAmt      CurrencyAmount,
1242   aRvRqExtensions [3] MsgExtensions {{ARvRqExtensionsIOS}} OPTIONAL
1243 }

1247 AuthRevReqBaggage ::= SEQUENCE {
1248   pi              PI,
1249   capToken       CapToken OPTIONAL
1250 }

1252 AuthRevTags ::= SEQUENCE {
1253   authRevRRTags   AuthRevRRTags,
1254   authRetNum     AuthRetNum OPTIONAL
1255 }

990 AuthReqData ::= SEQUENCE {
991   authReqItem    AuthReqItem,
992   mThumbs        [0] EXPLICIT Thumbs OPTIONAL,
993   captureNow    BOOLEAN DEFAULT FALSE,
994   saleDetail     [1] SaleDetail OPTIONAL
995 } ( WITH COMPONENTS {..., captureNow (TRUE) } |
996     WITH COMPONENTS {..., captureNow (FALSE), saleDetail ABSENT } )

822 PI ::= CHOICE {
823   piUnsigned     [0] EXPLICIT PIUnsigned,
824   piDualSigned   [1] EXPLICIT PIDualSigned,
825   authToken      [2] EXPLICIT AuthToken
826 }

1816 CapToken ::= CHOICE {
1817   encX          [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
1818   enc           [1] EXPLICIT Enc { P1, P2, CapTokenData },
1819   null          [2] EXPLICIT NULL
1820 }

1257 AuthRevRRTags ::= RRTags

1259 AuthRetNum ::= INTEGER (0..MAX)

Continued on next page
Authorization Reversal Pair, continued

### AuthRevRes

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthRevResBaggage</td>
<td>{[CapTokenNew], [AuthTokenNew]}</td>
</tr>
<tr>
<td>AuthRevCode</td>
<td>Enumerated code indicating outcome of payment authorization reversal processing</td>
</tr>
<tr>
<td>AuthRevTags</td>
<td>Copied from corresponding AuthRevReq</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>List of current CRLs for all CAs under a Brand CA. See page 151.</td>
</tr>
<tr>
<td>PEThumb</td>
<td>Thumbprint of Payment Gateway certificate provided if AuthRevReq.MThumbs indicates Merchant needs one</td>
</tr>
<tr>
<td>AuthNewAmt</td>
<td>Copied from corresponding AuthRevReq</td>
</tr>
<tr>
<td>AuthResDataNew</td>
<td>{TransIDs, [AuthResPayloadNew]}</td>
</tr>
<tr>
<td></td>
<td>If AuthNewAmt is not 0, Payment Gateway creates a new instance of AuthResData (see “AuthRes” on page 99).</td>
</tr>
<tr>
<td>ARvRsExtensions</td>
<td>The data in an extension to the authorization reversal response must be financial and should be important for the processing of the authorization reversal response or a subsequent capture request by the Payment Gateway, the financial network, or the issuer.</td>
</tr>
<tr>
<td>CapTokenNew</td>
<td>New Capture Token (with updated fields), if AuthNewAmt is not 0. This replaces the CapToken returned in the corresponding AuthRes.</td>
</tr>
</tbody>
</table>

**Continued on next page**
Authorization Reversal Pair, continued

| TransIDs     | Copied from corresponding AuthRevReq |
| AuthResPayloadNew | Formally identical to AuthResPayload (see page 101); if AuthNewAmt is not 0. |

Table 53: AuthRevRes

1261 AuthRevRes ::= CHOICE {
1264 }
1278 AuthRevResData ::= SEQUENCE {
1279   authRevCode AuthRevCode,
1280   authRevTags AuthRevTags,
1281   brandCRLIdentifier [0] EXPLICIT BrandCRLIdentifier OPTIONAL,
1282   peThumb    [1] EXPLICIT CertThumb OPTIONAL,
1283   authNewAmt CurrencyAmount, -- May be zero
1284   authResDataNew AuthResDataNew,
1285   aRvRsExtensions [2] MsgExtensions {{ARvRsExtensionsIOS}} OPTIONAL
1286 }

1273 AuthRevResBaggage ::= SEQUENCE {
1274   capTokenNew CapToken OPTIONAL,
1275   authTokenNew AuthToken OPTIONAL
1276 }

Continued on next page
Authorization Reversal Pair, continued

AuthRevRes (continued)

1252 AuthRevTags ::= SEQUENCE {
1253    authRevRRTags AuthRevRRTags,
1254    authRetNum AuthRetNum OPTIONAL
1255 }

191 BrandCRLIdentifier ::= SIGNED {
192    EncodedBrandCRLID
193 } ( CONstrained by { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } )

1303 AuthResDataNew ::= SEQUENCE {
1304    transIDs TransIDs,
1305    authResPayloadNew AuthResPayload OPTIONAL -- Contains new data
1306 }

337 TransIDs ::= SEQUENCE {
338    lid-C LocalID,
339    lid-M [0] LocalID OPTIONAL,
340    xid XID,
341    pReqDate Date,
342    paySysID [1] PaySysID OPTIONAL,
343    language Language -- Cardholder requested session language
344 }
## Capture Pair

### CapReq

<table>
<thead>
<tr>
<th>CapReq</th>
<th>&lt; EncB(M, P, CapReqData, CapTokenSeq), EncBX(M, P, CapReqData, CapTokenSeq, PANToken) &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapTokenSeq</td>
<td>is external “baggage”. If PANToken is included, it must correspond to a single CapItem and a single CapToken in CapTokenSeq.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CapReqData</th>
<th>{CapRRTags, [MThumbs], CapItemSeq, [CRqExtensions]}</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CapTokenSeq</th>
<th>[{CapToken} +]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>One or more CapTokens, in ordered one-to-one correspondence with CapItems in CapItemSeq. Note: Any CapToken may be omitted; that is, may be NULL.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PANToken</th>
<th>See page 46</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CapRRTags</th>
<th>RRTags, see page 34.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fresh RRPID and Date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MThumbs</th>
<th>Thumbprints of certificates, CRLs, and Brand CRL Identifiers currently held in Merchant’s cache</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CapItemSeq</th>
<th>{CapItem} +}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>One or more CapItem in an ordered array</td>
</tr>
</tbody>
</table>

**Table 54: CapReq**

*Continued on next page*
Capture Pair, continued

CapReq (continued)

| CRqExtensions | The data in an extension to the capture request must be financial and should be important for the processing of a capture message by the Payment Gateway, the financial network, or the issuer. Note: The data in this extension applies to every item in the capture request; data related to a specific item should be placed in an extension to CapPayload. |
| CapToken | Copied from corresponding AuthRes (see page 99) or AuthRevRes (see page 109) |
| CapItem | {TransIDs, AuthRRPID, CapPayload} |
| TransIDs | Copied from corresponding AuthRes (see page 99) or AuthRevRes (see page 109) |
| AuthRRPID | The RRPID that appeared in the corresponding AuthReq (see page 92) or AuthRevReq (see page 106) |
| CapPayload | See page 115. |

Table 54: CapReq, continued

1310 CapReq ::= CHOICE {
1311   encB   [0] EXPLICIT EncB { M, P, CapReqData, CapTokenSeq },
1312   encBX  [1] EXPLICIT EncBX { M, P, CapReqData, CapTokenSeq, PANToken }
1313 }

1330 CapReqData ::= SEQUENCE {
1331   capRRTags     CapRRTags,
1332   mThumbs       [0] EXPLICIT Thumbs  OPTIONAL,
1333   capItemSeq    CapItemSeq,
1334   crqExtensions  [1] MsgExtensions {{CRqExtensionsIOS}} OPTIONAL
1335 }

1841 CapTokenSeq ::= SEQUENCE SIZE(1..MAX) OF CapToken

314 PANToken ::= SEQUENCE {
315   pan         PAN,
316   cardExpiry  CardExpiry,
317   exNonce     Nonce
318 }

1339 CapRRTags ::= RRTags

1341 CapItemSeq ::= SEQUENCE SIZE(1..MAX) OF CapItem

Continued on next page
Capture Pair, continued

CapReq (continued)

1816 CapToken ::= CHOICE {
1817   encX [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
1818   enc [1] EXPLICIT Enc { P1, P2, CapTokenData },
1819   null [2] EXPLICIT NULL
1820 }

1343 CapItem ::= SEQUENCE {
1344   transIDs    TransIDs,
1345   authRRPID   RRPID,
1346   capPayload CapPayload
1347 }

337 TransIDs ::= SEQUENCE {
338   lid-C      LocalID,
339   lid-M      [0] LocalID OPTIONAL,
340   xid        XID,
341   pReqDate   Date,
342   paySysID   [1] PaySysID OPTIONAL,
343   language   Language          -- Cardholder requested session language
344 }

1349 CapPayload ::= SEQUENCE {
1350   capDate         Date,
1351   capReqAmt       CurrencyAmount,
1352   authReqItem     [0] AuthReqItem OPTIONAL,
1354   saleDetail      [2] SaleDetail OPTIONAL,
1356 }

Continued on next page
Capture Pair, continued

CapPayload

<table>
<thead>
<tr>
<th>CapPayload</th>
<th>{CapDate, CapReqAmt, [AuthReqItem], [AuthResPayload], [SaleDetail], [CPayExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapDate</td>
<td>Date of capture; this is the Transaction Date that will appear on the Cardholder’s statement</td>
</tr>
<tr>
<td>CapReqAmt</td>
<td>Capture amount requested by Merchant, may differ from AuthAmt; this is the Transaction Amount (before any currency conversion) that will appear on the Cardholder’s statement</td>
</tr>
<tr>
<td>AuthReqItem</td>
<td>See “AuthReq” on page 92. Required if the corresponding CapToken is not present or the Payment Gateway/acquirer systems do not contain the relevant authorization request data</td>
</tr>
<tr>
<td>AuthResPayload</td>
<td>See page 101. Required if the corresponding CapToken is not present or the Payment Gateway/acquirer systems do not contain the relevant authorization response data</td>
</tr>
<tr>
<td>SaleDetail</td>
<td>See page 53.</td>
</tr>
</tbody>
</table>
| CPayExtensions | The data in an extension to the capture request payload must be financial and should be important for the processing of a capture message by the Payment Gateway, the financial network, or the issuer.

Note: The data in this extension applies to an individual item in the capture request; data related to the entire capture request message should be placed in an extension to CapReqData.

Table 55: CapPayload

```
1349 CapPayload ::= SEQUENCE {
1350     capDate Date,
1351     capReqAmt CurrencyAmount,
1352     authReqItem [0] AuthReqItem OPTIONAL,
1354     saleDetail [2] SaleDetail OPTIONAL,
1356 }
```

Continued on next page
Capture Pair, continued

CapPayload (continued)

998 AuthReqItem ::= SEQUENCE {
999   authTags        AuthTags,
1000  checkDigests    [0] CheckDigests OPTIONAL,
1001  authReqPayload  AuthReqPayload
1002 }

1126 AuthResPayload ::= SEQUENCE {
1127   authHeader     AuthHeader,
1128   capResPayload  CapResPayload OPTIONAL,
1129   aRsExtensions  [0] MsgExtensions {{ARsExtensionsIOS}} OPTIONAL
1130 }

1920 SaleDetail ::= SEQUENCE {
1921   batchID                  [ 0] BatchID OPTIONAL,
1922   batchSequenceNum         [ 1] BatchSequenceNum OPTIONAL,
1923   payRecurInd              [ 2] PayRecurInd OPTIONAL,
1924   merOrderNum              [ 3] MerOrderNum OPTIONAL,
1925   authCharInd              [ 4] AuthCharInd OPTIONAL,
1927   commercialCardData       [ 6] CommercialCardData OPTIONAL,
1928   orderSummary             [ 7] EXPLICIT SETString { ub-summary } OPTIONAL,
1929   customerReferenceNumber  [ 8] EXPLICIT SETString { ub-reference } OPTIONAL,
1930   customerServicePhone     [ 9] EXPLICIT Phone OPTIONAL,
1931   okToPrintPhoneInd        [10] BOOLEAN DEFAULT TRUE,
1933 }

Continued on next page
**Capture Pair, continued**

<table>
<thead>
<tr>
<th>CapRes</th>
<th>Enc(P, M, CapResData)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapResData</td>
<td>{CapRRTags, [BrandCRLIdentifier], [PEThumb], [BatchStatusSeq], CapResItemSeq, [CRsExtensions]}</td>
</tr>
<tr>
<td>CapRRTags</td>
<td>RRTags (see page 34); copied from CapReq</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>List of current CRLs for all CAs under a Brand CA. See page 151.</td>
</tr>
<tr>
<td>PEThumb</td>
<td>Thumbprint of Payment Gateway certificate provided if CapReqData.MThumbs indicates Merchant needs one</td>
</tr>
<tr>
<td>BatchStatusSeq</td>
<td>{BatchStatus +}</td>
</tr>
<tr>
<td>CapResItemSeq</td>
<td>{CapResItem +}</td>
</tr>
</tbody>
</table>
| CRsExtensions | The data in an extension to the capture response must be financial and should be important for the processing of the capture response or a subsequent capture reversal or credit request by the Payment Gateway, the financial network, or the issuer.  
   Note: The data in this extension applies to every item in the capture response; data related to a specific item should be placed in an extension to CapResPayload. |
| BatchStatus | See page 47. |
| CapResItem | {TransIDs, AuthRRPID, CapResPayload} |
| TransIDs | Copied from corresponding CapReq. |
| AuthRRPID | The RRPID that appeared in the corresponding AuthReq or AuthRevReq; copied from corresponding CapReq. |
| CapResPayload | See page 119. |

Table 56: CapRes

Continued on next page
Capture Pair, continued

CapRes (continued)

1360 CapRes ::= Enc { P, M, CapResData }

1365 CapResData ::= SEQUENCE {
1366  capRRTags CapRRTags,
1367  brandCRLIdentifier [0] EXPLICIT BrandCRLIdentifier OPTIONAL,
1368  pThumb [1] EXPLICIT CertThumb OPTIONAL,
1369  batchStatusSeq [2] BatchStatusSeq OPTIONAL,
1370  capResItemSeq CapResItemSeq,
1372 }

1373 CapRRTags ::= RRTags

191 BrandCRLIdentifier ::= SIGNED {
192  EncodedBrandCRLID
193 } (CONSTRAINED BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- })

1716 BatchStatusSeq ::= SEQUENCE OF BatchStatus

1766 CapResItemSeq ::= SEQUENCE SIZE(1..MAX) OF CapResItem

1718 BatchStatus ::= SEQUENCE {
1719  openDateTime Date,
1720  closedWhen [0] ClosedWhen OPTIONAL,
1721  batchDetails BatchDetails,
1722  batchExtensions [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL
1723 }

1378 CapResItem ::= SEQUENCE {
1379  transIDs TransIDs,
1380  authRRPID RRPID,
1381  capResPayload CapResPayload
1382 }

337 TransIDs ::= SEQUENCE {
338  lid-C LocalID,
339  lid-M [0] LocalID OPTIONAL,
340  xid XID,
341  pReqDate Date,
342  paySysID [1] PaySysID OPTIONAL,
343  language Language -- Cardholder requested session language
344 }

1384 CapResPayload ::= SEQUENCE {
1385  capCode CapCode,
1386  capAmt CurrencyAmount,
1387  batchID [0] BatchID OPTIONAL,
1388  batchSequenceNum [1] BatchSequenceNum OPTIONAL,
1390 }

Continued on next page
**Capture Pair, continued**

<table>
<thead>
<tr>
<th>CapResPayload</th>
<th>{CapCode, CapAmt, [BatchID], [BatchSequenceNum], [CRsPayExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CapCode</strong></td>
<td>Enumerated code indicating status of capture.</td>
</tr>
<tr>
<td><strong>CapAmt</strong></td>
<td>Copied from corresponding CapReq</td>
</tr>
<tr>
<td><strong>BatchID</strong></td>
<td>Identification of the settlement batch for merchant-acquirer</td>
</tr>
<tr>
<td></td>
<td>accounting; copied from corresponding CapReq</td>
</tr>
<tr>
<td><strong>BatchSequenceNum</strong></td>
<td>The sequence number of this item within the batch; copied from</td>
</tr>
<tr>
<td></td>
<td>corresponding CapReq</td>
</tr>
<tr>
<td><strong>CRsPayExtensions</strong></td>
<td>The data in an extension to the capture response payload must</td>
</tr>
<tr>
<td></td>
<td>be financial and should be important for the processing of the</td>
</tr>
<tr>
<td></td>
<td>capture response or a subsequent capture reversal or credit</td>
</tr>
<tr>
<td></td>
<td>request by the Payment Gateway, the financial network, or the</td>
</tr>
<tr>
<td></td>
<td>issuer.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The data in this extension applies to an individual item</td>
</tr>
<tr>
<td></td>
<td>in the capture response; data related to the entire capture</td>
</tr>
<tr>
<td></td>
<td>response message should be placed in an extension to CapResData.</td>
</tr>
</tbody>
</table>

Table 57: CapResPayload

1384 CapResPayload ::= SEQUENCE {
1385    capCode CapCode,
1386    capAmt CurrencyAmount,
1387    batchID [0] BatchID OPTIONAL,
1388    batchSequenceNum [1] BatchSequenceNum OPTIONAL,
1390 }

Continued on next page
Capture Pair, continued

CapResPayload (continued)

1394  CapCode ::=ENUMERATED {
1395       success             (0),
1396       unspecifiedFailure (1),
1397       duplicateRequest   (2),
1398       authExpired        (3),
1399       authDataMissing    (4),
1400       invalidAuthData   (5),
1401       capTokenMissing    (6),
1402       invalidCapToken    (7),
1403       batchUnknown       (8),
1404       batchClosed        (9),
1405       unknownXID         (10),
1406       unknownLID         (11)
1407      }

1812  BatchID ::= INTEGER (0..MAX)

1814  BatchSequenceNum ::= INTEGER (1..MAX)
Capture Reversal Or Credit

Why group these messages?

An intermediate, syntactic abstraction exists because Capture Reversal and Credit messages are formally identical.

Payment card brand rules will establish minimum storage times for authorization data and Capture Tokens, but the protocol must assume they will not be stored forever.

Organization

This topic includes:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapRevOrCredReqData</td>
<td>122</td>
</tr>
<tr>
<td>CapRevOrCredResData</td>
<td>125</td>
</tr>
<tr>
<td>CapRevOrCredResPayload</td>
<td>127</td>
</tr>
</tbody>
</table>

Continued on next page
Capture Reversal Or Credit, continued

CapRevOrCredReqData

<table>
<thead>
<tr>
<th>CapRevOrCredReqData</th>
<th>{CapRevOrCredRRTags, [MThumbs], CapRevOrCredReqItemSeq, [CRvRqExtensions]}</th>
</tr>
</thead>
</table>
| CapRevOrCredRRTags          | RRTags, see page 34.  
                              | Fresh RRPID and Date for this pair |
| MThumbs                     | Thumbprints of certificates, CRLs, and Brand CRL  
                              | Identifiers currently held Merchant’s cache |
| CapRevOrCredReqItemSeq      | {CapRevOrCredReqItem +}  
                              | One or more CapRevOrCredReqItem in an ordered array |
| CRvRqExtensions             | The data in an extension to the capture reversal or credit request must be financial and should be important for the processing of a capture reversal or credit by the Payment Gateway, the financial network, or the issuer.  
                              | Note: The data in this extension applies to every item in the capture reversal or credit request; data related to a specific item should be placed in an extension to CapRevOrCredReqItem. |
| CapRevOrCredReqItem         | {TransIDs, AuthRRPID, CapPayload,  
                              | [NewBatchID], CapRevOrCredReqDate,  
                              | [CapRevOrCredReqAmt], NewAccountInd,  
                              | [CRvRqItemExtensions]} |
| TransIDs                    | Copied from the corresponding CapRes (see page 117).  
                              | Required if the corresponding CapToken is not present or does not contain the relevant authorization request data. |

Table 58: CapRevOrCredReqData

Continued on next page
Capture Reversal Or Credit, continued

CapRevOrCredReqData (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthRRPID</td>
<td>The RRPID that appeared in the corresponding AuthReq or AuthRevReq.</td>
</tr>
<tr>
<td>CapPayload</td>
<td>See page 115.</td>
</tr>
<tr>
<td>NewBatchID</td>
<td>This field specifies a new batch identifier; it is used for reversal requests for items submitted in a batch that has subsequently been closed. The BatchID in CapPayload identifies the original batch.</td>
</tr>
<tr>
<td>CapRevOrCredReqDate</td>
<td>The date the request is submitted.</td>
</tr>
<tr>
<td>CapRevOrCredReqAmt</td>
<td>In credit requests, the amount of credit requested, which may differ from AuthAmt in CapToken and CapReqAmt in CapPayload.</td>
</tr>
<tr>
<td>NewAccountInd</td>
<td>Indicates that a new account number is specified in PANToken; when this field is set, the new account number overrides the account information in the CaptureToken or authorization data retained by the acquirer. Use of this field is subject to payment card brand and acquirer policies.</td>
</tr>
<tr>
<td>CRvRqItemExtensions</td>
<td>The data in an extension to the capture reversal or credit request item must be financial and should be important for the processing of a capture reversal or credit by the Payment Gateway, the financial network or the issuer. Note: The data in this extension applies to an individual item in the capture reversal or credit request; data related to the entire capture reversal or credit request message should be placed in an extension to CapRevOrCredReqData.</td>
</tr>
</tbody>
</table>

Table 58: CapRevOrCredReqData, continued

Continued on next page
Capture Reversal Or Credit, continued

CapRevOrCredReqData (continued)

1411 CapRevOrCredReqData ::= SEQUENCE {
1412     capRevOrCredRRTags RRTags,
1413     mThumbs [0] EXPLICIT Thumbs OPTIONAL,
1414     capRevOrCredReqItemSeq CapRevOrCredReqItemSeq,
1415     cRvRqExtensions [1] MsgExtensions {{CRvRqExtensionsIOS}} OPTIONAL
1416 }

1420 CapRevOrCredReqItemSeq ::= SEQUENCE SIZE(1..MAX) OF CapRevOrCredReqItem

1422 CapRevOrCredReqItem ::= SEQUENCE {
1423     transIDs TransIDs,
1424     authRRPID RRPID,
1425     capPayload CapPayload,
1426     newBatchID [0] BatchID OPTIONAL,
1427     capRevOrCredReqDate Date,
1428     capRevOrCredReqAmt [1] CurrencyAmount OPTIONAL,
1429     newAccountInd BOOLEAN DEFAULT FALSE,
1430     cRvRqItemExtensions [2] MsgExtensions {{CRvRqItemExtensionsIOS}} OPTIONAL
1431 }

337 TransIDs ::= SEQUENCE {
338     lid-C LocalID,
339     lid-M [0] LocalID OPTIONAL,
340     xid XID,
341     pReqDate Date,
342     paySysID [1] PaySysID OPTIONAL,
343     language Language -- Cardholder requested session language
344 }

1349 CapPayload ::= SEQUENCE {
1350     capDate Date,
1351     capReqAmt CurrencyAmount,
1352     authReqItem [0] AuthReqItem OPTIONAL,
1354     saleDetail [2] SaleDetail OPTIONAL,
1356 }

Continued on next page
Capture Reversal Or Credit, continued

CapRevOrCredResData

<table>
<thead>
<tr>
<th>CapRevOrCredResData</th>
<th>{CapRevOrCredRRTags, [BrandCRLIdentifier], [PEThumb], [BatchStatusSeq], CapRevOrCredResItemSeq, [CRvRsExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapRevOrCredRRTags</td>
<td>RRTags (see page 34); copied CapRevOrCredRRTags from corresponding CapRevOrCredReqData</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>List of current CRLs for all CAs under a Brand CA. See page 151.</td>
</tr>
<tr>
<td>PEThumb</td>
<td>Thumbprint of Payment Gateway certificate provided if CapRevOrCredReq.MThumbs indicates Merchant needs one</td>
</tr>
<tr>
<td>BatchStatusSeq</td>
<td>{BatchStatus +}</td>
</tr>
<tr>
<td>CapRevOrCredResItemSeq</td>
<td>{CapRevOrCredResItem +} One or more CapRevOrCredResItem in an ordered array</td>
</tr>
<tr>
<td>CRvRsExtensions</td>
<td>The data in an extension to the capture reversal or credit response must be financial and should be important for the processing of the capture reversal or credit response by the Payment Gateway, the financial network, or the issuer. Note: The data in this extension applies to every item in the capture reversal or credit response; data related to a specific item should be placed in an extension to CapRevOrCredResPayload.</td>
</tr>
<tr>
<td>BatchStatus</td>
<td>See page 47.</td>
</tr>
<tr>
<td>CapRevOrCredResItem</td>
<td>{TransIDs, AuthRRPID, CapRevOrCredResPayload}</td>
</tr>
<tr>
<td>TransIDs</td>
<td>Copied from corresponding CapRevOrCredReqData.AuthReqData.AuthTags</td>
</tr>
<tr>
<td>AuthRRPID</td>
<td>The RRPID that appeared in the corresponding AuthReq or AuthRevReq</td>
</tr>
<tr>
<td>CapRevOrCredResPayload</td>
<td>See page 127.</td>
</tr>
</tbody>
</table>

Table 59: CapRevOrCredResData

Continued on next page
Capture Reversal Or Credit, continued

**CapRevOrCredResData (continued)**

```
1435 CapRevOrCredResData ::= SEQUENCE {
1436   capRevOrCredRRTags  RRTags,
1437   brandCRLIdentifier  [0] EXPLICIT BrandCRLIdentifier  OPTIONAL,
1438   peThumb             [1] EXPLICIT CertThumb  OPTIONAL,
1439   batchStatusSeq      [2] BatchStatusSeq  OPTIONAL,
1440   capRevOrCredResItemSeq  CapRevOrCredResItemSeq,
1442 }
191 BrandCRLIdentifier ::= SIGNED {
192   EncodedBrandCRLID
193 } ( CONstrained BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } )
1716 BatchStatusSeq ::= SEQUENCE OF BatchStatus
1446 CapRevOrCredResItemSeq ::= SEQUENCE SIZE(1..MAX) OF CapRevOrCredResItem
1718 BatchStatus ::= SEQUENCE {
1719   openDateTime     Date,
1720   closedWhen       [0] ClosedWhen OPTIONAL,
1721   batchDetails     BatchDetails,
1722   batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL
1723 }
1448 CapRevOrCredResItem ::= SEQUENCE {
1449   transIDs     TransIDs,
1450   authRRPID    RRPID,
1451   capRevOrCredResPayload  CapRevOrCredResPayload
1452 }
337 TransIDs ::= SEQUENCE {
338   lid-C      LocalID,
339   lid-M      [0] LocalID OPTIONAL,
340   xid        XID,
341   pReqDate   Date,
342   paySysID   [1] PaySysID OPTIONAL,
343   language   Language          -- Cardholder requested session language
344 }
1454 CapRevOrCredResPayload ::= SEQUENCE {
1455   capRevOrCredCode    CapRevOrCredCode,
1456   capRevOrCredActualAmt  CurrencyAmount,
1457   batchID      [0] BatchID OPTIONAL,
1458   batchSequenceNum [1] BatchSequenceNum OPTIONAL,
1460 }
```

*Continued on next page*
Capture Reversal Or Credit, continued

CapRevOrCredResPayload

<table>
<thead>
<tr>
<th>CapRevOrCredResPayload</th>
<th>{CapRevOrCredCode, CapRevOrCredActualAmt, [BatchID], [BatchSequenceNum], [CRvRsPayExtensions]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapRevOrCredCode</td>
<td>Enumerated code indicating capture reversal or credit status</td>
</tr>
<tr>
<td>CapRevOrCredActualAmt</td>
<td>Copied from corresponding CapRevOrCredReqItem</td>
</tr>
<tr>
<td>BatchID</td>
<td>Identification of the settlement batch for merchant-acquirer accounting</td>
</tr>
<tr>
<td>BatchSequenceNum</td>
<td>The sequence number of this item within the batch</td>
</tr>
<tr>
<td>CRvRsPayExtensions</td>
<td>The data in an extension to the capture reversal or credit response must be financial and should be important for the processing of the capture reversal or credit response. Note: The data in this extension applies to an individual item in the capture reversal or credit response; data related to the entire capture reversal or credit response message should be placed in an extension to CapRevOrCredResData.</td>
</tr>
</tbody>
</table>

Table 60: CapRevOrCredResPayload

1454 CapRevOrCredResPayload ::= SEQUENCE {
1455   capRevOrCredCode   CapRevOrCredCode,
1456   capRevOrCredActualAmt CurrencyAmount,
1457   batchID            [0] BatchID OPTIONAL,
1458   batchSequenceNum   [1] BatchSequenceNum OPTIONAL,
1460 }

Continued on next page
Capture Reversal Or Credit, continued

CapRevOrCredResPayload (continued)

1464 CapRevOrCredCode ::= ENUMERATED {
1465    success             (0),
1466    unspecifiedFailure (1),
1467    duplicateRequest   (2),
1468    originalProcessed  (3),
1469    originalNotFound   (4),
1470    capPurged          (5),
1471    capDataMismatch    (6),
1472    missingCapData     (7),
1473    missingCapToken    (8),
1474    invalidCapToken    (9),
1475    batchUnknown       (10),
1476    batchClosed        (11)
1477 }

1812 BatchID ::= INTEGER (0..MAX)

1814 BatchSequenceNum ::= INTEGER (1..MAX)
Capture Reversal Pair

CapRevReq

<table>
<thead>
<tr>
<th>CapRevReq</th>
<th>&lt; EncB(M, P, CapRevData, CapTokenSeq), EncBX(M, P, CapRevData, CapTokenSeq, PANToken)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CapTokenSeq is external “baggage”.</td>
</tr>
<tr>
<td></td>
<td>If PANToken is included, it must correspond to a single entry in CapRevData.CapRevOrCredReqItemSeq and a single CapToken in CapTokenSeq</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CapRevData</th>
<th>CapRevOrCredReqData; see page 122.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapTokenSeq</td>
<td>{{CapToken} +}</td>
</tr>
<tr>
<td></td>
<td>One or more CapTokens, in ordered one-to-one correspondence with CapRevOrCredReqItem sequence in CapRevOrCredReqData.CapRevOrCredReqItemSeq.</td>
</tr>
<tr>
<td></td>
<td>Note: Any CapToken may be omitted; that is, may be NULL.</td>
</tr>
<tr>
<td>PANToken</td>
<td>See page 46</td>
</tr>
<tr>
<td>CapToken</td>
<td>Copied from corresponding AuthRes or AuthRevRes</td>
</tr>
</tbody>
</table>

Table 61: CapRevReq

1481 CapRevReq ::= CHOICE {
1482    encB  [0] EXPLICIT EncB { M, P, CapRevData, CapTokenSeq },
1483    encBX [1] EXPLICIT EncBX { M, P, CapRevData, CapTokenSeq, PANToken }
1484 }

1501 CapRevData ::= [0] EXPLICIT CapRevOrCredReqData

1841 CapTokenSeq ::= SEQUENCE SIZE(1..MAX) OF CapToken

Continued on next page
Capture Reversal Pair, continued

CapRevReq (continued)

314 PANToken ::= SEQUENCE {
315    pan         PAN,
316    cardExpiry  CardExpiry,
317    exNonce     Nonce
318 }

1816 CapToken ::= CHOICE {
1817    encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
1818    enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
1819    null  [2] EXPLICIT NULL
1820 }

Continued on next page
Capture Reversal Pair, continued

|-------------------|--------------------------|

Table 62: CapRevRes


1508 CapRevResData ::= [0] EXPLICIT CapRevOrCredResData
Credit Pair

<table>
<thead>
<tr>
<th>CredReq</th>
<th>&lt; EncB(M, P, CredReqData, CapTokenSeq), EncBX(M, P, CredReqData, CapTokenSeq, PANToken) &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapTokenSeq</td>
<td>is external “baggage”.</td>
</tr>
<tr>
<td>CredReqData</td>
<td>CapRevOrCredReqData: see page 122.</td>
</tr>
<tr>
<td>CapTokenSeq</td>
<td>{[CapToken] +}</td>
</tr>
<tr>
<td>One or more CapTokens in ordered one-to-one correspondence with CapRevOrCredReqItem sequence in CapRevOrCredReqData.CapRevOrCredReqItemSeq.</td>
<td></td>
</tr>
<tr>
<td>Note: Any CapToken may be omitted; that is, may be NULL.</td>
<td></td>
</tr>
<tr>
<td>PANToken</td>
<td>See page 46</td>
</tr>
<tr>
<td>CapToken</td>
<td>Copied from corresponding AuthRes or AuthRevRes.</td>
</tr>
</tbody>
</table>

Table 63: CredReq

```
1512 CredReq ::= CHOICE {
1513   encB [0] EXPLICIT EncB { M, P, CredReqData, CapTokenSeq },
1514   encBX [1] EXPLICIT EncBX { M, P, CredReqData, CapTokenSeq, PANToken } 
1515 }

1532 CredReqData ::= [1] EXPLICIT CapRevOrCredReqData

1841 CapTokenSeq ::= SEQUENCE SIZE(1..MAX) OF CapToken

314 PANToken ::= SEQUENCE {
315   pan PAN,
316   cardExpiry CardExpiry,
317   exNonce Nonce
318 }
```

Continued on next page
Credit Pair, continued

CredReq (continued)

1816 CapToken ::= CHOICE {
1817   encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
1818   enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
1819   null  [2] EXPLICIT NULL
1820 }

Continued on next page
Credit Pair, continued

CredRes

<table>
<thead>
<tr>
<th>CredRes</th>
<th>Enc(P, M, CredResData)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CredResData</td>
<td>CapRevOrCredResData: see page 125.</td>
</tr>
</tbody>
</table>

Table 64: CredRes

1534 CredRes ::= Enc { P, M, CredResData }

Credit Reversal Pair

CredRevReq

<table>
<thead>
<tr>
<th>CredRevReq</th>
<th>&lt; EncB(M, P, CredRevReqData, CapTokenSeq), EncBX(M, P, CredRevReqData, CapTokenSeq, PANToken) &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapTokenSeq</td>
<td>is external “baggage”. If PANToken is included, it must correspond to a single entry in CredRevReqData.CredRevReqSeq and a single CapToken in CapTokenSeq.</td>
</tr>
</tbody>
</table>

| CredRevReqData | CapRevOrCredReqData: see page 122. |
| CapTokenSeq    | {[CapToken] +} One or more CapTokens, in ordered one-to-one correspondence with CredRevReqItem in CapRevOrCredReqData.CapRevOrCredReqItemSeq. |
| PANToken       | See page 46 |
| CapToken       | Copied from corresponding AuthRes or AuthRevRes. |

Table 65: CredRevReq

1543 CredRevReq ::= CHOICE {
    1544     encB  [0] EXPLICIT EncB { M, P, CredRevReqData, CapTokenSeq },
    1545     encBX [1] EXPLICIT EncBX { M, P, CredRevReqData, CapTokenSeq, PANToken }
    1546 }

1563 CredRevReqData ::= [2] EXPLICIT CapRevOrCredReqData

1841 CapTokenSeq ::= SEQUENCE SIZE(1..MAX) OF CapToken

Continued on next page
Credit Reversal Pair, continued

CredRevReq (continued)

314 PANToken ::= SEQUENCE {
315   pan             PAN,
316   cardExpire      CardExpire,
317   exNonce         Nonce
318 }

1816 CapToken ::= CHOICE {
1817   encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
1818   enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
1819   null  [2] EXPLICIT NULL
1820 }

Continued on next page
**Credit Reversal Pair**, continued

---

CredRevRes

|---------------------------|----------------------------|

Table 66: CredRevRes


Chapter 5
Payment Gateway Certificate Request and Batch Administration

Overview

Organization  Chapter 5 describes two message pairs:
- Payment Gateway Certificate Request Pair
- Batch Administration Pair
### Payment Gateway Certificate Request Pair

**PCertReq**

Merchant uses this message pair to request fresh key-exchange certificates from Payment Gateway.

<table>
<thead>
<tr>
<th>PCertReq</th>
<th>S(M, PCertReqData)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PCertReqData</strong></td>
<td>{PCertRRTags, [MThumbs], BrandAndBINSeq, [PCRqExtensions]}</td>
</tr>
<tr>
<td><strong>PCertRRTags</strong></td>
<td>RRTags, see page 34.</td>
</tr>
<tr>
<td><strong>Fresh RRPID</strong></td>
<td>For this PCertReq, Merchant-supplied MerTermIDs, and current date</td>
</tr>
<tr>
<td><strong>MThumbs</strong></td>
<td>Thumbprints of Payment Gateway certificates currently in Merchant cache</td>
</tr>
<tr>
<td><strong>BrandAndBINSeq</strong></td>
<td>{BrandAndBIN +}</td>
</tr>
<tr>
<td><strong>Merchant requests</strong></td>
<td>Payment Gateway certificates for these payment card brands if the thumbprint of the current certificate does not appear in MThumbs.</td>
</tr>
<tr>
<td><strong>PCRqExtensions</strong></td>
<td>Note: The Payment Gateway certificate request is not encrypted so this extension must not contain confidential information.</td>
</tr>
<tr>
<td><strong>BrandAndBIN</strong></td>
<td>{BrandID, [BIN]}</td>
</tr>
<tr>
<td><strong>BrandID</strong></td>
<td>Payment card brand (without product type)</td>
</tr>
<tr>
<td><strong>BIN</strong></td>
<td>Bank Identification Number for the processing of Merchant’s transactions at the Payment Gateway</td>
</tr>
</tbody>
</table>

**Table 67: PCertReq**

```
1574 PCertReq ::= S { M, PCertReqData }
1576 PCertReqData ::= SEQUENCE {
1577   pCertRRTags   RRTags, 
1578   mThumbs       [0] EXPLICIT Thumbs  OPTIONAL, 
1579   brandAndBINSeq BrandAndBINSeq, 
1580   pcRqExtensions [1] MsgExtensions {{PCRqExtensionsIOS}}  OPTIONAL 
1581 }
1585 BrandAndBINSeq ::= SEQUENCE SIZE(1..MAX) OF BrandAndBIN
1587 BrandAndBIN ::= SEQUENCE {
1588   brandID BrandID, 
1589   bin BIN OPTIONAL 
1590 }
232 BrandID ::= SETString { ub-BrandID }
250 BIN ::= NumericString (SIZE(6))  -- Bank identification number
```

*Continued on next page*
Payment Gateway Certificate Request Pair, continued

PCertRes

<table>
<thead>
<tr>
<th>PCertRes</th>
<th>S(P, PCertResTBS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCertResTBS</td>
<td>{PCertRRTags, [BrandCRLIdentifierSeq], PCertResItemSeq, [PCRsExtensions]}</td>
</tr>
<tr>
<td>PCertRRTags</td>
<td>RRTags (see page 34); copied from PCertReq</td>
</tr>
<tr>
<td>BrandCRLIdentifierSeq</td>
<td>{BrandCRLIdentifier +}</td>
</tr>
<tr>
<td>PCertResItemSeq</td>
<td>{PCertResItem +}</td>
</tr>
<tr>
<td>PCRsExtensions</td>
<td>Note: The Payment Gateway certificate response is not encrypted so this extension must not contain confidential information.</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>List of current CRLs for all CAs under a Brand CA. See page 151.</td>
</tr>
<tr>
<td>PCertResItem</td>
<td>{PCertCode, [CertThumb]}</td>
</tr>
<tr>
<td>PCertCode</td>
<td>Enumerated code indicating result of PCertReq</td>
</tr>
<tr>
<td>CertThumb</td>
<td>Thumbprint of returned certificate</td>
</tr>
</tbody>
</table>

Table 68: PCertRes

1592 PCertRes ::= S { P, PCertResTBS }
1594 PCertResTBS ::= SEQUENCE {
1595     pCertRRTags RRTags,  
1596     pCertResItemSeq PCertResItemSeq,  
1597     brandCRLIdentifierSeq [0] BrandCRLIdentifierSeq OPTIONAL,  
1598     pcRsExtensions [1] MsgExtensions {{PCRsExtensionsIOS}} OPTIONAL  
1599}
1610 PCertCode ::= ENUMERATED {
1611     success (0),  
1612     unspecifiedFailure (1),  
1613     brandNotSupported (2),  
1614     unknownBIN (3)  
1615}
1617 BrandCRLIdentifierSeq ::= SEQUENCE SIZE(1..MAX) OF [0] EXPLICIT 
BrandCRLIdentifier
191 BrandCRLIdentifier ::= SIGNED {
192     EncodedBrandCRLID  
193 } { CONstrained BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } }
Batch Administration Pair

Merchant sends these to Payment Gateway to administer batches of Capture Tokens.

<table>
<thead>
<tr>
<th>BatchAdminReq</th>
<th>Enc(M, P, BatchAdminReqData)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchAdminReqData</td>
<td>{BatchAdminRRTags, [BatchID], [BrandAndBINSeq], [BatchOperation], ReturnBatchSummaryInd, [ReturnTransactionDetail], [BatchStatus], [TransDetails], [BARqExtensions]}</td>
</tr>
<tr>
<td>BatchAdminRRTags</td>
<td>RRTags, see page 34. Fresh RRPPID and Date</td>
</tr>
<tr>
<td>BatchID</td>
<td>Identification of the settlement batch for merchant-acquirer accounting</td>
</tr>
<tr>
<td>BrandAndBINSeq</td>
<td>{BrandAndBIN +}</td>
</tr>
<tr>
<td>BatchOperation</td>
<td>Enumerated value indicating the action to be performed on the batch.</td>
</tr>
<tr>
<td>ReturnBatchSummaryInd</td>
<td>Indicates batch summary data is to be returned in BatchAdminRes</td>
</tr>
<tr>
<td>ReturnTransactionDetail</td>
<td>{StartingPoint, MaximumItems, ErrorsOnlyInd, [BrandID]} If BrandID is specified, only items for that payment card brand are returned.</td>
</tr>
<tr>
<td>BatchStatus</td>
<td>See page 47.</td>
</tr>
<tr>
<td>TransDetails</td>
<td>{NextStartingPoint, TransactionDetailSeq}</td>
</tr>
<tr>
<td>BARqExtensions</td>
<td>The data in an extension to the batch administration message must be financial and should be important for the processing of the batch administration request.</td>
</tr>
</tbody>
</table>

Table 69: BatchAdminReq

Continued on next page
### Batch Administration Pair, continued

**Table 69: BatchAdminReq, continued**

<table>
<thead>
<tr>
<th>BrandAndBIN</th>
<th>(BrandID, [BIN])</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartingPoint</td>
<td>Zero indicates to send detail for the first group of items; otherwise, NextStartingPoint from a prior BatchAdminRes.</td>
</tr>
<tr>
<td>MaximumItems</td>
<td>The maximum number of items to be returned in this group of items.</td>
</tr>
<tr>
<td>ErrorsOnlyInd</td>
<td>Boolean indicating if only items with an error status should be returned.</td>
</tr>
<tr>
<td>BrandID</td>
<td>Payment card brand (without product type)</td>
</tr>
<tr>
<td>NextStartingPoint</td>
<td>Zero indicates that this is the last group of items; otherwise, an opaque value used to identify the starting point of the next group of items.</td>
</tr>
<tr>
<td>TransactionDetailSeq</td>
<td>{TransactionDetail +}</td>
</tr>
<tr>
<td>BIN</td>
<td>Bank Identification Number for the processing of Merchant’s transactions at the Acquirer</td>
</tr>
<tr>
<td>TransactionDetail</td>
<td>See page 50.</td>
</tr>
</tbody>
</table>

1621 BatchAdminReq ::= Enc { M, P, BatchAdminReqData }

1626 BatchAdminReqData ::= SEQUENCE {
    batchAdminRRTags          RRTags,
    batchID                  [0] BatchID  OPTIONAL,
    brandAndBINSeq           [1] BrandAndBINSeq  OPTIONAL,
    batchOperation           [2] BatchOperation  OPTIONAL,
    returnBatchSummaryInd    BOOLEAN DEFAULT FALSE,
    returnTransactionDetail  [3] ReturnTransactionDetail  OPTIONAL,
    batchStatus              [4] BatchStatus  OPTIONAL,
    transDetails             [5] TransDetails  OPTIONAL,
    baRqExtensions           [6] MsgExtensions {{BARqExtensionsIOS}} OPTIONAL
}

1812 BatchID ::= INTEGER (0..MAX)

1585 BrandAndBINSeq ::= SEQUENCE SIZE(1..MAX) OF BrandAndBIN

1640 BatchOperation ::= ENUMERATED {
    open   (0),
    purge  (1),
    close  (2)
}

**Continued on next page**
Batch Administration Pair, continued

**BatchAdminReq** (continued)

```
1646 ReturnTransactionDetail ::= SEQUENCE {
1647    startingPoint  INTEGER (MIN..MAX),
1648    maximumItems   INTEGER (1..MAX),
1649    errorsOnlyInd  BOOLEAN DEFAULT FALSE,
1650    brandID        [0] EXPLICIT BrandID  OPTIONAL
1651 }

1718 BatchStatus ::= SEQUENCE {
1719    openDateTime     Date,
1720    closedWhen       [0] ClosedWhen OPTIONAL,
1721    batchDetails     BatchDetails,
1722    batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL
1723 }

1653 TransDetails ::= SEQUENCE {
1654    nextStartingPoint     INTEGER (MIN..MAX),
1655    transactionDetailSeq  TransactionDetailSeq
1656 }

1587 BrandAndBIN ::= SEQUENCE {
1588    brandID  BrandID,
1589    bin      BIN  OPTIONAL
1590 }

232 BrandID ::= SETString { ub-BrandID }

1749 TransactionDetailSeq ::= SEQUENCE OF TransactionDetail

250 BIN ::= NumericString (SIZE(6))            -- Bank identification number

1751 TransactionDetail ::= SEQUENCE {
1752    transIDs      TransIDs,
1753    authRRPID     RRPID,
1754    brandID      BrandID,
1755    batchSequenceNum BatchSequenceNum,
1756    reimbursementID ReimbursementID  OPTIONAL,
1757    transactionAmt CurrencyAmount,
1758    transactionAmtType AmountType,
1759    transactionStatus [0] TransactionStatus OPTIONAL,
1760    transExtensions [1] MsgExtensions {{TransExtensionsIOS}} OPTIONAL
1761 }
```

Continued on next page
Batch Administration Pair, continued

### BatchAdminRes

<table>
<thead>
<tr>
<th>BatchAdminRes</th>
<th>Enc(P, M, BatchAdminResData)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchAdminResData</td>
<td>{BatchAdminTags, BatchID, [BAStatus], [BatchStatus], [TransmissionStatus], [SettlementInfo], [TransDetails], [BARsExtensions]}</td>
</tr>
<tr>
<td>BatchAdminTags</td>
<td>RRTags (see page 34); copied from prior BatchAdminReq</td>
</tr>
<tr>
<td>BatchID</td>
<td>Identification of the settlement batch for merchant-acquirer accounting</td>
</tr>
<tr>
<td>BAStatus</td>
<td>Enumerated code indicating status of batch open</td>
</tr>
<tr>
<td>BatchStatus</td>
<td>See page 47.</td>
</tr>
<tr>
<td>TransmissionStatus</td>
<td>Enumerated value indicating the status of the transmission from the gateway to the next upstream system</td>
</tr>
<tr>
<td>SettlementInfo</td>
<td>{SettlementAmount, SettlementType, SettlementAccount, SettlementDepositDate}</td>
</tr>
<tr>
<td>TransDetails</td>
<td>{NextStartingPoint, TransactionDetailSeq}</td>
</tr>
<tr>
<td>BARsExtensions</td>
<td>The data in an extension to the batch administration response message must be financial and should be important for the processing of the batch administration request. Note: Information regarding the processing of the request itself should appear in an extension to BatchAdminResData; information regarding the status of a batch should appear in an extension to BatchStatus; information regarding detail for an item within the capture batch should appear in an extension to TransactionDetail.</td>
</tr>
<tr>
<td>SettlementAmount</td>
<td>The net settlement amount to the Merchant’s account</td>
</tr>
<tr>
<td>SettlementType</td>
<td>Enumerated code indicating the type of amount</td>
</tr>
<tr>
<td>SettlementAccount</td>
<td>The Merchant’s account</td>
</tr>
<tr>
<td>SettlementDepositDate</td>
<td>The date that the SettlementAmount will be credited to/debited from the Merchant’s account</td>
</tr>
</tbody>
</table>

Table 70: BatchAdminRes

Continued on next page
Batch Administration Pair, continued

BatchAdminRes (continued)

<table>
<thead>
<tr>
<th>NextStartingPoint</th>
<th>Zero indicates that this is the last group of items; otherwise, an opaque value used to identify the starting point of the next group of items.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransactionDetailSeq</td>
<td>{TransactionDetail +}</td>
</tr>
<tr>
<td>TransactionDetail</td>
<td>See page 50</td>
</tr>
</tbody>
</table>

Table 70: BatchAdminRes, continued

1658 BatchAdminRes ::= Enc { P, M, BatchAdminResData }

1663 BatchAdminResData ::= SEQUENCE {
1664   batchAdminTags    RRTags,  
1665   batchID           BatchID,  
1666   baStatus          BAStatus  OPTIONAL,  
1667   batchStatus       [0] BatchStatus  OPTIONAL,  
1668   transmissionStatus [1] TransmissionStatus  OPTIONAL,  
1669   settlementInfo    [2] SettlementInfo  OPTIONAL,  
1670   transDetails      [3] TransDetails  OPTIONAL,  
1671   baRsExtensions    [4] MsgExtensions {{BARsExtensionsIOS}} OPTIONAL  
1672 }

1812 BatchID ::= INTEGER (0..MAX)

1691 BAStatus ::= ENUMERATED {
1692   success                ( 0),  
1693   unspecifiedFailure     ( 1),  
1694   brandNotSupported      ( 2),  
1695   unknownBIN             ( 3),  
1696   batchIDunavailable     ( 4),  
1697   batchAlreadyOpen       ( 5),  
1698   unknownBatchID        ( 6),  
1699   brandBatchMismatch     ( 7),  
1700   totalsOutOfBalance     ( 8),  
1701   unknownStartingPoint   ( 9),  
1702   stopItemDetail         (10),  
1703   unknownBatchOperation (11)  
1704 }

Continued on next page
Batch Administration Pair, continued

1718 BatchStatus ::= SEQUENCE {
1719   openDateTime     Date,  
1720   closedWhen       [0] ClosedWhen OPTIONAL, 
1721   batchDetails     BatchDetails, 
1722   batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL 
1723 }

1676 TransmissionStatus ::= ENUMERATED {
1677   pending                  (0), 
1678   inProgress               (1), 
1679   batchRejectedByAcquirer  (2), 
1680   completedSuccessfully    (3), 
1681   completedWithItemErrors  (4) 
1682 }

1684 SettlementInfo ::= SEQUENCE {
1685   settlementAmount       CurrencyAmount, 
1686   settlementType         AmountType, 
1687   settlementAccount      SETString { ub-SettlementAccount }, 
1688   settlementDepositDate  Date 
1689 }

1653 TransDetails ::= SEQUENCE {
1654   nextStartingPoint     INTEGER (MIN..MAX), 
1655   transactionDetailSeq  TransactionDetailSeq 
1656 }

1749 TransactionDetailSeq ::= SEQUENCE OF TransactionDetail 

1751 TransactionDetail ::= SEQUENCE {
1752   transIDs            TransIDs, 
1753   authRRPID           RRPID, 
1754   brandID             BrandID, 
1755   batchSequenceNum    BatchSequenceNum, 
1756   reimbursementID     ReimbursementID OPTIONAL, 
1757   transactionAmt      CurrencyAmount, 
1758   transactionAmtType  AmountType, 
1759   transactionStatus   [0] TransactionStatus OPTIONAL, 
1760   transExtensions     [1] MsgExtensions {{TransExtensionsIOS}} OPTIONAL 
1761 }
Chapter 6
Certificate Management Payload Components

Overview

Introduction
Chapter 6 describes the payload components of certificate management messages. Certificate management messages themselves are described in Chapter 7.

Organization
Chapter 6 includes the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDData</td>
<td>148</td>
</tr>
<tr>
<td>RequestType</td>
<td>149</td>
</tr>
<tr>
<td>End Entity and CA Types</td>
<td>150</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>151</td>
</tr>
<tr>
<td>PANData0</td>
<td>153</td>
</tr>
<tr>
<td>AcctData</td>
<td>154</td>
</tr>
<tr>
<td>RegFormOrReferral</td>
<td>155</td>
</tr>
</tbody>
</table>
IDData

IDData

<table>
<thead>
<tr>
<th>IDData</th>
<th>&lt; MerchantAcquirerID, AcquirerID &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantAcquirerID</td>
<td>{MerchantBIN, MerchantID}</td>
</tr>
<tr>
<td>AcquirerID</td>
<td>{AcquirerBIN, [AcquirerBusinessID]}</td>
</tr>
<tr>
<td>MerchantBIN</td>
<td>Bank Identification Number for the processing of Merchant’s transactions at the Acquirer</td>
</tr>
<tr>
<td>MerchantID</td>
<td>Merchant ID assigned by Acquirer</td>
</tr>
<tr>
<td>AcquirerBIN</td>
<td>The Bank Identification Number of this Acquirer</td>
</tr>
<tr>
<td>AcquirerBusinessID</td>
<td>The Business Identification Number of this Acquirer</td>
</tr>
</tbody>
</table>

Table 71: IDData

404 IDData ::= CHOICE { -- Merchants and Acquirers only
405   merchantAcquirerID [0] MerchantAcquirerID,
406     acquirerID [1] AcquirerID
407 }

409 MerchantAcquirerID ::= SEQUENCE {
410   merchantBIN BIN,
411   merchantID MerchantID -- By prior agreement of Merchant/Acquirer
412 }

414 AcquirerID ::= SEQUENCE {
415   acquirerBIN BIN,
416   acquirerBusinessID AcquirerBusinessID OPTIONAL
417 }

294 MerchantID ::= SETString { ub-MerchantID }

419 AcquirerBusinessID ::= NumericString (SIZE(1..ub-acqBusinessID))
RequestType

---

### RequestType

<table>
<thead>
<tr>
<th>RequestType</th>
<th>Enumerated code that indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• whether a Cardholder, Merchant, or Payment Gateway is issuing the request, and</td>
</tr>
<tr>
<td></td>
<td>• whether it is for a new or renewed signature and/or encryption certificate.</td>
</tr>
</tbody>
</table>

**Table 72: RequestType**

```plaintext
RequestType ::= ENUMERATED {
  cardInitialSig    (1),
  -- cardInitialEnc    (2),                                      Reserved
  -- cardInitialBoth   (3),                                      Reserved
  merInitialSig     (4),
  merInitialEnc     (5),
  merInitialBoth     (6),
  pgwyInitialSig    (7),
  pgwyInitialEnc    (8),
  pgwyInitialBoth   (9),
  cardRenewalSig   (10),
  -- cardRenewalEnc   (11),                                    Reserved
  -- cardRenewalBoth (12),                                    Reserved
  merRenewalSig    (13),
  merRenewalEnc    (14),
  merRenewalBoth   (15),
  pgwyRenewalSig   (16),
  pgwyRenewalEnc   (17),
  pgwyRenewalBoth   (18)
}
```
End Entity and CA Types

End Entity types

<table>
<thead>
<tr>
<th>EE</th>
<th>&lt; C, M, P &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For Cardholder, Merchant, or Payment Gateway, respectively. EE is short for “End Entity,” also known as requester.</td>
</tr>
</tbody>
</table>

Table 73: EE (End Entity types)

2947 EE ::= ENTITY-IDENTIFIER -- End Entity
2944 C ::= ENTITY-IDENTIFIER -- Cardholder
2945 M ::= ENTITY-IDENTIFIER -- Merchant
2946 P ::= ENTITY-IDENTIFIER -- Payment Gateway

CA types

<table>
<thead>
<tr>
<th>CA</th>
<th>&lt; CCA, MCA, PCA &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For EE = C, M, P, respectively</td>
</tr>
</tbody>
</table>

Table 74: CA (CA types)

2948 CA ::= ENTITY-IDENTIFIER -- Certifying Authority
**BrandCRLIdentifier**

The **BrandCRLIdentifier** is a signed list of CRL identifiers that indicates all of the current CRLs that the recipient should use to screen certificates.

<table>
<thead>
<tr>
<th><strong>BrandCRLIdentifier</strong></th>
<th>S(CA, UnsignedBrandCRLIdentifier)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UnsignedBrandCRLIdentifier</strong></td>
<td>{Version, SequenceNum, BrandID, NotBefore, NotAfter, [CRLIdentifierSeq], Extensions}</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>The version number, indicating this format of the BrandCRLIdentifier</td>
</tr>
<tr>
<td><strong>SequenceNum</strong></td>
<td>Sequence number that is incremented for each new BrandCRLIdentifier</td>
</tr>
<tr>
<td><strong>BrandID</strong></td>
<td>Identification of the payment card brand whose CRLs are contained in this list</td>
</tr>
<tr>
<td><strong>NotBefore</strong></td>
<td>The beginning of the validity period of the BrandCRLIdentifier</td>
</tr>
<tr>
<td><strong>NotAfter</strong></td>
<td>The end of the validity period of the BrandCRLIdentifier</td>
</tr>
<tr>
<td><strong>CRLIdentifierSeq</strong></td>
<td>{CRLIdentifier +} One or more CRLIdentifiers used to identify the CRLs that the End Entity should be holding</td>
</tr>
<tr>
<td><strong>Extensions</strong></td>
<td>This field incorporates CRL extensions into the BrandCRLIdentifier.</td>
</tr>
<tr>
<td><strong>CRLIdentifier</strong></td>
<td>{IssuerName, CRLNumber}</td>
</tr>
<tr>
<td><strong>NotBefore</strong></td>
<td>The start date of the Brand CRL Identifier’s validity period</td>
</tr>
<tr>
<td><strong>NotAfter</strong></td>
<td>The end date of the Brand CRL Identifier’s validity period</td>
</tr>
<tr>
<td><strong>IssuerName</strong></td>
<td>The Distinguished Name of the CA (issuer) of the CRL</td>
</tr>
<tr>
<td><strong>CRLNumber</strong></td>
<td>The sequence number of the CRL, obtained from the CRLNumber extension</td>
</tr>
</tbody>
</table>

| **Table 75: BrandCRLIdentifier** |

191 BrandCRLIdentifier ::= SIGNED {  
192     EncodedBrandCRLID  
193 } ( CONSTRAINED BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } )  

Continued on next page
BrandCRLIdentifier, continued

BrandCRLIdentifier (continued)

197 UnsignedBrandCRLIdentifier ::= SEQUENCE {
198     version INTEGER { bVer1(0) } (bVer1),
199     sequenceNum INTEGER (0..MAX),
200     brandID BrandID,
201     notBefore GeneralizedTime,
202     notAfter GeneralizedTime,
203     crlIdentifierSeq [0] CRLIdentifierSeq OPTIONAL,
204     bCRLExtensions [1] Extensions OPTIONAL
205 }

232 BrandID ::= SETString { ub-BrandID }

234 CRLIdentifierSeq ::= SEQUENCE OF CRLIdentifier

236 CRLIdentifier ::= SEQUENCE {
237     issuerName Name,  -- CRL issuer Distinguished Name
238     crlNumber INTEGER (0..MAX) -- crlNumber extension sequence number
239 }

2264 CRLNumber ::= INTEGER (0..MAX)
PANData0

PANData0 is formally like PANData (see page 45), except that the third field contains CardSecret, the Cardholder’s proposed half of the shared secret, which will be shared between Cardholder and CCA. The CCA will generate the other half, Nonce-CCA, and both parties will XOR CardSecret and Nonce-CCA to calculate the shared secret, PANSecret.

See also “Optimal Asymmetric Encryption Padding (OAEP)” on page 15. The description of PANData0 begins on page 21.

<table>
<thead>
<tr>
<th>PANData0</th>
<th>{PAN, CardExpiry, CardSecret, EXNonce}</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN</td>
<td>Primary Account Number; typically, the account number on the card</td>
</tr>
<tr>
<td>CardExpiry</td>
<td>Expiration date on the card</td>
</tr>
<tr>
<td>CardSecret</td>
<td>Cardholder’s proposed half of the shared secret, PANSecret. Note: this value is saved for use in generating TransStain (see “PIHead” on page 37).</td>
</tr>
<tr>
<td>EXNonce</td>
<td>A fresh nonce to foil dictionary attacks on PANData0</td>
</tr>
</tbody>
</table>

Table 76: PANData0

307 PANData0 ::= SEQUENCE {
308     pan     PAN,
309     cardExpiry  CardExpiry,
310     cardSecret  Secret,
311     exNonce     Nonce
312 }

298 PAN ::= NumericString (SIZE(1..19))

252 CardExpiry ::= NumericString (SIZE(6)) -- YYYYMM expiration date of card
AcctData

See also Optimal Asymmetric Encryption Padding (OAEP) on page 15. The description of AcctData begins on page 23.

<table>
<thead>
<tr>
<th>AcctData</th>
<th>{AcctIdentification, EXNonce}</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcctIdentification</td>
<td>For a Merchant, this field is unique to the Merchant as defined by the payment card brand and Acquirer. For an Acquirer, this field is unique to the Acquirer as defined by the payment card brand.</td>
</tr>
<tr>
<td>EXNonce</td>
<td>A fresh nonce to foil dictionary attacks on AcctIdentification</td>
</tr>
</tbody>
</table>

Table 77: AcctData

397 AcctData ::= SEQUENCE {
398    acctIdentification         AcctIdentification,
399    exNonce                    Nonce
400 }

402 AcctIdentification ::= VisibleString (SIZE(ub-acctIdentification))
### RegFormOrReferral

<table>
<thead>
<tr>
<th>RegFormOrReferral</th>
<th>&lt; RegFormData, ReferralData &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegFormData</td>
<td>([RegTemplate], PolicyText)</td>
</tr>
<tr>
<td>ReferralData</td>
<td>([Reason], [ReferralURLEq])</td>
</tr>
<tr>
<td>RegTemplate</td>
<td>(RegFormID, [BrandLogoURL], [CardLogoURL], [RegFieldSeq])</td>
</tr>
<tr>
<td>PolicyText</td>
<td>Statement to be displayed along with RegTemplate on requester's system</td>
</tr>
<tr>
<td>Reason</td>
<td>Statement concerning request to be displayed on requester's system</td>
</tr>
<tr>
<td>ReferralURLEq</td>
<td>{ReferralURL +}</td>
</tr>
<tr>
<td>RegFormID</td>
<td>CA-assigned identifier</td>
</tr>
<tr>
<td>BrandLogoURL</td>
<td>The URL for the payment card brand logo</td>
</tr>
<tr>
<td>CardLogoURL</td>
<td>The URL for the financial institution logo</td>
</tr>
<tr>
<td>RegFieldSeq</td>
<td>{RegField +}</td>
</tr>
<tr>
<td>ReferralURL</td>
<td>Uniform Resource Locator of alternate CA for processing of certificate requests for this entity.</td>
</tr>
<tr>
<td>RegField</td>
<td>([FieldId], FieldName, [FieldDesc], [FieldLen], FieldRequired, FieldInvisible)</td>
</tr>
<tr>
<td>FieldID</td>
<td>See Object Identifiers appendix in SET Book 2: Programmer's Guide</td>
</tr>
<tr>
<td>FieldName</td>
<td>One or more field names to be displayed as labels for a fill-in form on requester's system; text is in the language specified in RegFormReq or Me-AqCInitReq</td>
</tr>
<tr>
<td>FieldDesc</td>
<td>Description of contents of field in the language specified in RegFormReq or Me-AqCInitReq; contains additional information for use when the cardholder requests help filling out the form.</td>
</tr>
<tr>
<td>FieldLen</td>
<td>Maximum length of field</td>
</tr>
<tr>
<td>FieldRequired</td>
<td>Boolean indicating whether data is required (either entered by the Cardholder or, if the field is invisible, populated by the application)</td>
</tr>
<tr>
<td>FieldInvisible</td>
<td>Boolean indicating that the field should not be displayed to the user; the application should either fill in the FieldValue based on FieldID or leave it empty.</td>
</tr>
</tbody>
</table>

Table 78: RegFormOrReferral

Continued on next page
RegFormOrReferral, continued

RegFormOrReferral (continued)

442 RegFormOrReferral ::= CHOICE {
443    regFormData   [0] RegFormData,
444    referralData  [1] ReferralData
445 }

447 RegFormData ::= SEQUENCE {
448    regTemplate  RegTemplate  OPTIONAL,
449    policy       PolicyText
450 }

470 ReferralData ::= SEQUENCE {
471    reason          Reason  OPTIONAL,  -- Displayed on requestor's system
472    referralURLSeq  ReferralURLSeq  OPTIONAL
473 } ( WITH COMPONENTS { ..., reason PRESENT } |
474     WITH COMPONENTS { ..., referralURLSeq PRESENT } )

452 RegTemplate ::= SEQUENCE {
453    regFormID     INTEGER (0..MAX),    -- CA assigned identifier
454    brandLogoURL  [0] URL OPTIONAL,
455    cardLogoURL   [1] URL OPTIONAL,
456    regFieldSeq   RegFieldSeq  OPTIONAL
457 }

482 PolicyText ::= SETString { ub-PolicyText }

476 Reason ::= SETString { ub-Reason }

478 ReferralURLSeq ::= SEQUENCE OF ReferralURL   -- Ordered by preference

459 RegFieldSeq ::= SEQUENCE SIZE(1..ub-FieldList) OF RegField

480 ReferralURL ::= URL

461 RegField ::= SEQUENCE {
462    fieldId         [0] OBJECT IDENTIFIER  OPTIONAL,
463    fieldName       FieldName,
464    fieldDesc       [1] EXPLICIT SETString { ub-FieldDesc }  OPTIONAL,
465    fieldLen        INTEGER (1..ub-FieldValue) DEFAULT ub-FieldValue,
466    fieldRequired   [2] BOOLEAN DEFAULT FALSE,
467    fieldInvisible  [3] BOOLEAN DEFAULT FALSE
468 }

616 FieldName ::= SETString { ub-FieldName }
Chapter 7
Certificate Management Messages

Overview

Introduction
Chapter 7 describes certificate management messages. Payload components of these messages are described in Chapter 6, which begins on page 147.

Organization
Chapter 7 includes the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate Initialization Pair - Cardholder</td>
<td>158</td>
</tr>
<tr>
<td>Certificate Initialization Pair - Merchant or Payment Gateway</td>
<td>160</td>
</tr>
<tr>
<td>Registration Form Pair - Cardholder Only</td>
<td>165</td>
</tr>
<tr>
<td>Certificate Request Pair</td>
<td>169</td>
</tr>
<tr>
<td>Certificate Inquiry Pair</td>
<td>177</td>
</tr>
</tbody>
</table>
Certificate Initialization Pair - Cardholder

CardCInitReq

<table>
<thead>
<tr>
<th>CardCInitReq</th>
<th>{RRPID, LID-EE, Chall-EE, BrandID, [Thumbs]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>LID-EE</td>
<td>Local ID; generated by and for the Cardholder system</td>
</tr>
<tr>
<td>Chall-EE</td>
<td>Cardholder’s challenge to CCA’s signature freshness</td>
</tr>
<tr>
<td>BrandID</td>
<td>BrandID of certificate requested</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Lists of Certificate (including Root), CRL, and BrandCRLIdentifier thumbprints currently held by Cardholder</td>
</tr>
</tbody>
</table>

Table 79: CardCInitReq

486 CardCInitReq ::= SEQUENCE {
487   rrpid     RRPID,
488   lid-EE    LocalID,
489   chall-EE  Challenge,
490   brandID   BrandID,
491   thumbs    [0] EXPLICIT Thumbs  OPTIONAL
492 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

232 BrandID ::= SETString { ub-BrandID }

330 Thumbs ::= SEQUENCE {
331   digestAlgorithm AlgorithmIdentifier {{DigestAlgorithms}},
332   certThumbs      [0] EXPLICIT Digests  OPTIONAL,
333   crlThumbs       [1] EXPLICIT Digests  OPTIONAL,
334   brandCRLIdThumbs [2] EXPLICIT Digests  OPTIONAL
335 }

Continued on next page
Certificate Initialization Pair - Cardholder, continued

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CardCInitRes</td>
<td>S(CA, CardCInitResTBS).</td>
</tr>
<tr>
<td>CardCInitResTBS</td>
<td>{RRPID, LID-EE, Chall-EE, [LID-CA], CAEThumb, [BrandCRLIdentifier], [Thumbs]}</td>
</tr>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>LID-EE</td>
<td>Copied from CardCInitReq</td>
</tr>
<tr>
<td>Chall-EE</td>
<td>Copied from CardCInitReq</td>
</tr>
<tr>
<td>LID-CA</td>
<td>Local ID; Generated by and for the CCA system</td>
</tr>
<tr>
<td>CAEThumb</td>
<td>Thumbprint of CCA key-exchange certificate that Cardholder should use to encrypt RegFormReq</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>See page 151.</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Copied from CardCInitReq</td>
</tr>
</tbody>
</table>

Table 80: CardCInitRes

494 CardCInitRes ::= S { CA, CardCInitResTBS }  
496 CardCInitResTBS ::= SEQUENCE {  
497   rrpid<RRPID>,  
498   lid-EE<LocalID>,  
499   chall-EE<Challenge>,  
500   lid-CA<LocalID OPTIONAL>,  
501   caeThumb[0] EXPLICIT CertThumb,  
502   brandCRLIdentifier[1] EXPLICIT BrandCRLIdentifier OPTIONAL,  
503   thumbs[2] EXPLICIT Thumbs OPTIONAL  
504 }

324 RRPID ::= OCTET STRING(SIZE(20))  
326 rrpid ::= OCTET STRING(SIZE(20))  
328 lid-EE ::= OCTET STRING(SIZE(20))  
329 chall-EE ::= OCTET STRING(SIZE(20))  
330 lid-CA ::= OCTET STRING(SIZE(20))  
331 caeThumb ::= OCTET STRING(SIZE(20))  
332 brandCRLIdentifier ::= OCTET STRING(SIZE(20))  
333 thumbs ::= OCTET STRING(SIZE(20))

Version 1.0
Certificate Initialization Pair - Merchant or Payment Gateway

Table 81: Me-AqCInitReq

<table>
<thead>
<tr>
<th>Me-AqCInitReq</th>
<th>{RRPID, LID-EE, Chall-EE, RequestType, IDData, BrandID, Language, [Thumbs]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>LID-EE</td>
<td>Local ID; generated by and for EE system</td>
</tr>
<tr>
<td>Chall-EE</td>
<td>EE’s challenge to CA’s signature freshness</td>
</tr>
<tr>
<td>RequestType</td>
<td>See page 149</td>
</tr>
<tr>
<td>IDData</td>
<td>See page 148</td>
</tr>
<tr>
<td>BrandID</td>
<td>BrandID of certificate requested</td>
</tr>
<tr>
<td>Language</td>
<td>Desired natural language for the rest of this flow</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Lists of Certificate (including Root), CRL, and BrandCRLIdentifier currently held by EE</td>
</tr>
</tbody>
</table>

508 Me-AqCInitReq ::= SEQUENCE {
509     rrpid       RRPID, 
510     lid-EE      LocalID, 
511     chall-EE    Challenge, 
512     requestType RequestType, 
513     idData      IDData, 
514     brandID     BrandID, 
515     language    Language, 
516     thumbs      [0] EXPLICIT Thumbs OPTIONAL 
517 }

Continued on next page
Certificate Initialization Pair - Merchant or Payment Gateway, continued

Me-AqCInitReq (continued)

324 RRIPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

421 RequestType ::= ENUMERATED { -- Indicates requestor and type of request
422    cardInitialSig    (1),
423    -- cardInitialEnc    (2),                                       Reserved
424    -- cardInitialBoth  (3),                                       Reserved
425    merInitialSig     (4),
426    merInitialEnc     (5),
427    merInitialBoth    (6),
428    pgwyInitialSig    (7),
429    pgwyInitialEnc    (8),
430    pgwyInitialBoth   (9),
431    cardRenewalSig   (10),
432    -- cardRenewalEnc   (11),                                       Reserved
433    -- cardRenewalBoth (12),                                       Reserved
434    merRenewalSig    (13),
435    merRenewalEnc    (14),
436    merRenewalBoth   (15),
437    pgwyRenewalSig   (16),
438    pgwyRenewalEnc   (17),
439    pgwyRenewalBoth  (18)
440 }

404 IDData ::= CHOICE {                      -- Merchants and Acquirers only
405    merchantAcquirerID  [0] MerchantAcquirerID,
406    acquirerID          [1] AcquirerID
407 }

232 BrandID ::= SETString { ub-BrandID }

282 Language ::= VisibleString (SIZE(1..ub-RFC1766-language))

330 Thumbs ::= SEQUENCE {
331    digestAlgorithm   AlgorithmIdentifier {{DigestAlgorithms}},
332    certThumbs        [0] EXPLICIT Digests  OPTIONAL,
333    crlThumbs         [1] EXPLICIT Digests  OPTIONAL,
334    brandCRLIdThumbs  [2] EXPLICIT Digests  OPTIONAL
335 }

Continued on next page
Certificate Initialization Pair - Merchant or Payment Gateway, continued

Me-AqCInitRes

<table>
<thead>
<tr>
<th>Me-AqCInitRes</th>
<th>S(CA, Me-AqCInitResTBS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Me-AqCInitResTBS</td>
<td>{RRPID, LID-EE, Chall-EE, [LID-CA], Chall-CA, RequestType, RegFormOrReferral, [AcctDataField], CAEThumb, [BrandCRLIdentifier], [Thumbs]}</td>
</tr>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>LID-EE</td>
<td>Copied from Me-AqCInitReq</td>
</tr>
<tr>
<td>Chall-EE</td>
<td>Copied from Me-AqCInitReq</td>
</tr>
<tr>
<td>LID-CA</td>
<td>Local ID; generated by and for CA system</td>
</tr>
<tr>
<td>Chall-CA</td>
<td>CA’s challenge to EE’s signature freshness</td>
</tr>
<tr>
<td>RequestType</td>
<td>See page 149</td>
</tr>
<tr>
<td>RegFormOrReferral</td>
<td>See page 155.</td>
</tr>
<tr>
<td>AcctDataField</td>
<td>RegField (see “RegFormOrReferral” on page 155); an additional registration field to be displayed to collect the value for AcctData in CertReq</td>
</tr>
<tr>
<td>CAEThumb</td>
<td>Thumbprint of CA key-exchange certificate that should be used to encrypt CertReq</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>See page 151</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Copied from Me-AqCInitReq</td>
</tr>
</tbody>
</table>

Table 82: Me-AqCInitRes

Continued on next page
Certificate Initialization Pair - Merchant or Payment Gateway, continued

Me-AqCInitRes (continued)

519 Me-AqCInitRes ::= S { CA, Me-AqCInitResTBS }

521 Me-AqCInitResTBS ::= SEQUENCE {
522     rrpid               RRPID,
523     lid-EE              LocalID,
524     chall-EE            Challenge,
525     lid-CA              [0] LocalID OPTIONAL,
526     chall-CA            Challenge,
527     requestType         RequestType,
528     regFormOrReferral   RegFormOrReferral,
529     acctDataField       [1] RegField OPTIONAL,
530     caeThumb            [2] EXPLICIT CertThumb,
531     brandCRLIdentifier  [3] EXPLICIT BrandCRLIdentifier OPTIONAL,
532     thumbs              [4] EXPLICIT Thumbs OPTIONAL
533 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

421 RequestType ::= ENUMERATED {  -- Indicates requestor and type of request
422     cardInitialSig    (1),
423     -- cardInitialEnc    (2), Reserved
424     -- cardInitialBoth  (3), Reserved
425     merInitialSig     (4),
426     merInitialEnc     (5),
427     merInitialBoth    (6),
428     pgwIniInitialSig   (7),
429     pgwIniInitialEnc   (8),
430     pgwIniInitialBoth  (9),
431     cardRenewalSig    (10),
432     -- cardRenewalEnc    (11), Reserved
433     -- cardRenewalBoth  (12), Reserved
434     merRenewalSig     (13),
435     merRenewalEnc     (14),
436     merRenewalBoth    (15),
437     pgwIniRenewalSig   (16),
438     pgwIniRenewalEnc   (17),
439     pgwIniRenewalBoth  (18)
440 }

Continued on next page
Certificate Initialization Pair - Merchant or Payment Gateway, continued

Me-AqCInitRes (continued)

442 RegFormOrReferral ::= CHOICE {
443   regFormData   [0] RegFormData,
444   referralData  [1] ReferralData
445 }

191 BrandCRLIdentifier ::= SIGNED {
192   EncodedBrandCRLID
193 } ( CONSTRANIED BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } )

330 Thumbs ::= SEQUENCE {
331   digestAlgorithm   AlgorithmIdentifier {{DigestAlgorithms}},
332   certThumbs        [0] EXPLICIT Digests  OPTIONAL,
333   crlThumbs         [1] EXPLICIT Digests  OPTIONAL,
334   brandCRLIdThumbs  [2] EXPLICIT Digests  OPTIONAL
335 }
Registration Form Pair - Cardholder Only

RegFormReq

<table>
<thead>
<tr>
<th>RegFormReq</th>
<th>EXH(CA, RegFormReqData, PANOnly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegFormReqData</td>
<td>{RRPID, LID-EE, Chall-EE2, [LID-CA], RequestType, Language, [Thumbs]}</td>
</tr>
<tr>
<td>PANOnly</td>
<td>See page 23</td>
</tr>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>LID-EE</td>
<td>Copied from CardCInitRes</td>
</tr>
<tr>
<td>Chall-EE2</td>
<td>EE’s challenge to CA’s signature freshness</td>
</tr>
<tr>
<td>LID-CA</td>
<td>Copied from CardCInitRes</td>
</tr>
<tr>
<td>RequestType</td>
<td>See page 149</td>
</tr>
<tr>
<td>Language</td>
<td>Desired natural language for the rest of this flow</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Lists of Certificate (including Root), CRL, and BrandCRLIdentifier currently held by Cardholder</td>
</tr>
</tbody>
</table>

Table 83: RegFormReq

537 RegFormReq ::= EXH { CA, RegFormReqData, PANOnly } |
542 RegFormReqData ::= SEQUENCE { |
543    rrpid    RRPID, |
544    lid-EE   LocalID, |
545    chall-EE2 Challenge, |
546    lid-CA   [0] LocalID OPTIONAL, |
547    requestType RequestType, |
548    language Language, |
549    thumbs   [1] EXPLICIT Thumbs OPTIONAL |
550 } |
552 PANOnly ::= SEQUENCE { |
553    pan       PAN, |
554    exNonce   Nonce |
555 } |

Continued on next page
Registration Form Pair - Cardholder Only, continued

RegFormReq (continued)

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

421 RequestType ::= ENUMERATED { -- Indicates requestor and type of request
  422 cardInitialSig    (1),
  423 -- cardInitialEnc (2),  Reserved
  424 -- cardInitialBoth (3),  Reserved
  425 merInitialSig     (4),
  426 merInitialEnc     (5),
  427 merInitialBoth    (6),
  428 pgwyInitialSig    (7),
  429 pgwyInitialEnc    (8),
  430 pgwyInitialBoth   (9),
  431 cardRenewalSig   (10),
  432 -- cardRenewalEnc (11),  Reserved
  433 -- cardRenewalBoth (12),  Reserved
  434 merRenewalSig     (13),
  435 merRenewalEnc     (14),
  436 merRenewalBoth    (15),
  437 pgwyRenewalSig    (16),
  438 pgwyRenewalEnc    (17),
  439 pgwyRenewalBoth   (18)
  440 }

282 Language ::= VisibleString (SIZE(1..ub-RFC1766-language))

Continued on next page
Registration Form Pair - Cardholder Only, continued

<table>
<thead>
<tr>
<th>RegFormRes</th>
<th>S(CA, RegFormResTBS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegFormResTBS</td>
<td>{RRPID, LID-EE, Chall-EE2, [LID-CA], Chall-CA, [CAEThumb], RequestType, RegFormOrReferral, [BrandCRLIdentifier], [Thumbs]}</td>
</tr>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>LID-EE</td>
<td>Copied from RegFormReq</td>
</tr>
<tr>
<td>Chall-EE2</td>
<td>Copied from RegFormReq</td>
</tr>
<tr>
<td>LID-CA</td>
<td>Local ID; generated by and for CA system (new value may be specified)</td>
</tr>
<tr>
<td>Chall-CA</td>
<td>CA’s challenge to requestor’s signature freshness</td>
</tr>
<tr>
<td>CAEThumb</td>
<td>Thumbprint of CA key-exchange certificate that should be used to encrypt CertReq; if this field is not present, the certificate identified in CardCInitRes is used.</td>
</tr>
<tr>
<td>RequestType</td>
<td>See page 149</td>
</tr>
<tr>
<td>RegFormOrReferral</td>
<td>See page 155.</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>See page 151.</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Copied from RegFormReq</td>
</tr>
</tbody>
</table>

Table 84: RegFormRes

557  RegFormRes ::= S { CA, RegFormResTBS }  
559  RegFormResTBS ::= SEQUENCE {  
560    rrpid RRPID,  
561    lid-EE LocalID,  
562    chall-EE2 Challenge,  
563    lid-CA [0] LocalID OPTIONAL,  
564    chall-CA Challenge,  
565    caeThumb [1] EXPLICIT CertThumb OPTIONAL,  
566    requestType RequestType,  
567    formOrReferral RegFormOrReferral,  
568    brandCRLIdentifier [2] EXPLICIT BrandCRLIdentifier OPTIONAL,  
569    thumbs [3] EXPLICIT Thumbs OPTIONAL  
570   }  

Continued on next page
Registration Form Pair - Cardholder Only, continued

RegFormRes (continued)

\[
324 \text{ RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification}
\]

\[
421 \text{ RequestType ::= ENUMERATED { -- Indicates requestor and type of request}
422 \quad \text{ cardInitialSig (1),}
423 \quad \text{ cardInitialEnc (2),} \quad \text{ Reserved}
424 \quad \text{ cardInitialBoth (3),} \quad \text{ Reserved}
425 \quad \text{ merInitialSig (4),}
426 \quad \text{ merInitialEnc (5),}
427 \quad \text{ merInitialBoth (6),}
428 \quad \text{ pgwyInitialSig (7),}
429 \quad \text{ pgwyInitialEnc (8),}
430 \quad \text{ pgwyInitialBoth (9),}
431 \quad \text{ cardRenewalSig (10),}
432 \quad \text{ cardRenewalEnc (11),} \quad \text{ Reserved}
433 \quad \text{ cardRenewalBoth (12),} \quad \text{ Reserved}
434 \quad \text{ merRenewalSig (13),}
435 \quad \text{ merRenewalEnc (14),}
436 \quad \text{ merRenewalBoth (15),}
437 \quad \text{ pgwyRenewalSig (16),}
438 \quad \text{ pgwyRenewalEnc (17),}
439 \quad \text{ pgwyRenewalBoth (18)}
440 }
\]

\[
442 \text{ RegFormOrReferral ::= CHOICE {}
443 \quad \text{ regFormData [0] RegFormData,}
444 \quad \text{ referralData [1] ReferralData}
445 }
\]

\[
191 \text{ BrandCRLIdentifier ::= SIGNED {}
192 \quad \text{ EncodedBrandCRLID}
193 \quad \text{ CONstrained BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- } }
\]

\[
330 \text{ Thumbs ::= SEQUENCE {}
331 \quad \text{ digestAlgorithm AlgorithmIdentifier {{DigestAlgorithms}},}
332 \quad \text{ certThumbs [0] EXPLICIT Digests OPTIONAL,}
333 \quad \text{ crlThumbs [1] EXPLICIT Digests OPTIONAL,}
334 \quad \text{ brandCRLIdThumbs [2] EXPLICIT Digests OPTIONAL}
335 }
\]
Certificate Request Pair

CertReq

CertReq

< EncX(EE, CA, CertReqData, AcctInfo), 
Enc(EE, CA, CertReqData) >

Up to two signatures are implicit in the encapsulation. 
CertReqTBE and AcctInfo may be signed by any or all of 
the private keys corresponding to the following end entity 
certificates:
• the private key for which a new Signature certificate,
• an existing Signature certificate, for an Encryption certificate 
request, or
• an existing Signature certificate, for a renewal request. 
These “signatures” without a corresponding signature 
certificate are pro forma only; they prove only that EE holds 
the private key.

CertReqData

{RRPID, LID-EE, Chall-EE3, [LID-CA], [Chall-CA], 
RequestType, RequestDate, [IDData], RegFormID, 
[RegForm], [CABackKeyData], PublicKeySorE, 
[EEThumb], [Thumbs]}

AcctInfo

< PANData0, AcctData >

If the requester is a Cardholder, PANData0 is included. 
If the requester is a Merchant or an Acquirer, AcctData is 
optional.

RRPID

Request/response pair ID

LID-EE

Copied from RegFormRes or Me-AqCInitRes

Chall-EE3

EE’s challenge to CA’s signature freshness

LID-CA

Copied from RegFormRes or Me-AqCInitRes

Chall-CA

Copied from RegFormRes or Me-AqCInitRes

RequestType

See page 149.

RequestDate

Date of certificate request

IDData

See page 148. Omit if EE is Cardholder.

Table 85: CertReq

Continued on next page
Certificate Request Pair, continued

CertReq (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegFormID</td>
<td>CA-assigned identifier</td>
</tr>
<tr>
<td>RegForm</td>
<td>{RegFormItems +} The field names copied from RegFormRes or Me-AqCInitRes, now accompanied by values filled in by EE’s implementation</td>
</tr>
<tr>
<td>CABackKeyData</td>
<td>{CAAlgId, CAAKey}</td>
</tr>
<tr>
<td>PublicKeySorE</td>
<td>{PublicKeyS, PublicKeyE} The entity’s public key(s). At least one key shall be specified. A user may request a signature certificate, an encryption certificate, or both.</td>
</tr>
<tr>
<td>EEThumb</td>
<td>Thumbprint of entity key-encryption certificate that is being renewed.</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Lists of Certificate (including Root), CRL, and BrandCRLIdentifier currently held by EE</td>
</tr>
<tr>
<td>PANData0</td>
<td>See page 153.</td>
</tr>
<tr>
<td>AcctData</td>
<td>See page 154.</td>
</tr>
<tr>
<td>RegFormFields</td>
<td>{FieldName, FieldValue}</td>
</tr>
<tr>
<td>CAAlgId</td>
<td>Symmetric key algorithm identifier</td>
</tr>
<tr>
<td>CAAKey</td>
<td>Secret key corresponding to the algorithm identifier</td>
</tr>
<tr>
<td>PublicKeyS</td>
<td>Proposed public signature key to certify</td>
</tr>
<tr>
<td>PublicKeyE</td>
<td>Proposed public encryption key to certify</td>
</tr>
<tr>
<td>FieldName</td>
<td>One or more field names to be displayed as a fill-in form on the requester’s system, as a text field in the language specified in RegFormReq or Me-AqCInitReq</td>
</tr>
<tr>
<td>FieldValue</td>
<td>Values entered by EE</td>
</tr>
</tbody>
</table>

Table 85: CertReq, continued

Continued on next page
Certificate Request Pair, continued

CertReq (continued)

574 CertReq ::= CHOICE {
575   encx  [0] EXPLICIT EncX { EE, CA, CertReqData, AcctInfo },
576   enc   [1] EXPLICIT Enc { EE, CA, CertReqData }
577 }

592 CertReqData ::= SEQUENCE {
593   rrpid    RRPID,
594   lid-EE   LocalID,
595   chall-EE3 Challenge,
596   lid-CA   [0] LocalID OPTIONAL,
597   chall-CA [1] Challenge OPTIONAL,
598   requestType RequestType,
599   requestDate Date,
600   idData   [2] EXPLICIT IDData OPTIONAL,
601   regFormID INTEGER (0..MAX), -- CA assigned identifier
602   regForm   [3] RegForm OPTIONAL,
603   caBackKeyData [4] EXPLICIT BackKeyData OPTIONAL,
604   publicKeySorE PublicKeySorE,
605   eeThumb   [5] EXPLICIT CertThumb OPTIONAL,
606   thumbs    [6] EXPLICIT Thumbs OPTIONAL
607 }

392 AcctInfo ::= CHOICE {
393   panData0  [0] EXPLICIT PANData0,
394   acctData  [1] EXPLICIT AcctData
395 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

421 RequestType ::= ENUMERATED { -- Indicates requestor and type of request
422   cardInitialSig    (1),       Reserved
423   -- cardInitialEnc  (2),       Reserved
424   -- cardInitialBoth (3),       Reserved
425   merInitialSig     (4),
426   merInitialEnc     (5),
427   merInitialBoth    (6),
428   pgwyInitialSig    (7),
429   pgwyInitialEnc    (8),
430   pgwyInitialBoth   (9),
431   cardRenewalSig    (10),      Reserved
432   -- cardRenewalEnc (11),      Reserved
433   -- cardRenewalBoth (12),      Reserved
434   merRenewalSig     (13),
435   merRenewalEnc     (14),
436   merRenewalBoth    (15),
437   pgwyRenewalSig    (16),
438   pgwyRenewalEnc    (17),
439   pgwyRenewalBoth   (18)
440 }

Continued on next page
Certificate Request Pair, continued

**CertReq** (continued)

404 IIDData ::= CHOICE { -- Merchants and Acquirers only
  405  merchantAcquirerID [0] MerchantAcquirerID,
  406  acquirerID [1] AcquirerID
  407 }

609 RegForm ::= SEQUENCE SIZE(1..ub-FieldList) OF RegFormItems

623 PublicKeySorE ::= SEQUENCE {
  624  publicKeyS [0] EXPLICIT SubjectPublicKeyInfo{{SignatureAlgorithms}}
  625  OPTIONAL,
  626  publicKeyE [1] EXPLICIT SubjectPublicKeyInfo{{KeyEncryptionAlgorithms}}
  627  OPTIONAL
  628 } --

629 -- At least one component shall be present. A user may request a
630 -- signature certificate, an encryption certificate, or both.
631 --
632 { WITH COMPONENTS { ..., publicKeyS PRESENT } |
633  WITH COMPONENTS { ..., publicKeyE PRESENT } }

307 PANData0 ::= SEQUENCE {
  308  pan PAN,
  309  cardExpiry CardExpiry,
  310  cardSecret Secret,
  311  exNonce Nonce
  312 }

397 AcctData ::= SEQUENCE {
  398  acctIdentification AcctIdentification,
  399  exNonce Nonce
  400 }

611 RegFormItems ::= SEQUENCE {
  612  fieldName FieldName,
  613  fieldValue FieldValue
  614 }

692 CAKey ::= BackKeyData

616 FieldName ::= SETString { ub-FieldName }

618 fieldValue ::= CHOICE {
  619  setString SETString { ub-FieldValue },
  620  octetString OCTET STRING (SIZE(1..ub-FieldValue))
  621 }

Continued on next page
Certificate Request Pair, continued

<table>
<thead>
<tr>
<th>CertRes</th>
<th>&lt; S(CA, CertResData), EncK(CABackKeyData, CA, CertResData) &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CertResData</td>
<td>{RRPID, LID-EE, Chall-EE3, LID-CA, CertStatus, [CertThumbs], [BrandCRLIdentifier], [Thumbs]}</td>
</tr>
<tr>
<td>CABackKeyData</td>
<td>Copied from CertReq</td>
</tr>
<tr>
<td>RRPID</td>
<td>Request/response pair ID</td>
</tr>
<tr>
<td>LID-EE</td>
<td>Copied from prior CertReq</td>
</tr>
<tr>
<td>Chall-EE3</td>
<td>Copied from CertReq. Requester checks for match with remembered value.</td>
</tr>
<tr>
<td>LID-CA</td>
<td>Copied from CertReq. If not present in the CertReq, new values are assigned.</td>
</tr>
<tr>
<td>CertStatus</td>
<td>{CertStatusCode, [Nonce-CCA], [EEMessage], [CaMsg], [FailedItemSeq]}</td>
</tr>
<tr>
<td>CertThumbs</td>
<td>If request is complete, the thumbprints of the enclosed signature and or encryption certificates</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>See page 151.</td>
</tr>
<tr>
<td>Thumbs</td>
<td>Copied from CertReq</td>
</tr>
</tbody>
</table>

Continued on next page
Certificate Request Pair, continued

CertRes (continued)

<table>
<thead>
<tr>
<th>CertStatusCd</th>
<th>Enumerated code indicating the status of the certificate request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonce-CCA</td>
<td>If request is complete and from a cardholder, the other half of</td>
</tr>
<tr>
<td></td>
<td>the ultimate shared secret between Cardholder and CCA. See</td>
</tr>
<tr>
<td></td>
<td>PANData0 under “CertReq” on page 169. Present only if EE</td>
</tr>
<tr>
<td></td>
<td>is Cardholder.</td>
</tr>
<tr>
<td>EEMessage</td>
<td>Message in natural language to be displayed on the EE system</td>
</tr>
<tr>
<td>CAMsg</td>
<td>{[CardLogoURL], [BrandLogoURL], [CardCurrency],</td>
</tr>
<tr>
<td></td>
<td>[CardholderMsg] }</td>
</tr>
<tr>
<td></td>
<td>If request is complete and from a cardholder</td>
</tr>
<tr>
<td>FailedItemSeq</td>
<td>{FailedItem+}</td>
</tr>
<tr>
<td>CardLogoURL</td>
<td>URL pointing to graphic of card logo (issuer-specific)</td>
</tr>
<tr>
<td>BrandLogoURL</td>
<td>URL pointing to graphic of payment card brand logo</td>
</tr>
<tr>
<td>CardCurrency</td>
<td>Cardholder billing currency</td>
</tr>
<tr>
<td>CardholderMsg</td>
<td>A message in the Cardholder's natural language to be</td>
</tr>
<tr>
<td></td>
<td>displayed by the software</td>
</tr>
<tr>
<td>FailedItem</td>
<td>{ItemNumber, ItemReason}</td>
</tr>
<tr>
<td>ItemNumber</td>
<td>Indicates the position of the failed item in the list of</td>
</tr>
<tr>
<td></td>
<td>registration fields. A value of 0 indicates the AcctData field.</td>
</tr>
<tr>
<td>ItemReason</td>
<td>The reason for the failure, as a text field in the language</td>
</tr>
<tr>
<td></td>
<td>specified in RegFormReq</td>
</tr>
</tbody>
</table>

Table 85: CertReq, continued

Continued on next page
Certificate Request Pair, continued

CertRes (continued)

635 CertRes ::= CHOICE {
636   certResTBS   [0] EXPLICIT S { CA, CertResData },
637   certResTBSK  [1] EXPLICIT EncK { CAKey, CA, CertResData }
638 }

643 CertResData ::= SEQUENCE {
644   rrpid             RRPID,
645   lid-EE           LocalID,
646   chall-EE3        Challenge,
647   lid-CA           LocalID,
648   certStatus      CertStatus,
649   certThumbs      [0] EXPLICIT Thumbs  OPTIONAL,
650   brandCRLIdentifier [1] EXPLICIT BrandCRLIdentifier  OPTIONAL,
651   thumbs          [2] EXPLICIT Thumbs  OPTIONAL
652 }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

654 CertStatus ::= SEQUENCE {
655   certStatusCode  CertStatusCode,
656   nonceCCA        [0] Nonce  OPTIONAL,
657   eeMessage       SETString { ub-eeMessage }  OPTIONAL,
658   caMsg           [1] CAMsg  OPTIONAL,
659   failedItemSeq   [2] FailedItemSeq  OPTIONAL
660 }

669 CertStatusCode ::= ENUMERATED {          -- In-process status of CertReq
670   requestComplete           (1),
671   invalidLanguage           (2),
672   invalidBIN                (3),
673   sigValidationFail         (4),
674   decryptionError           (5),
675   requestInProgress         (6),
676   rejectedByIssuer          (7),
677   requestPended             (8),
678   rejectedByAquirer         (9),
679   regFormAnswerMalformed    (10),
680   rejectedByCA              (11),
681   unableToEncryptResponse   (12)
682 }

Continued on next page
CertRes (continued)

684 CAMsg ::= SEQUENCE {
685    cardLogoURL     [0] URL  OPTIONAL,
686    brandLogoURL    [1] URL  OPTIONAL,
687    cardCurrency    [2] Currency  OPTIONAL,
688    cardholderMsg   [3] EXPLICIT
689        SETString { ub-cardholderMsg }  OPTIONAL
690 }

662 FailedItemSeq ::= SEQUENCE SIZE(1..ub-FieldList) OF FailedItem

664 FailedItem ::= SEQUENCE {
665    itemNumber     INTEGER (1..50),
666    itemReason     SETString { ub-Reason }
667 }
Certificate Inquiry Pair

<table>
<thead>
<tr>
<th>CertInqReq</th>
<th>CertInqReqTBS</th>
<th>RRPID</th>
<th>LID-EE</th>
<th>Chall-EE3</th>
<th>LID-CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(EE, CertInqReqTBS)</td>
<td>{RRPID, LID-EE, Chall-EE3, LID-CA}</td>
<td>Request/response pair identifier.</td>
<td>Copied from CertRes</td>
<td>EE's challenge to CA's signature freshness</td>
<td>Copied from CertRes</td>
</tr>
</tbody>
</table>

| Table 86: CertInqReq |

696 CertInqReq ::= S { EE, CertInqReqTBS }

698 CertInqReqTBS ::= SEQUENCE {
699   rrpid    RRPID,
700   lid-EE  LocalID,
701   chall-EE3 Challenge,
702   lid-CA  LocalID
703  }

324 RRPID ::= OCTET STRING(SIZE(20)) -- Request response pair identification

CertInqRes

<table>
<thead>
<tr>
<th>CertInqRes</th>
<th>Identical to a CertRes: see page 173.</th>
</tr>
</thead>
</table>

| Table 87: CertInqRes |

705 CertInqRes ::= CertRes
Part II
ASN.1 Code

SET modules

The following modules are defined in SET.

<table>
<thead>
<tr>
<th>Module</th>
<th>Starting Line Number</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetAttribute</td>
<td>2989</td>
<td>231</td>
</tr>
<tr>
<td>SetCertificate</td>
<td>1980</td>
<td>213</td>
</tr>
<tr>
<td>SetCertificateExtensions</td>
<td>2046</td>
<td>214</td>
</tr>
<tr>
<td>SetCertMsgs</td>
<td>362</td>
<td>185</td>
</tr>
<tr>
<td>SetCRL</td>
<td>2612</td>
<td>224</td>
</tr>
<tr>
<td>SetMessage</td>
<td>4</td>
<td>179</td>
</tr>
<tr>
<td>SetPayMsgs</td>
<td>721</td>
<td>191</td>
</tr>
<tr>
<td>SetPKCS7Plus</td>
<td>2660</td>
<td>225</td>
</tr>
</tbody>
</table>

1 -- History
2 -- 31 May 1997 Version 1.0
3
4 SetMessage
5 { joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 0 }
6 DEFINITIONS IMPLICIT TAGS ::= BEGIN
7
8 --
9 -- This module defines types for use in the SET protocol certificate and
10 -- payment flow messages.
11 --
12
13 -- EXPORTS All;
14
15 IMPORTS
16
17 ALGORITHM-IDENTIFIER, AlgorithmIdentifier {}, Name, SETString {}
18 FROM SetAttribute
19
20 SIGNED {} FROM SetCertificate
21
22 EXTENSION, Extensions, ub-cityName, ub-postalCode, ub-stateProvince
23 FROM SetCertificateExtensions
24
25 CardCInitReq, CardCInitRes, CertInqReq, CertInqRes, CertReq, CertRes,
26 Me-AqCInitReq, Me-AqCInitRes, RegFormReq, RegFormRes
27 FROM SetCertMsgs
28
29 AuthReq, AuthRes, AuthRevReq, AuthRevRes, BatchAdminReq,
30 BatchAdminRes, CapReq, CapRes, CapRevReq, CapRevRes, CredReq,
31 CredRes, CredRevReq, CredRevRes, InqReq, InqRes, PCertReq,
32 PCertRes, PInitReq, PInitRes, PReq, PRes
33 FROM SetPayMsgs
34
35

Version 1.0
36 CA, ContentEncryptionAlgorithms, Digest, DigestAlgorithms, Digests, EE, S {}
37 FROM SetPKCS7Plus
38
39 ub-phone
40 FROM SetMarketData;
41
42
43 MessageWrapper ::= SEQUENCE {
44 messageHeader MessageHeader,
45 message [0] EXPLICIT MESSAGE.&Type (Message),
46 mwExtensions [1] MsgExtensions {{MWExtensionsIOS}} OPTIONAL
47 }
48
49 -- An information object set is defined for each extensible PDU
50 --
51 -- Note: each of these information object sets uses the extension
52 -- marker (...) to allow vendors to add supported extensions to
53 -- their local copy of the ASN.1. Extensions added by vendors
54 -- should appear after the extension marker.
55
56 MWExtensionsIOS EXTENSION ::= { ... }
57
58 MessageHeader ::= SEQUENCE {
59 version INTEGER { setVer1(1) } (setVer1),
60 revision INTEGER (0) DEFAULT 0, -- This is version 1.0
61 date Date,
62 messageIDs [0] MessageIDs OPTIONAL,
63 rrpid [1] RRPID OPTIONAL,
64 swIdent SWIdent
65 }
66
67 MessageIDs ::= SEQUENCE {
68 lid-C [0] LocalID OPTIONAL,
69 lid-M [1] LocalID OPTIONAL,
70 xID [2] XID OPTIONAL
71 }
72
73 MESSAGE ::= TYPE-IDENTIFIER -- ISO/IEC 8824-2:1995(E), Annex A
74
75 Message ::= CHOICE {
76
77 purchaseInitRequest [ 0] EXPLICIT PInitReq,
78 purchaseInitResponse [ 1] EXPLICIT PInitRes,
79
80 purchaseRequest [ 2] EXPLICIT PReq,
81 purchaseResponse [ 3] EXPLICIT PRes,
82
83 inquiryRequest [ 4] EXPLICIT InqReq,
84 inquiryResponse [ 5] EXPLICIT InqRes,
85
86 authorizationRequest [ 6] EXPLICIT AuthReq,
87 authorizationResponse [ 7] EXPLICIT AuthRes,
88
89 authReversalRequest [ 8] EXPLICIT AuthRevReq,
90 authReversalResponse [ 9] EXPLICIT AuthRevRes,
91
92 captureRequest [10] EXPLICIT CapReq,
captureReversalRequest  [12] EXPLICIT CapRevReq,
creditRequest          [14] EXPLICIT CredReq,
creditResponse         [15] EXPLICIT CredRes,
creditReversalRequest  [16] EXPLICIT CredRevReq,
creditReversalResponse [17] EXPLICIT CredRevRes,
pCertificateRequest    [18] EXPLICIT PCertReq,
pCertificateResponse  [19] EXPLICIT PCertRes,
batchAdministrationRequest [20] EXPLICIT BatchAdminReq,
batchAdministrationResponse [21] EXPLICIT BatchAdminRes,
cardholderCInitRequest  [22] EXPLICIT CardCInitReq,
cardholderCInitResponse [23] EXPLICIT CardCInitRes,
meAqCInitRequest      [24] EXPLICIT Me-AqCInitReq,
meAqCInitResponse     [25] EXPLICIT Me-AqCInitRes,
registrationFormRequest [26] EXPLICIT RegFormReq,
registrationFormResponse [27] EXPLICIT RegFormRes,
certificateRequest     [28] EXPLICIT CertReq,
certificateResponse    [29] EXPLICIT CertRes,
certificateInquiryRequest [30] EXPLICIT CertInqReq,
certificateInquiryResponse [31] EXPLICIT CertInqRes,
error                 [999] EXPLICIT Error
}

(MsgExtensions {EXTENSION:InfoObjectSet} ::= 
  SEQUENCE OF MsgExtension {{InfoObjectSet}})

(MsgExtension {EXTENSION:InfoObjectSet} ::= 
  SEQUENCE {
    extnID EXTENSION.&id({InfoObjectSet}),
critical EXTENSION.&critical({InfoObjectSet}{@extnID})
    DEFAULT FALSE,
    extnValue [0] EXPLICIT EXTENSION.& ExtenType {{InfoObjectSet}{@extnID}}
  })

Error ::= CHOICE {
  signedError  [0] EXPLICIT SignedError,
  unsignedError [1] EXPLICIT ErrorTBS
}

SignedError ::= S {EE, ErrorTBS}

ErrorTBS ::= SEQUENCE {


errorCode     ErrorCode,
errorNonce     Nonce,
errorOID      [0] OBJECT IDENTIFIER  OPTIONAL,
errorThumb    [1] EXPLICIT CertThumb  OPTIONAL,
errorMsg      [2] EXPLICIT ErrorMsg
 }               -- Either the

ErrorMsg ::= CHOICE { messageHeader  [0] EXPLICIT MessageHeader, -- MessageHeader or a
badWrapper     [1] OCTET STRING (SIZE(1..20000)) -- copy of the message
 }               -- Either the

ErrorCode ::= ENUMERATED {
  unspecifiedFailure      (1),
  messageNotSupported     (2),
  decodingFailure         (3),
  invalidCertificate      (4),
  expiredCertificate      (5),
  revokedCertificate      (6),
  missingCertificate      (7),
  signatureFailure        (8),
  badMessageHeader        (9),
  wrapperMsgMismatch     (10),
  versionTooOld          (11),
  versionTooNew          (12),
  unrecognizedExtension  (13),
  messageTooBig          (14),
  signatureRequired      (15),
  messageTooOld          (16),
  messageTooNew          (17),
  thumbsMismatch         (18),
  unknownRRPID           (19),
  unknownXID             (20),
  unknownLID             (21),
  challengeMismatch      (22)
 }               -- Either the

BrandCRLIdentifier ::= SIGNED {
  EncodedBrandCRLID
} ( CONSTRAINED BY { -- Verify Or Sign UnsignedBrandCRLIdentifier -- })

EncodedBrandCRLID ::= TYPE-IDENTIFIER.&Type (UnsignedBrandCRLIdentifier)

UnsignedBrandCRLIdentifier ::= SEQUENCE {
  version          INTEGER { bVer1(0) } (bVer1),
  sequenceNum      INTEGER (0..MAX),
  brandID          BrandID,
  notBefore        GeneralizedTime,
  notAfter         GeneralizedTime,
  crlIdentifierSeq [0] CRLIdentifierSeq  OPTIONAL,
  bCRLExtensions   [1] Extensions  OPTIONAL
 }

-- Notification to Brand CA that a CRL has been updated

CRLNotification ::= S{CA, CRLNotificationTBS}
CRLNotificationTBS ::= SEQUENCE {
    date Date, -- Date of notification
    crlThumbprint Digest
}

CRLNotificationRes ::= S{CA, CRLNotificationResTBS}

CRLNotificationResTBS ::= SEQUENCE {
    date Date, -- Copied from CRLNotification
    crlThumbprint Digest
}

BCIDistribution ::= S{CA, BCIDistributionTBS}

BCIDistributionTBS ::= SEQUENCE {
    date Date,
    bci [0] BrandCRLIdentifier
}

BrandID ::= SETString { ub-BrandID }

CRLIdentifierSeq ::= SEQUENCE OF CRLIdentifier

CRLIdentifier ::= SEQUENCE {
    issuerName Name, -- CRL issuer Distinguished Name
    crlNumber INTEGER (0..MAX) -- crlNumber extension sequence number
}

-- Common definitions

BackKeyData ::= SEQUENCE {
    backAlgID ALGORITHM-IDENTIFIER.&id({ContentEncryptionAlgorithms}),
    backKey BackKey
}

BackKey ::= OCTET STRING (SIZE(1..24)) -- Secret

BIN ::= NumericString (SIZE(6)) -- Bank identification number

CardExpiry ::= NumericString (SIZE(6)) -- YYYYMM expiration date of card

CertThumb ::= SEQUENCE {
    digestAlgorithm AlgorithmIdentifier {{DigestAlgorithms}},
    thumbprint Digest
}

Challenge ::= OCTET STRING (SIZE(20)) -- Signature freshness challenge

CountryCode ::= INTEGER (1..999) -- ISO-3166 country code

Currency ::= INTEGER (1..999) -- ISO-4217 currency code

Date ::= GeneralizedTime

DateTime ::= SEQUENCE {
Distance ::= SEQUENCE {
    scale DistanceScale,  
    dist INTEGER (0..MAX)  
}

DistanceScale ::= ENUMERATED {
    miles       (0),
    kilometers  (1)
}

Language ::= VisibleString (SIZE(1..ub-RFC1766-language))

LocalID ::= OCTET STRING (SIZE(1..20))

Location ::= SEQUENCE {
    countryCode CountryCode,  
    city       [0] EXPLICIT SETString { ub-cityName } OPTIONAL,  
    stateProvince [1] EXPLICIT SETString { ub-stateProvince } OPTIONAL,  
    postalCode  [2] EXPLICIT SETString { ub-postalCode } OPTIONAL,  
    locationID  [3] EXPLICIT SETString { ub-locationID } OPTIONAL  
}

MerchantID ::= SETString { ub-MerchantID }
326  Secret ::= OCTET STRING (SIZE(20))
327
328  SWIdent ::= VisibleString (SIZE(1..ub-SWIdent)) -- Software identification
329
330  Thumbs ::= SEQUENCE {
331    digestAlgorithm AlgorithmIdentifier {{DigestAlgorithms}},
332    certThumbs [0] EXPLICIT Digests OPTIONAL,
333    crlThumbs [1] EXPLICIT Digests OPTIONAL,
334    brandCRLIdThumbs [2] EXPLICIT Digests OPTIONAL
335 }
336
337  TransIDs ::= SEQUENCE {
338    lid-C   LocalID,
339    lid-M   [0] LocalID OPTIONAL,
340    xid     XID,
341    pReqDate Date,
342    paySysID [1] PaySysID OPTIONAL,
343    language Language -- Cardholder requested session language
344 }
345
346  URL ::= VisibleString (SIZE(1..ub-URL)) -- Universal Resource Locator
347
348  XID ::= OCTET STRING (SIZE(20))
349
350  -- Upper bounds of SETString{} types
351
352  ub-BrandID INTEGER ::= 40
353  ub-MerchantID INTEGER ::= 30
354  ub-SWIdent INTEGER ::= 256
355  ub-acqBusinessID INTEGER ::= 32
356  ub-locationID INTEGER ::= 10
357  ub-paySysID INTEGER ::= 64
358  ub-RFC1766-language INTEGER ::= 35
359  ub-URL INTEGER ::= 512
360
361 END

362 SetCertMsgs
363   { joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 1}
364   DEFINITIONS IMPLICIT TAGS ::= BEGIN
365
366 --
367 -- Types used in the SET Certificate Management Protocol messages.
368 --
369
370 -- EXPORTS All;
371
372 IMPORTS
373
374   SETString {}, SignatureAlgorithms
375     FROM SetAttribute
376
377   SubjectPublicKeyInfo{}
378     FROM SetCertificate
379
380   BackKeyData, BIN, BrandCRLIdentifier, BrandID,
381    CertThumb, Challenge, Currency, Date, Language, LocalID, MerchantID,
382    Nonce, PAN, PANData0, RRPIID, Thumbs, ub-acqBusinessID, URL
383    FROM SetMessage
384
385    CA, EE, Enc {}, EncK {}, EncX {}, EXH {}, KeyEncryptionAlgorithms, L {},
386    S {}, SO {}
387    FROM SetPKCS7Plus;
388
389
390    -- Certificate Management Payload Components
391
392    AcctInfo ::= CHOICE {
393       panData0  [0] EXPLICIT PANData0,
394       acctData  [1] EXPLICIT AcctData
395    }
396
397    AcctData ::= SEQUENCE {
398       acctIdentification  AcctIdentification,
399       exNonce             Nonce
400    }
401
402    AcctIdentification ::= VisibleString (SIZE(ub-acctIdentification))
403
404    IDData ::= CHOICE { -- Merchants and Acquirers only
405       merchantAcquirerID  [0] MerchantAcquirerID,
406       acquirerID          [1] AcquirerID
407    }
408
409    MerchantAcquirerID ::= SEQUENCE {
410       merchantBIN  BIN,
411       merchantID   MerchantID    -- By prior agreement of Merchant/Acquirer
412    }
413
414    AcquirerID ::= SEQUENCE {
415       acquirerBIN         BIN,
416       acquirerBusinessID  AcquirerBusinessID  OPTIONAL
417    }
418
419    AcquirerBusinessID ::= NumericString (SIZE(1..ub-acqBusinessID))
420
421    RequestType ::= ENUMERATED { -- Indicates requestor and type of request
422       cardInitialSig    (1),
423       cardInitialEnc    (2),  Reserved
424       cardInitialBoth   (3),  Reserved
425       merInitialSig     (4),
426       merInitialEnc     (5),
427       merInitialBoth    (6),
428       pgwyInitialSig    (7),
429       pgwyInitialEnc    (8),
430       pgwyInitialBoth   (9),
431       cardRenewalSig   (10),
432       cardRenewalEnc   (11),  Reserved
433       cardRenewalBoth   (12),  Reserved
434       merRenewalSig   (13),
435       merRenewalEnc   (14),
436       merRenewalBoth   (15),
437       pgwyRenewalSig   (16),
438       pgwyRenewalEnc   (17),
SET Secure Electronic Transaction Specification

RegFormOrReferral ::= CHOICE {
    regFormData   [0] RegFormData,
    referralData  [1] ReferralData
}

RegFormData ::= SEQUENCE {
    regTemplate  RegTemplate  OPTIONAL,
    policy       PolicyText
}

RegTemplate ::= SEQUENCE {
    regFormID     INTEGER (0..MAX),    -- CA assigned identifier
    brandLogoURL  [0] URL OPTIONAL,
    cardLogoURL   [1] URL OPTIONAL,
    regFieldSeq   RegFieldSeq  OPTIONAL
}

RegFieldSeq ::= SEQUENCE SIZE(1..ub-FieldList) OF RegField

RegField ::= SEQUENCE {
    fieldId         [0] OBJECT IDENTIFIER  OPTIONAL,
    fieldName       FieldName,
    fieldDesc       [1] EXPLICIT SETString { ub-FieldDesc }  OPTIONAL,
    fieldLen        INTEGER (1..ub-FieldValue) DEFAULT ub-FieldValue,
    fieldRequired   [2] BOOLEAN DEFAULT FALSE,
    fieldInvisible  [3] BOOLEAN DEFAULT FALSE
}

ReferralData ::= SEQUENCE {
    reason          Reason  OPTIONAL,  -- Displayed on requestor's system
    referralURLSeq  ReferralURLSeq  OPTIONAL
} ( WITH COMPONENTS { ..., reason PRESENT } |
    WITH COMPONENTS { ..., referralURLSeq PRESENT } )

Reason ::= SETString { ub-Reason }

ReferralURLSeq ::= SEQUENCE OF ReferralURL   -- Ordered by preference

ReferralURL ::= URL

PolicyText ::= SETString { ub-PolicyText }

CardCInitReq ::= SEQUENCE {
    rrpid     RRPID,
    lid-EE    LocalID,
    chall-EE  Challenge,
    brandID   BrandID,
    thumbs    [0] EXPLICIT Thumbs  OPTIONAL
}

CardCInitRes ::= S { CA, CardCInitResTBS }

CardCInitResTBS ::= SEQUENCE {
497    rrpid               RRPID,
498    lid-EE              LocalID,
499    chall-EE            Challenge,
500    lid-CA              LocalID OPTIONAL,
501    caeThumb            [0] EXPLICIT CertThumb,
502    brandCRLIdentifier  [1] EXPLICIT BrandCRLIdentifier OPTIONAL,
503    thumbs              [2] EXPLICIT Thumbs OPTIONAL
504 }
505
506 -- Certificate Initialization Pair - Merchant or Payment Gateway
507
508 Me-AqCInitReq ::= SEQUENCE {
509    rrpid        RRPID,
510    lid-EE       LocalID,
511    chall-EE     Challenge,
512    requestType  RequestType,
513    idData       IDData,
514    brandID      BrandID,
515    language     Language,
516    thumbs       [0] EXPLICIT Thumbs OPTIONAL
517 }  
518
519 Me-AqCInitRes ::= S { CA, Me-AqCInitResTBS }
520
521 Me-AqCInitResTBS ::= SEQUENCE {
522    rrpid               RRPID,
523    lid-EE              LocalID,
524    chall-EE            Challenge,
525    lid-CA              [0] LocalID OPTIONAL,
526    chall-CA            Challenge,
527    requestType         RequestType,
528    regFormOrReferral   RegFormOrReferral,
529    acctDataField       [1] RegField OPTIONAL,
530    caeThumb            [2] EXPLICIT CertThumb,
531    brandCRLIdentifier  [3] EXPLICIT BrandCRLIdentifier OPTIONAL,
532    thumbs              [4] EXPLICIT Thumbs OPTIONAL
533 }
534
535 -- Registration Form Pair - Cardholder Only
536
537 RegFormReq ::= EXH { CA, RegFormReqData, PANOnly }
538
539 -- Intermediate results of EXH
540 RegFormReqTBE ::= L { RegFormReqData, PANOnly }
541
542 RegFormReqData ::= SEQUENCE {
543    rrpid        RRPID,
544    lid-EE       LocalID,
545    chall-EE2    Challenge,
546    lid-CA       [0] LocalID OPTIONAL,
547    requestType  RequestType,
548    language     Language,
549    thumbs       [1] EXPLICIT Thumbs OPTIONAL
550 } 
551
552 PANOnly ::= SEQUENCE {
553    pan       PAN,
554    exNonce   Nonce
555}
RegFormRes ::= S { CA, RegFormResTBS }

RegFormResTBS ::= SEQUENCE {
  rrpid               RRPID,
  lid-EE              LocalID,
  chall-EE2           Challenge,
  lid-CA              [0] LocalID  OPTIONAL,
  chall-CA            Challenge,
  caeThumb            [1] EXPLICIT CertThumb  OPTIONAL,
  requestType         RequestType,
  formOrReferral      RegFormOrReferral,
  brandCRLIdentifier  [2] EXPLICIT BrandCRLIdentifier  OPTIONAL,
  thumbs              [3] EXPLICIT Thumbs  OPTIONAL
}

-- Certificate Request Pair

CertReq ::= CHOICE {
  encx  [0] EXPLICIT EncX { EE, CA, CertReqData, AcctInfo },
  enc   [1] EXPLICIT Enc { EE, CA, CertReqData }
}

-- Intermediate results of Enc and EncX

CertReqTBE ::= S { EE, CertReqData }

CertReqTBEX ::= SEQUENCE {
  certReqData  CertReqData,
  s            SO { EE, CertReqTBS }
}

CertReqTBS ::= SEQUENCE {
  certReqData  CertReqData,
  acctInfo    AcctInfo
}

CertReqData ::= SEQUENCE {
  rrpid          RRPID,
  lid-EE         LocalID,
  chall-EE3      Challenge,
  lid-CA         [0] LocalID  OPTIONAL,
  chall-CA       [1] Challenge  OPTIONAL,
  requestType    RequestType,
  requestDate    Date,
  idData         [2] EXPLICIT IDData  OPTIONAL,
  regFormID      INTEGER (0..MAX),    -- CA assigned identifier
  regForm        [3] RegForm  OPTIONAL,
  caBackKeyData  [4] EXPLICIT BackKeyData  OPTIONAL,
  publicKeySorE  PublicKeySorE,
  eeThumb        [5] EXPLICIT CertThumb  OPTIONAL,
  thumbs         [6] EXPLICIT Thumbs  OPTIONAL
}

RegForm ::= SEQUENCE SIZE(1..ub-FieldList) OF RegFormItems

RegFormItems ::= SEQUENCE {
  fieldName   FieldName,
613   fieldValue  FieldValue
614 }
615
616 FieldName ::= SETString { ub-FieldName }
617
618 FieldValue ::= CHOICE {
619   setString    SETString { ub-FieldValue },
620   octetString  OCTET STRING (SIZE(1..ub-FieldValue))
621 }
622
623 PublicKeySorE ::= SEQUENCE {
624   publicKeyS  [0] EXPLICIT SubjectPublicKeyInfo({SignatureAlgorithms})
625                 OPTIONAL,
626   publicKeyE  [1] EXPLICIT SubjectPublicKeyInfo({KeyEncryptionAlgorithms})
627                 OPTIONAL
628 }  --
629   -- At least one component shall be present. A user may request a
630   -- signature certificate, an encryption certificate, or both.
631   --
632   { WITH COMPONENTS { ...}, publicKeyS PRESENT } |
633   WITH COMPONENTS { ...}, publicKeyE PRESENT }
634
635 CertRes ::= CHOICE {
636   certResTBS   [0] EXPLICIT S { CA, CertResData },
637   certResTBSK  [1] EXPLICIT EncK { CAKey, CA, CertResData }
638 }
639
640   -- Intermediate results of EncK
641 CertResTBE ::= S { CA, CertResData }
642
643 CertResData ::= SEQUENCE {
644   rrpid       RRPID,
645   lid-EE      LocalID,
646   chall-EE3   Challenge,
647   lid-CA      LocalID,
648   certStatus  CertStatus,
649   certThumbs  [0] EXPLICIT Thumbs OPTIONAL,
650   brandCRLIdentifier  [1] EXPLICIT BrandCRLIdentifier OPTIONAL,
651   thumbs      [2] EXPLICIT Thumbs OPTIONAL
652 }
653
654 CertStatus ::= SEQUENCE {
655   certStatusCode CertStatusCode,
656   nonceCCA     [0] Nonce OPTIONAL,
657   eeMessage    SETString { ub-eeMessage } OPTIONAL,
658   caMsg        [1] CAMsg OPTIONAL,
659   failedItemSeq [2] FailedItemSeq OPTIONAL
660 }
661
662 FailedItemSeq ::= SEQUENCE SIZE(1..ub-FieldList) OF FailedItem
663
664 FailedItem ::= SEQUENCE {
665   itemNumber   INTEGER (1..50),
666   itemReason   SETString { ub-Reason }
667 }
668
669 CertStatusCode ::= ENUMERATED {          -- In-process status of CertReq
670   requestComplete           (1),
671   ... }
CAMsg ::= SEQUENCE {
    cardLogoURL    [0] URL  OPTIONAL,
    brandLogoURL   [1] URL  OPTIONAL,
    cardCurrency   [2] Currency  OPTIONAL,
    cardholderMsg  [3] EXPLICIT
                           SETString { ub-cardholderMsg }  OPTIONAL
    }

CAKey ::= BackKeyData

-- Certificate Inquiry Pair

CertInqReq ::= S { EE, CertInqReqTBS }

CertInqReqTBS ::= SEQUENCE {
    rrpid      RRPID,
    lid-EE     LocalID,
    chall-EE3  Challenge,
    lid-CA     LocalID
    }

CertInqRes ::= CertRes

-- Upper bounds of SETString{} types

ub-acctIdentification  INTEGER ::=  74
ub-cardholderMsg       INTEGER ::=   128
ub-eeMessage           INTEGER ::=   128
ub-FieldDesc           INTEGER ::=   200
ub-FieldList           INTEGER ::=    50
ub-FieldName           INTEGER ::=   128
ub-FieldValue          INTEGER ::=   128
ub-PolicyText          INTEGER ::= 20000
ub-Reason              INTEGER ::=   512

SetPayMsgs

{ joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 2 }

DEFINITIONS IMPLICIT TAGS ::= BEGIN


This module defines types for SET protocol payment messages.

Exports All;

Imports

SETString {} FROM SetAttribute

EXTENSION FROM SetCertificateExtensions

BackKeyData, BIN, BrandCRLIdentifier, BrandID,
CertThumb, Challenge, Currency, Date, Language, LocalID,
Location, MerchantID, MsgExtensions {}, Nonce, PANData, PANToken,
Phone, RRPID, Secret, SWIdent, Thumbs, TransIDs, URL, XID
FROM SetMessage

C, DD {},
Enc {}, EncB {}, EncBX {}, EncK{}, EncX {}, EX {},
EXH {}, HMAC {}, L {}, M, P, P1, P2, S {}, SO {}
FROM SetPKCS7Plus

CommercialCardData, MarketAutoCap, MarketHotelCap, MarketTransportCap,
ub-reference
FROM SetMarketData;

Purchase Initialization Pair

PInitReq ::= SEQUENCE { -- Purchase Initialization Request
  rrpid           RRPID,
  language        Language,
  localID-C       LocalID,
  localID-M       [0] LocalID OPTIONAL,
  chall-C         Challenge,
  brandID         BrandID,
  bin             BIN,
  thumbs          [1] EXPLICIT Thumbs OPTIONAL,
}

PIrqExtensionsIOS EXTENSION ::= { ... }

PInitRes ::= S { M, PInitResData }

PInitResData ::= SEQUENCE {
  transIDs            TransIDs,
  rrpid               RRPID,
  chall-C              Challenge,
  chall-M              Challenge,
  brandCRLIdentifier  [0] EXPLICIT BrandCRLIdentifier OPTIONAL,
  peThumb             [1] EXPLICIT CertThumb,
  thumbs              [2] EXPLICIT Thumbs OPTIONAL,
  piRsExtensions      [3] MsgExtensions {{PIRsExtensionsIOS}} OPTIONAL
}

PIRsExtensionsIOS EXTENSION ::= { ... }
784
785 -- Purchase Pair
786
787 PReq ::= CHOICE {
788    pReqDualSigned [0] EXPLICIT PReqDualSigned,
789    pReqUnsigned    [1] EXPLICIT PReqUnsigned
790 }
791
792 -- Signed components used by a cardholder with a certificate
793
794 PReqDualSigned ::= SEQUENCE {
795    piDualSigned    PIDualSigned,
796    oiDualSigned    OIDualSigned
797 }
798
799 PIDualSigned ::= SEQUENCE {
800    piSignature     PISignature,
801    exPIData       EX { P, PI-OILink, PANData }
802 }
803
804 -- Intermediate results of EX
805 PIDualSignedTBE ::= L { PI-OILink, PANData }
806
807 PI-OILink ::= L { PIHead, OIData }
808
809 OIDualSigned ::= L { OIData, PIData }
810
811 PISignature ::= SO { C, PI-TBS }
812
813 PI-TBS ::= SEQUENCE {
814    hPIData      HPIData,
815    hOIData     HOIData
816 }
817
818 HPIData ::= DD { PIData }                         -- PKCS#7 DigestedData
819
820 HOIData ::= DD { OIData }                         -- PKCS#7 DigestedData
821
822 PI ::= CHOICE {
823    piUnsigned    [0] EXPLICIT PIUnsigned,
824    piDualSigned [1] EXPLICIT PIDualSigned,
825    authToken     [2] EXPLICIT AuthToken
826 }
827
828 PIData ::= SEQUENCE {
829    piHead   PIHead,
830    panData  PANData
831 }
832
833 PIHead ::= SEQUENCE {
834    transIDs    TransIDs,
835    inputs      Inputs,
836    merchantID  MerchantID,
837    installRecurData [0] InstallRecurData OPTIONAL,
838    transStain  TransStain,
839    swIdent     SWIdent,
840    acqBackKeyData [1] EXPLICIT BackKeyData OPTIONAL,
842}
PIExtensionsIOS EXTENSION ::= { ... }

Inputs ::= SEQUENCE {
  hod       HOD,
  purchAmt  CurrencyAmount
}

TransStain ::= HMAC { XID, Secret }

OIData ::= SEQUENCE {                          -- Order Information Data
  transIDs   TransIDs,
  rrpid      RRPID,
  chall-C    Challenge,
  hod        HOD,
  odSalt     Nonce,
  chall-M    Challenge OPTIONAL,
  brandID    BrandID,
  bin        BIN,
  odExtOIDs  [0] OIDList OPTIONAL,
  oiExtensions [1] MsgExtensions {{OIExtensionsIOS}} OPTIONAL
}

ODExtensionsIOS EXTENSION ::= { ... }

OIDList ::= SEQUENCE OF OBJECT IDENTIFIER

HOD ::= DD { HODInput }

HODInput ::= SEQUENCE {
  od          OD,
  purchAmt    CurrencyAmount,
  odSalt      Nonce,
  installRecurData [0] InstallRecurData OPTIONAL,
  odExtensions [1] MsgExtensions {{ODExtensionsIOS}} OPTIONAL
}

ODExtensionsIOS EXTENSION ::= { ... }

OD ::= OCTET STRING                                 -- Order description

-- Unsigned components used by a cardholder without a certificate

PReqUnsigned ::= SEQUENCE {  -- Sent by cardholders without certificates
  piUnsigned  PIUnsigned,
  oiUnsigned  OIUnsigned
}

OIUnsigned ::= L { OIData, PIDataUnsigned }

PIDataUnsigned ::= SEQUENCE {
  piHead   PIHead,
  panToken PANToken
}

PIUnsigned ::= EXH { P, PI-OILink, PANToken }

-- Order description

-- Unsigned components used by a cardholder without a certificate
900 -- Intermediate results of EXH
901 PIUnsignedTBE ::= L { PI-OILink, PANToken }
902
903 PRes ::= S { M, PResData }
904
905 PResData ::= SEQUENCE {
906   transIDs            TransIDs,
907   rrpid               RRPIID,
908   chall-C             Challenge,
909   brandCRLIdentifier [0] EXPLICIT BrandCRLIdentifier  OPTIONAL,
910   pResPayloadSeq      PResPayloadSeq
911 }
912
913 PResPayloadSeq ::= SEQUENCE SIZE(1..MAX) OF PResPayload
914
915 PResPayload ::= SEQUENCE {
916   completionCode  CompletionCode,
917   results         Results  OPTIONAL,
918   pRsExtensions   [0] MsgExtensions {{PRsExtensionsIOS}} OPTIONAL
919 }
920
921 PRsExtensionsIOS EXTENSION ::= { ... }
922
923 CompletionCode ::= ENUMERATED {
924   meaninglessRatio        (0),  -- PurchAmt = 0; ratio cannot be computed
925   orderRejected           (1),  -- Merchant cannot process order
926   orderReceived           (2),  -- No processing to report
927   orderNotReceived        (3),  -- InqReq received without PReq
928   authorizationPerformed  (4),  -- See AuthStatus for details
929   capturePerformed        (5),  -- See CapStatus for details
930   creditPerformed         (6)   -- See CreditStatus for details
931 }
932
933 Results ::= SEQUENCE {
934   acqCardMsg   [0] EXPLICIT AcqCardMsg  OPTIONAL,
935   authStatus   [1] AuthStatus  OPTIONAL,
936   capStatus    [2] CapStatus  OPTIONAL,
937   credStatusSeq  [3] CreditStatusSeq  OPTIONAL
938 }
939
940 AuthStatus ::= SEQUENCE {
941   authDate   Date,
942   authCode   AuthCode,
943   authRatio  FloatingPoint,
944   currConv   [0] CurrConv  OPTIONAL
945 }
946
947 CapStatus ::= SEQUENCE {
948   capDate   Date,
949   capCode   CapCode,
950   capRatio  FloatingPoint
951 }
952
953 CreditStatusSeq ::= SEQUENCE SIZE(1..MAX) OF CreditStatus
954
955 CreditStatus ::= SEQUENCE {
956   creditDate   Date,
957   creditCode   CapRevOrCredCode,
creditRatio  FloatingPoint

-- Purchase Inquiry Pair

InqReq ::= CHOICE {
  inqReqSigned  [0] EXPLICIT InqReqSigned,
  inqReqUnsigned [1] EXPLICIT InqReqData
}

InqReqSigned ::= S { C, InqReqData }

InqReqData ::= SEQUENCE {
  -- Signed by cardholder, if signed
  transIDs  TransIDs,
  rrpid     RRPID,
  chall-C2  Challenge,
  inqRqExtensions [0] MsgExtensions {{InqRqExtensionsIOS}}  OPTIONAL
}

InqRqExtensionsIOS EXTENSION ::= { ... }

InqRes ::= PRes

-- Authorization Pair

AuthReq ::= EncB { M, P, AuthReqData, PI }

-- Intermediate results of EncB

AuthReqTBE ::= S { M, AuthReqTBS }

AuthReqTBS ::= L { AuthReqData, PI }

AuthReqData ::= SEQUENCE {
  authReqItem  AuthReqItem,
  mThumbs      [0] EXPLICIT Thumbs  OPTIONAL,
  captureNow   BOOLEAN DEFAULT FALSE,
  saleDetail   [1] SaleDetail  OPTIONAL
} ( WITH COMPONENTS {..., captureNow (TRUE) } | WITH COMPONENTS {..., captureNow (FALSE), saleDetail ABSENT } )

AuthReqItem ::= SEQUENCE {
  authTags        AuthTags,
  checkDigests    [0] CheckDigests OPTIONAL,
  authReqPayload  AuthReqPayload
}

AuthTags ::= SEQUENCE {
  authRRTags  RRTags,
  transIDs    TransIDs,
  authRetNum  AuthRetNum  OPTIONAL
}

CheckDigests ::= SEQUENCE {
  hOIData  HOIData,
  hod2     HOD
}

AuthReqPayload ::= SEQUENCE {
SET Secure Electronic Transaction Specification

May 31, 1997

Book 3: Formal Protocol Definition

Page 197

1016    subsequentAuthInd   BOOLEAN DEFAULT FALSE,
1017    authReqAmt CurrencyAmount, -- May differ from PurchAmt
1018    avsData [0] AVSData OPTIONAL,
1019    specialProcessing [1] SpecialProcessing OPTIONAL,
1020    cardSuspect [2] CardSuspect OPTIONAL,
1021    requestCardTypeInd BOOLEAN DEFAULT FALSE,
1022    installRecurData [3] InstallRecurData OPTIONAL,
1023    marketSpecAuthData [4] EXPLICIT MarketSpecAuthData OPTIONAL,
1024    merchData MerchData,
1025    aRqExtensions [5] MsgExtensions {{ARqExtensionsIOS}} OPTIONAL
1026 }
1027
1028 ARqExtensionsIOS EXTENSION ::= { ... }
1029
1030 AVSData ::= SEQUENCE {
1031    streetAddress SETString { ub-AVSData } OPTIONAL,
1032    location Location
1033 }
1034
1035 SpecialProcessing ::= ENUMERATED {
1036    directMarketing (0),
1037    preferredCustomer (1)
1038 }
1039
1040 CardSuspect ::= ENUMERATED { -- Indicates merchant suspects cardholder
1041    --
1042    -- Specific values indicate why the merchant is suspicious
1043    --
1044    unspecifiedReason (0) -- Either the merchant does not differentiate
1045    -- reasons for suspicion, or the specific
1046    -- reason does not appear in the list
1047 }
1048
1049 MerchData ::= SEQUENCE {
1050    merchCatCode MerchCatCode OPTIONAL,
1051    merchGroup MerchGroup OPTIONAL
1052 }
1053
1054 MerchCatCode ::= NumericString (SIZE(ub-merType)) -- ANSI X9.10
1055    -- Merchant Category Code (MCCs) are assigned by acquirer to
1056    -- describe the merchant's product, service or type of business
1057
1058 MerchGroup ::= ENUMERATED {
1059    commercialTravel (1),
1060    lodging (2),
1061    automobileRental (3),
1062    restaurant (4),
1063    medical (5),
1064    mailOrPhoneOrder (6),
1065    riskyPurchase (7),
1066    other (8)
1067 }
1068
1069 AuthRes ::= CHOICE {
1072 }
1073

Version 1.0
1074 -- Intermediate results of EncB and EncBX
1075 AuthResTBE ::= S { P, AuthResTBS }
1076
1077 AuthResTBEX ::= SEQUENCE {
1078   authResTBS   AuthResTBS,
1079   s           SO { P, AuthResTBSX }
1080 }
1081
1082 AuthResTBS ::= L { AuthResData, AuthResBaggage}
1083
1084 AuthResTBSX ::= SEQUENCE {
1085   authResTBS   AuthResTBS,
1086   panToken    PANToken
1087 }
1088
1089 AuthResData ::= SEQUENCE {
1090   authTags            AuthTags,
1091   brandCRLIdentifier [0] EXPLICIT BrandCRLIdentifier OPTIONAL,
1092   pThumb             [1] EXPLICIT CertThumb OPTIONAL,
1093   authResPayload    AuthResPayload
1094 }
1095
1096 AuthResBaggage ::= SEQUENCE {
1097   capToken    [0] EXPLICIT CapToken OPTIONAL,
1098   acqCardMsg  [1] EXPLICIT AcqCardMsg OPTIONAL,
1099   authToken   [2] EXPLICIT AuthToken OPTIONAL
1100 }
1101
1102 AcqBackKey ::= BackKeyData
1103
1104 AcqCardMsg ::= EncK { AcqBackKey, P, AcqCardCodeMsg }
1105
1106 -- Intermediate result of EncK
1107 AcqCardCodeMsgTBE ::= S { P, AcqCardCodeMsg }
1108
1109 AcqCardCodeMsg ::= SEQUENCE {
1110   acqCardCode   AcqCardCode,
1111   acqCardMsgData AcqCardMsgData
1112 }
1113
1114 AcqCardCode ::= ENUMERATED {
1115   messageOfDay         (0),
1116   accountInfo          (1),
1117   callCustomerService  (2)
1118 }
1119
1120 AcqCardMsgData ::= SEQUENCE {
1121   acqCardText   [0] EXPLICIT SETString { ub-acqCardText } OPTIONAL,
1122   acqCardURL    [1] URL OPTIONAL,
1123   acqCardPhone  [2] EXPLICIT SETString { ub-acqCardPhone } OPTIONAL
1124 }
1125
1126 AuthResPayload ::= SEQUENCE {
1127   authHeader     AuthHeader,
1128   capResPayload CapResPayload OPTIONAL,
1129   aRsExtensions  [0] MsgExtensions {{ARsExtensionsIOS}} OPTIONAL
1130 }
1131
AuthHeader ::= SEQUENCE {
  authAmt      CurrencyAmount,
  authCode     AuthCode,
  batchStatus  [0] BatchStatus  OPTIONAL,
  currConv     CurrConv  OPTIONAL          -- Merchant to cardholder
}

AuthCode ::= ENUMERATED {
  approved    ( 0),
  unspecifiedFailure   ( 1),
  declined   ( 2),
  noReply    ( 3),
  callIssuer ( 4),
  amountError     ( 5),
  expiredCard     ( 6),
  invalidTransaction ( 7),
  systemError    ( 8),
  piPreviouslyUsed ( 9),
  recurringTooSoon (10),
  recurringExpired (11),
  piAuthMismatch     (12),
  installRecurMismatch (13),
  captureNotSupported (14),
  signatureRequired (15),
  cardMerchBrandMismatch (16)
}

ResponseData ::= SEQUENCE {
  authValCodes  [0] AuthValCodes  OPTIONAL,
  respReason    [1] RespReason  OPTIONAL,
  cardType      CardType  OPTIONAL,
  avsResult     [2] AVSResult  OPTIONAL,
  logRefID      LogRefID  OPTIONAL
}

AuthValCodes ::= SEQUENCE {
  approvalCode    [0] ApprovalCode  OPTIONAL,
  authCharInd     [1] AuthCharInd  OPTIONAL,
  validationCode  [2] ValidationCode  OPTIONAL,
  marketSpec     MarketSpecDataID  OPTIONAL
}

RespReason ::= ENUMERATED {
  issuer                    (0),
  standInTimeOut            (1),
  standInFloorLimit         (2),
  standInSuppressInquiries  (3),
  standInIssuerUnavailable (4),
  standInIssuerRequest     (5)
}

CardType ::= ENUMERATED {
  unavailable            ( 0),
  classic                ( 1),
  gold                   ( 2),
  ...
AVSResult ::= ENUMERATED {
    resultUnavailable (0),
    noMatch (1),
    addressMatchOnly (2),
    postalCodeMatchOnly (3),
    fullMatch (4)
}

LogRefID ::= NumericString (SIZE(1..ub-logRefID))

ApprovalCode ::= VisibleString (SIZE(ub-approvalCode))

AuthCharInd ::= ENUMERATED {
    directMarketing (0),
    recurringPayment (1),
    addressVerification (2),
    preferredCustomer (3),
    incrementalAuth (4)
}

ValidationCode ::= VisibleString (SIZE(ub-validationCode))

-- Auth Reversal Pair

AuthRevReq ::= EncB { M, P, AuthRevReqData, AuthRevReqBaggage }

-- Intermediate results of EncB

AuthRevReqTBE ::= S { M, AuthRevReqTBS }

AuthRevReqTBS ::= L { AuthRevReqData, AuthRevReqBaggage }

AuthRevReqData ::= SEQUENCE {
    authRevTags  AuthRevTags,
    mThumbs [0] EXPLICIT Thumbs  OPTIONAL,
    authReqData [1] AuthReqData  OPTIONAL,
    authNewAmt  CurrencyAmount,
    aRvRqExtensions [3] MsgExtensions {{ARvRqExtensionsIOS}}  OPTIONAL
}

ARvRqExtensionsIOS EXTENSION ::= { ... }

AuthRevReqBaggage ::= SEQUENCE {
1248     pi        PI,
1249     capToken  CapToken OPTIONAL
1250 )
1251
1252 AuthRevTags ::= SEQUENCE {
1253     authRevRRTags AuthRevRRTags,
1254     authRetNum   AuthRetNum OPTIONAL
1255 )
1256
1257 AuthRevRRTags ::= RRTags
1258
1259 AuthRetNum ::= INTEGER (0..MAX)
1260
1261 AuthRevRes := CHOICE {
1264 }
1265
1266 -- Intermediate results of Enc and EncB
1267 AuthRevResTBE ::= S { P, AuthRevResData }
1268
1269 AuthRevResTBEB ::= S { P, AuthRevResTBS }
1270
1272
1273 AuthRevResBaggage ::= SEQUENCE {
1274     capTokenNew   CapToken OPTIONAL,
1275     authTokenNew  AuthToken  OPTIONAL
1276 }
1277
1278 AuthRevResData ::= SEQUENCE {
1279     authRevCode    AuthRevCode,
1280     authRevTags   AuthRevTags,
1281     brandCRLIdentifier [0] EXPLICIT BrandCRLIdentifier OPTIONAL,
1282     peThumb       [1] EXPLICIT CertThumb OPTIONAL,
1283     authNewAmt    CurrencyAmount, -- May be zero
1284     authResDataNew AuthResDataNew,
1285     aRvRsExtensions [2] MsgExtensions {{ARvRsExtensionsIOS}} OPTIONAL
1286 }
1287
1288 ARvRsExtensionsIOS EXTENSION ::= {
1289
1290     AuthRevCode ::= ENUMERATED {
1291         approved     ( 0),
1292         unspecifiedFailure     ( 1),
1293         noReply      ( 2),
1294         amountError    ( 3),
1295         expiredCard    ( 4),
1296         invalidTransaction ( 5),
1297         systemError    ( 6),
1298         missingCapToken ( 7),
1299         invalidCapToken ( 8),
1300         invalidAmount    ( 9)
1301 }
1302
1303 AuthResDataNew ::= SEQUENCE {
1304     transIDs    TransIDs,
1305     authResPayloadNew AuthResPayload OPTIONAL -- Contains new data
CapReq ::= CHOICE {
  encB   [0] EXPLICIT EncB { M, P, CapReqData, CapTokenSeq },
  encBX  [1] EXPLICIT EncBX { M, P, CapReqData, CapTokenSeq, PANToken }
}

CapReqTBE ::= S { M, CapReqTBS }

CapReqTBS ::= L { CapReqData, CapTokenSeq }

CapReqTBSX ::= SEQUENCE {
  capReqTBS  CapReqTBS,
  s          SO { M, CapReqTBSX }
}

CapReqData ::= SEQUENCE {
  capRRTags      CapRRTags,
  mThumbs        [0] EXPLICIT Thumbs  OPTIONAL,
  capItemSeq     CapItemSeq,
  cRqExtensions  [1] MsgExtensions {{CRqExtensionsIOS}} OPTIONAL
}

CRqExtensionsIOS EXTENSION ::= { ... }

CapRRTags ::= RRTags

CapItemSeq ::= SEQUENCE SIZE(1..MAX) OF CapItem

CapItem ::= SEQUENCE {
  transIDs    TransIDs,
  authRRPID   RRPID,
  capPayload  CapPayload
}

CapPayload ::= SEQUENCE {
  capDate         Date,
  capReqAmt       CurrencyAmount,
  authReqItem     [0] AuthReqItem OPTIONAL,
  saleDetail      [2] SaleDetail  OPTIONAL,
}

CPayExtensionsIOS EXTENSION ::= { ... }

CapRes ::= Enc { P, M, CapResData }

CapResTBE ::= S { P, CapResData }

Intermediate results of Enc and EncBX
CapResData ::= SEQUENCE {
    capRRTags CapRRTags,
    brandCRLIdentifier [0] EXPLICIT BrandCRLIdentifier OPTIONAL,
    peThumb [1] EXPLICIT CertThumb OPTIONAL,
    batchStatusSeq [2] BatchStatusSeq OPTIONAL,
    capResItemSeq CapResItemSeq,
}

CRsExtensionsIOS EXTENSION ::= { ... }

CapResItemSeq ::= SEQUENCE SIZE(1..MAX) OF CapResItem

CapResItem ::= SEQUENCE {
    transIDs TransIDs,
    authRRPID RRPID,
    capResPayload CapResPayload
}

CapResPayload ::= SEQUENCE {
    capCode CapCode,
    capAmt CurrencyAmount,
    batchID [0] BatchID OPTIONAL,
    batchSequenceNum [1] BatchSequenceNum OPTIONAL,
}

CRsPayExtensionsIOS EXTENSION ::= { ... }

CapCode ::= ENUMERATED {
    success (0),
    unspecifiedFailure (1),
    duplicateRequest (2),
    authExpired (3),
    authDataMissing (4),
    invalidAuthData (5),
    capTokenMissing (6),
    invalidCapToken (7),
    batchUnknown (8),
    batchClosed (9),
    unknownXID (10),
    unknownLID (11)
}

-- Capture Reversal Or Credit

CapRevOrCredReqData ::= SEQUENCE {
    capRevOrCredRRTags RRTags,
    mThumbs [0] EXPLICIT Thumbs OPTIONAL,
    capRevOrCredReqItemSeq CapRevOrCredReqItemSeq,
    crvRqExtensions [1] MsgExtensions {{CRvRqExtensionsIOS}} OPTIONAL
}

CRvRqExtensionsIOS EXTENSION ::= { ... }

CapRevOrCredReqItemSeq ::= SEQUENCE SIZE(1..MAX) OF CapRevOrCredReqItem
CapRevOrCredReqItem ::= SEQUENCE {
  transIDs TransIDs,
  authRRPID RRPID,
  capPayload CapPayload,
  newBatchID [0] BatchID OPTIONAL,
  capRevOrCredReqDate Date,
  capRevOrCredReqAmt [1] CurrencyAmount OPTIONAL,
  newAccountInd BOOLEAN DEFAULT FALSE,
  crvRqItemExtensions [2] MsgExtensions {{CRvRqItemExtensionsIOS}} OPTIONAL
}

CRvRqItemExtensionsIOS EXTENSION ::= { ... }

CapRevOrCredResData ::= SEQUENCE {
  capRevOrCredRRTags RRTags,
  brandCRLIdentifier [0] EXPLICIT BrandCRLIdentifier OPTIONAL,
  peThumb [1] EXPLICIT CertThumb OPTIONAL,
  batchStatusSeq [2] BatchStatusSeq OPTIONAL,
  capRevOrCredResItemSeq CapRevOrCredResItemSeq,
}

CRvRsExtensionsIOS EXTENSION ::= { ... }

CapRevOrCredResItemSeq ::= SEQUENCE SIZE(1..MAX) OF CapRevOrCredResItem

CapRevOrCredResItem ::= SEQUENCE {
  transIDs TransIDs,
  authRRPID RRPID,
  capRevOrCredResPayload CapRevOrCredResPayload
}

CapRevOrCredResPayload ::= SEQUENCE {
  capRevOrCredCode CapRevOrCredCode,
  capRevOrCredActualAmt CurrencyAmount,
  batchID [0] BatchID OPTIONAL,
  batchSequenceNum [1] BatchSequenceNum OPTIONAL,
}

CRvRsPayExtensionsIOS EXTENSION ::= { ... }

CapRevOrCredCode ::= ENUMERATED {
  success (0),
  unspecifiedFailure (1),
  duplicateRequest (2),
  originalProcessed (3),
  originalNotFound (4),
  capPurged (5),
  capDataMismatch (6),
  missingCapData (7),
  missingCapToken (8),
  invalidCapToken (9),
  batchUnknown (10),
  batchClosed (11)
}

-- Capture Reversal Pair
CapRevReq ::= CHOICE {
  encB   [0] EXPLICIT EncB { M, P, CapRevData, CapTokenSeq },
  encBX  [1] EXPLICIT EncBX { M, P, CapRevData, CapTokenSeq, PANToken }
}

-- Intermediate results of EncB and EncBX
CapRevReqTBE ::= S { M, CapRevReqTBS }

CapRevReqTBEX ::= SEQUENCE {
  capRevReqTBS  CapRevReqTBS,
  s             SO { M, CapRevReqTBSX }
}

CapRevReqTBS ::= L { CapRevData, CapTokenSeq }

CapRevReqTBSX ::= SEQUENCE {
  capRevReqTBS  CapRevReqTBS,
  panToken      PANToken
}

CapRevData ::= [0] EXPLICIT CapRevOrCredReqData

CapRevRes ::= Enc { P, M, CapRevResData }

-- Intermediate results of Enc
CapRevResTBE ::= S { P, CapRevResData }

CapRevResData ::= [0] EXPLICIT CapRevOrCredResData

-- Credit Pair

CredReq ::= CHOICE {
  encB   [0] EXPLICIT EncB { M, P, CredReqData, CapTokenSeq },
  encBX  [1] EXPLICIT EncBX { M, P, CredReqData, CapTokenSeq, PANToken }
}

-- Intermediate results of EncB and EncBX
CredReqTBE ::= S { M, CredReqTBS }

CredReqTBEX ::= SEQUENCE {
  credReqTBS  CredReqTBS,
  s             SO { M, CredReqTBSX }
}

CredReqTBS ::= L { CredReqData, CapTokenSeq }

CredReqTBSX ::= SEQUENCE {
  credReqTBS  CredReqTBS,
  panToken      PANToken
}

CredReqData ::= [1] EXPLICIT CapRevOrCredReqData

CredRes ::= Enc { P, M, CredResData }

-- Intermediate results of Enc
CredResTBE ::= S { P, CredResData }

CredResData ::= [0] EXPLICIT CapRevOrCredResData
CredResData ::= [1] EXPLICIT CapRevOrCredResData

CredRevReq ::= CHOICE {
  encB  [0] EXPLICIT EncB { M, P, CredRevReqData, CapTokenSeq },
  encBX [1] EXPLICIT EncBX { M, P, CredRevReqData, CapTokenSeq, PANToken }
}

-- Intermediate results of EncB and EncBX

CredRevReqTBE ::= S { M, CredRevReqTBS }

CredRevReqTBEX ::= SEQUENCE {
  credRevReqTBS  CredRevReqTBS,
  s              SO { M, CredRevReqTBSX }
}

CredRevReqTBS ::= L { CredRevReqData, CapTokenSeq }

CredRevReqTBSX ::= SEQUENCE {
  credRevReqTBS  CredRevReqTBS,
  panToken       PANToken
}

CredRevReqData ::= [2] EXPLICIT CapRevOrCredReqData

CredRevRes ::= Enc { P, M, CredRevResData }

CredRevResTBE ::= S { P, CredRevResData }


PCertReq ::= S { M, PCertReqData }

PCertReqData ::= SEQUENCE {
  pCertRRTags    RRTags,
  mThumbs        [0] EXPLICIT Thumbs  OPTIONAL,
  brandAndBINSeq BrandAndBINSeq,
  pcRqExtensions [1] MsgExtensions {{PCRqExtensionsIOS}}  OPTIONAL
}

PCRqExtensionsIOS EXTENSION ::= { ... }

BrandAndBINSeq ::= SEQUENCE SIZE(1..MAX) OF BrandAndBIN

BrandAndBIN ::= SEQUENCE {
  brandID  BrandID,
  bin      BIN  OPTIONAL
}

PCertRes ::= S { P, PCertResTBS }

PCertResTBS ::= SEQUENCE {
  pCertRRTags    RRTags,
PCRsExtensionsIOS EXTENSION ::= { ... }

PCertResItemSeq ::= SEQUENCE OF PCertResItem

PCertResItem ::= SEQUENCE {
    pCertCode  PCertCode,
    certThumb  [0] EXPLICIT CertThumb  OPTIONAL
}

PCertCode ::= ENUMERATED {
    success             (0),
    unspecifiedFailure  (1),
    brandNotSupported   (2),
    unknownBIN          (3)
}

BrandCRLIdentifierSeq ::= SEQUENCE SIZE(1..MAX) OF [0] EXPLICIT BrandCRLIdentifier

-- Batch Administration Pair

BatchAdminReq ::= Enc { M, P, BatchAdminReqData }

BatchAdminReqTBE ::= S { M, BatchAdminReqData }

BatchAdminReqData ::= SEQUENCE {
    batchAdminRRTags         RRTags,
    batchID                  [0] BatchID  OPTIONAL,
    brandAndBINSeq           [1] BrandAndBINSeq  OPTIONAL,
    batchOperation           [2] BatchOperation  OPTIONAL,
    returnBatchSummaryInd    BOOLEAN DEFAULT FALSE,
    returnTransactionDetail  [3] ReturnTransactionDetail  OPTIONAL,
    batchStatus              [4] BatchStatus  OPTIONAL,
    transDetails             [5] TransDetails  OPTIONAL,
    baRqExtensions           [6] MsgExtensions {{BARqExtensionsIOS}} OPTIONAL
}

BARqExtensionsIOS EXTENSION ::= { ... }

BatchOperation ::= ENUMERATED {
    open   (0),
    purge  (1),
    close  (2)
}

ReturnTransactionDetail ::= SEQUENCE {
    startingPoint  INTEGER (MIN..MAX),
    maximumItems   INTEGER (1..MAX),
    errorsOnlyInd  BOOLEAN DEFAULT FALSE,
    brandID        [0] EXPLICIT BrandID  OPTIONAL
}

Version 1.0
TransDetails ::= SEQUENCE {
    nextStartingPoint INTEGER (MIN..MAX),
    transactionDetailSeq TransactionDetailSeq
}

BatchAdminRes ::= Enc { P, M, BatchAdminResData }

-- Intermediate results of Enc
BatchAdminResTBE ::= S { P, BatchAdminResData }

BatchAdminResData ::= SEQUENCE {
    batchAdminTags RRTags,
    batchID BatchID,
    baStatus BAStatus OPTIONAL,
    batchStatus [0] BatchStatus OPTIONAL,
    transmissionStatus [1] TransmissionStatus OPTIONAL,
    settlementInfo [2] SettlementInfo OPTIONAL,
    transDetails [3] TransDetails OPTIONAL,
    baRsExtensions [4] MsgExtensions {{BARsExtensionsIOS}} OPTIONAL
}

BARsExtensionsIOS EXTENSION ::= { ... }

TransmissionStatus ::= ENUMERATED {
    pending (0),
    inProgress (1),
    batchRejectedByAcquirer (2),
    completedSuccessfully (3),
    completedWithItemErrors (4)
}

SettlementInfo ::= SEQUENCE {
    settlementAmount CurrencyAmount,
    settlementType AmountType,
    settlementAccount SETString { ub-SettlementAccount },
    settlementDepositDate Date
}

BAStatus ::= ENUMERATED {
    success (0),
    unspecifiedFailure (1),
    brandNotSupported (2),
    unknownBIN (3),
    batchIDUnavailable (4),
    batchAlreadyOpen (5),
    unknownBatchID (6),
    brandBatchMismatch (7),
    totalsOutOfBalance (8),
    unknownStartingPoint (9),
    stopItemDetail (10),
    unknownBatchOperation (11)
}

ClosedWhen ::= SEQUENCE {
    closeStatus CloseStatus,
    closeDateTime Date
}
Set Secure Electronic Transaction Specification

Version 1.0

1711 CloseStatus ::= ENumerated {
1712         closedbyMerchant (0),
1713         closedbyAcquirer (1)
1714 }
1715
1716 BatchStatusSeq ::= SEQUENCE OF BatchStatus
1717
1718 BatchStatus ::= SEQUENCE {
1719         openDateTime     Date,
1720         closedWhen       [0] ClosedWhen  OPTIONAL,
1721         batchDetails     BatchDetails,
1722         batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL
1723 }
1724
1725 BSExtensionsIOS EXTENSION ::= { ... }
1726
1727 BatchDetails ::= SEQUENCE {
1728         batchTotals     BatchTotals,
1729         brandBatchDetailsSeq  BrandBatchDetailsSeq  OPTIONAL
1730 }
1731
1732 BrandBatchDetailsSeq ::= SEQUENCE SIZE(1..MAX) OF BrandBatchDetails
1733
1734 BrandBatchDetails ::= SEQUENCE {
1735         brandID      BrandID,
1736         batchTotals  BatchTotals
1737 }
1738
1739 BatchTotals ::= SEQUENCE {
1740         transactionCountCredit     INTEGER (0..MAX),
1741         transactionTotalAmtCredit  CurrencyAmount,
1742         transactionCountDebit      INTEGER (0..MAX),
1743         transactionTotalAmtDebit   CurrencyAmount,
1744         batchTotalExtensions       [0] MsgExtensions {{BTExtensionsIOS}} OPTIONAL
1745 }
1746
1747 BTExtensionsIOS EXTENSION ::= { ... }
1748
1749 TransactionDetailSeq ::= SEQUENCE OF TransactionDetail
1750
1751 TransactionDetail ::= SEQUENCE {
1752         transIDs            TransIDs,
1753         authRRPID           RRPID,
1754         brandID             BrandID,
1755         batchSequenceNum    BatchSequenceNum,
1756         reimbursementID     ReimbursementID  OPTIONAL,
1757         transactionAmt      CurrencyAmount,
1758         transactionAmtType  AmountType,
1759         transactionStatus   [0] TransactionStatus  OPTIONAL,
1760         transExtensions      [1] MsgExtensions {{TransExtensionsIOS}} OPTIONAL
1761 }
1762
1763 TransExtensionsIOS EXTENSION ::= { ... }
1764
1765 AmountType ::= ENumerated {
1766         credit  (0),
1767         debit   (1)
1768 }
1769 TransactionStatus ::= ENumerated {
1770 success             (0),
1771 unspecifiedFailure  (1)
1772 }
1773
1774 ReimbursementID ::= ENumerated {
1775 unspecified     (0),
1776 standard        (1),
1777 keyEntered      (2),
1778 electronic      (3),
1779 additionalData  (4),
1780 enhancedData    (5),
1781 marketSpecific  (6)
1782 }
1783
1784 -- Payment Message Components
1785
1786 AuthToken  ::= EncX { P1, P2, AuthTokenData, PANToken }
1787
1788 -- Intermediate results of EncX
1789 AuthTokenTBE ::= SEQUENCE {
1790 authTokenData  AuthTokenData,
1791 s              SO { P1, AuthTokenTBS }
1792 }
1793
1794 AuthTokenTBS ::= SEQUENCE {
1795 authTokenData  AuthTokenData,
1796 panToken       PANToken
1797 }
1798
1799 AuthTokenData ::= SEQUENCE {
1800 transIDs          TransIDs,
1801 purchAmt          CurrencyAmount,
1802 merchantID        MerchantID,
1803 acqBackKeyData    BackKeyData OPTIONAL,
1804 installRecrData   [0] InstallRecrData OPTIONAL,
1805 recurringCount    [1] INTEGER (1..MAX) OPTIONAL,
1806 prevAuthDateTime  Date,
1807 totalAuthAmount   [2] CurrencyAmount OPTIONAL,
1808 authTokenOpaque   [3] EXPLICIT TokenOpaque OPTIONAL
1809 }
1810
1811 BatchID ::= INTEGER (0..MAX)
1812
1813 BatchSequenceNum ::= INTEGER (1..MAX)
1814
1815 CapToken ::= CHOICE {
1816 encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
1817 enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
1818 null  [2] EXPLICIT NULL
1819 }
1820
1821 -- Intermediate results of Enc and EncX
1822 CapTokenTBE ::= S { P1, CapTokenData }
1823
1824 CapTokenTBEX ::= SEQUENCE {
1825 capTokenData CapTokenData,
CapTokenTBS ::= SEQUENCE {
  capTokenData  CapTokenData,
  panToken      PANToken
}

CapTokenData ::= SEQUENCE {
  authRRPID    RRPID,
  authAmt      CurrencyAmount,
  tokenOpaque  TokenOpaque
}

CapTokenSeq ::= SEQUENCE SIZE(1..MAX) OF CapToken

CurrencyAmount ::= SEQUENCE {
  currency  Currency, -- Currency code as defined in ISO-4217
  amount    INTEGER (0..MAX),
  amtExp10  INTEGER (MIN..MAX)
                        -- Base ten exponent, such that the value in local
                        -- currency is "amount * (10 ** amtExp10)"
                        -- The exponent shall be the same value as defined
                        -- for the minor unit of currency in ISO-4217.
}

CurrConv ::= SEQUENCE {
  currConvRate  FloatingPoint,
  cardCurr      Currency
}

FloatingPoint ::= REAL (WITH COMPONENTS {..., base (2)})

MarketAutoAuth ::= SEQUENCE {
  duration  Duration
}

MarketHotelAuth ::= SEQUENCE {
  duration  Duration,
  prestige  Prestige  OPTIONAL
}

Duration ::= INTEGER (1..99)                           -- Number of days

Prestige ::= ENUMERATED {
  unknown (0),
  level-1 (1), -- Transaction floor limits for each level are
  level-2 (2), -- defined by brand policy and may vary between
  level-3 (3) -- national markets.
}

MarketSpecAuthData ::= CHOICE {
  auto-rental [0] MarketAutoAuth,
  hotel        [1] MarketHotelAuth,
}

MarketSpecCapData ::= CHOICE {
auto-rental  [0] MarketAutoCap,
hotel        [1] MarketHotelCap,
}

MarketSpecSaleData ::= SEQUENCE {
  marketSpecDataID   MarketSpecDataID OPTIONAL,
  marketSpecCapData  MarketSpecCapData OPTIONAL
}

MarketTransportAuth ::= NULL

MarketSpecDataID ::= ENUMERATED {
  failedEdit  (0),
  auto        (1),
  hotel       (2),
  transport   (3)
}

MerOrderNum ::= VisibleString (SIZE(1..ub-merOrderNum))

MerTermIDs ::= SEQUENCE {
  merchantID  MerchantID,
  terminalID  VisibleString (SIZE(1..ub-terminalID)) OPTIONAL,
  agentNum    INTEGER (0..MAX)  OPTIONAL,
  chainNum    [0] INTEGER (0..MAX)  OPTIONAL,
  storeNum    [1] INTEGER (0..MAX)  OPTIONAL
}

RRTags ::= SEQUENCE {
  rrpid        RRPID,
  merTermIDs   MerTermIDs,
  currentDate  Date
}

SaleDetail ::= SEQUENCE {
  batchID                  [ 0] BatchID  OPTIONAL,
  batchSequenceNum         [ 1] BatchSequenceNum OPTIONAL,
  payRecurInd              [ 2] PayRecurInd OPTIONAL,
  merOrderNum              [ 3] MerOrderNum OPTIONAL,
  authCharInd              [ 4] AuthCharInd OPTIONAL,
  commercialCardData       [ 6] CommercialCardData OPTIONAL,
  orderSummary             [ 7] EXPPLICIT SETString { ub-summary }  OPTIONAL,
  customerReferenceNumber  [ 8] EXPPLICIT SETString { ub-reference }  OPTIONAL,
  customerServicePhone     [ 9] EXPPLICIT Phone  OPTIONAL,
  okToPrintPhoneInd        [10] BOOLEAN DEFAULT TRUE,
}

SaleExtensionsIOS EXTENSION ::= { ... }

PayRecurInd ::= ENUMERATED {
  unknown               (0),
  singleTransaction     (1),
  recurringTransaction  (2),
  installmentPayment    (3),
  otherMailOrder        (4)
InstallRecurData ::= SEQUENCE {
    installRecurInd  InstallRecurInd,
    irExtensions     [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL
}

IRExtensionsIOS EXTENSION ::= { ... }

InstallRecurInd ::= CHOICE {
    installTotalTrans  [0] INTEGER (2..MAX),
    recurring          [1] Recurring
}

Recurring ::= SEQUENCE {
    recurringFrequency  INTEGER (1..ub-recurringFrequency),
    recurringExpiry     Date
}

TokenOpaque ::= TYPE-IDENTIFIER.&Type       -- Gateway-defined data

ub-acqCardText         INTEGER ::= 128
ub-acqCardPhone        INTEGER ::=  50
ub-approvalCode        INTEGER ::=   6
ub-AVSData             INTEGER ::= 128
ub-logRefID            INTEGER ::=  32
ub-merOrderNum         INTEGER ::=  25
ub-merType             INTEGER ::=   4
ub-recurringFrequency  INTEGER ::= 366
ub-SettlementAccount   INTEGER ::=  50
ub-summary             INTEGER ::=  35
ub-terminalID          INTEGER ::=  48
ub-validationCode      INTEGER ::=   4

SetCertificate

2000 UnsignedCertificate ::= SEQUENCE {
    2001 version [0] CertificateVersion, 
    2002 serialNumber CertificateSerialNumber, 
    2003 signature AlgorithmIdentifier {{SignatureAlgorithms}}, 
    2004 issuer Name, 
    2005 validity Validity, 
    2006 subject Name, 
    2007 subjectPublicKeyInfo SubjectPublicKeyInfo{{SupportedAlgorithms}}, 
    2008 issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL, 
    2009 subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL, 
}

2011 

2012 CertificateVersion ::= INTEGER { ver3(2) } ( ver3 )

2014 CertificateSerialNumber ::= INTEGER

2016 -- Compute the encrypted hash of this value if issuing a certificate, 
2017 -- or recompute the issuer's signature on this value if validating a 
2018 -- certificate.

2019 

2020 

2021 EncodedCertificate ::= TYPE-IDENTIFIER.&Type (UnsignedCertificate)

2022 Certificate::= SIGNED { 
    2023 EncodedCertificate 
} ( CONstrained BY { -- Verify Or Sign Certificate -- } )

2026 

2027 SIGNED { ToBeSigned } ::= SEQUENCE {
    2028 toBeSigned ToBeSigned, 
    2029 algorithm AlgorithmIdentifier {{SignatureAlgorithms}}, 
    2030 signature BIT STRING 
}

2031 

2032 Validity ::= SEQUENCE {
    2034 notBefore UTCTime, -- Not valid before this date 
    2035 notAfter UTCTime -- Not valid after this date 
}

2036 

2037 

2038 UniqueIdentifier ::= BIT STRING -- Not used in the SET protocol

2039 SubjectPublicKeyInfo {ALGORITHM-IDENTIFIER:Algorithms} ::= SEQUENCE {
    2040 algorithm AlgorithmIdentifier {{Algorithms}},
    2041 subjectPublicKey BIT STRING 
}

2042 

2043 

2044 

2045 

2046 SetCertificateExtensions

2047 { joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 4 } 

2048 DEFINITIONS IMPLICIT TAGS ::= BEGIN 

2049 

2050 

2051 -- Defines X.509 Version 3 certificate extensions.

2052 --
SET Secure Electronic Transaction Specification
May 31, 1997
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110

Book 3: Formal Protocol Definition
Page 215

-- EXPORTS All;
IMPORTS
Name, SETString {}, SupportedAlgorithms
FROM SetAttribute
CertificateSerialNumber, SubjectPublicKeyInfo
FROM SetCertificate
BIN, CountryCode, Language, MerchantID, URL
FROM SetMessage
DD {}, DetachedDigest
FROM SetPKCS7Plus;

-- X.509v3 Certificate Extensions
EXTENSION ::= CLASS {
&id
OBJECT IDENTIFIER UNIQUE,
&critical
BOOLEAN DEFAULT FALSE,
&ExtenType
}
WITH SYNTAX {
SYNTAX
&ExtenType
[ CRITICAL
&critical ]
IDENTIFIED BY &id
}
Extensions ::= SEQUENCE OF Extension
ExtensionSet EXTENSION ::= {
--- Standard X.509v3 extensions
-authorityKeyIdentifier | -- not critical
keyUsage
| -- critical
privateKeyUsagePeriod | -- not critical
certificatePolicies
| -- critical
subjectAltName
| -- not critical
issuerAltName
| -- not critical
basicConstraints
| -- critical
cRLNumber
| -- not critical
--- SET Private extensions
-hashedRootKey
| -- critical
certificateType
| -- critical
merchantData
| -- not critical
cardCertRequired
| -- not critical
tunneling
| -- not critical
setExtensions,
-- not critical
...
}
Extension ::= SEQUENCE {

Version 1.0

-- Information Object Set


2111  extnID   EXTENSION.&id({ExtensionSet}),
2112  critical EXTENSION.&critical({ExtensionSet}{@extnID}) DEFAULT FALSE,
2113  extnValue OCTET STRING -- DER representation of &ExtenType extension
2114                   -- object for the object identified by extnID
2115 )
2116
2117 -- Key and policy information extensions --
2118
2119 authorityKeyIdentifier EXTENSION ::= {
2120    SYNTAX AuthorityKeyIdentifier
2121    IDENTIFIED BY id-ce-authorityKeyIdentifier
2122 }
2123
2124 AuthorityKeyIdentifier ::= SEQUENCE {
2125    keyIdentifier              [0] KeyIdentifier  OPTIONAL,
2126    authorityCertIssuer        [1] GeneralNames  OPTIONAL,
2127    authorityCertSerialNumber  [2] CertificateSerialNumber  OPTIONAL
2128 } ( WITH COMPONENTS { keyIdentifier ABSENT,
2129       authorityCertIssuer PRESENT, authorityCertSerialNumber PRESENT } )
2130
2131 KeyIdentifier ::= OCTET STRING
2132
2133 keyUsage EXTENSION ::= {
2134    SYNTAX KeyUsage
2135    CRITICAL TRUE
2136    IDENTIFIED BY id-ce-keyUsage
2137 }
2138
2139 KeyUsage ::= BIT STRING {
2140    digitalSignature  (0),
2141    nonRepudiation   (1),
2142    keyEncipherment  (2),
2143    dataEncipherment (3),
2144    keyAgreement     (4),
2145    keyCertSign      (5), -- For use in CA-certificates only
2146    cRLSign          (6) -- For use in CA-certificates only
2147 }
2148
2149 privateKeyUsagePeriod EXTENSION ::= {
2150    SYNTAX PrivateKeyUsagePeriod
2151    IDENTIFIED BY id-ce-privateKeyUsagePeriod
2152 }
2153
2154 PrivateKeyUsagePeriod ::= SEQUENCE {
2155    notBefore  [0] GeneralizedTime  OPTIONAL,
2156    notAfter   [1] GeneralizedTime  OPTIONAL
2157 } ( WITH COMPONENTS { ..., notBefore PRESENT } |
2158     WITH COMPONENTS { ..., notAfter  PRESENT } )
2159
2160 certificatePolicies EXTENSION ::= {
2161    SYNTAX CertificatePoliciesSyntax
2162    CRITICAL TRUE
2163    IDENTIFIED BY id-ce-certificatePolicies
2164 }
2165
2166 CertificatePoliciesSyntax ::= SEQUENCE SIZE(1..MAX) OF PolicyInformation
2167
2168 PolicyInformation ::= SEQUENCE {
policyIdentifier  CertPolicyId,
policyQualifiers  SEQUENCE SIZE(1..MAX) OF
                   PolicyQualifierInfo  OPTIONAL
}

CertPolicyId ::= OBJECT IDENTIFIER

PolicyQualifierInfo ::= SEQUENCE {
    policyQualifierId  CERT-POLICY-QUALIFIER.&id
        ((SupportedPolicyQualifiers)),
    qualifier          CERT-POLICY-QUALIFIER.&Qualifier
        ((SupportedPolicyQualifiers){@policyQualifierId})
    OPTIONAL
}

SupportedPolicyQualifiers CERT-POLICY-QUALIFIER ::= {
    setPolicyQualifier,
    ...
}

CERT-POLICY-QUALIFIER ::= CLASS {
    &id         OBJECT IDENTIFIER UNIQUE,
    &Qualifier  OPTIONAL
}

WITH SYNTAX {
    POLICY-QUALIFIER-ID  &id
        [ QUALIFIER-TYPE  &Qualifier ]
}

setPolicyQualifier CERT-POLICY-QUALIFIER ::= {
    POLICY-QUALIFIER-ID  id-set-setQualifier
    QUALIFIER-TYPE       SetPolicyQualifier
}

SetPolicyQualifier ::= SEQUENCE {
    rootQualifier       SETQualifier,
    additionalPolicies  AdditionalPolicies  OPTIONAL
}

AdditionalPolicies ::= SEQUENCE SIZE(1..3) OF AdditionalPolicy

AdditionalPolicy ::= SEQUENCE {
    policyOID        CertPolicyId  OPTIONAL,
    policyQualifier  SETQualifier  OPTIONAL,
    policyAddedBy    CertificateTypeSyntax
}

SETQualifier ::= SEQUENCE {
    policyDigest    DetachedDigest  OPTIONAL,
    terseStatement  SETString {ub-terseStatement}  OPTIONAL,
    policyURL       [0] URL  OPTIONAL,
    policyEmail     [1] URL  OPTIONAL
}

-- Certificate subject and certificate issuer attributes extensions --

subjectAltName EXTENSION ::= {
    SYNTAX         GeneralNames
}
2227       IDENTIFIED BY id-ce-subjectAltName
2228 }
2229
2230 GeneralNames ::= SEQUENCE SIZE(1..MAX) OF GeneralName
2231
2232 GeneralName ::= CHOICE {
2233    directoryName              [4] EXPLICIT Name,
2234    uniformResourceIdentifier  [6] IA5String,
2235    registeredID               [8] OBJECT IDENTIFIER
2236    -- Other choices defined in X.509 not used by SET
2237 }
2238
2239 issuerAltName EXTENSION ::= {
2240    SYNTAX         GeneralNames
2241    IDENTIFIED BY id-ce-issuerAltName
2242 }
2243
2244 -- Certification path constraints extensions --
2245
2246 basicConstraints EXTENSION ::= {
2247    SYNTAX         BasicConstraintsSyntax
2248    CRITICAL       TRUE
2249    IDENTIFIED BY id-ce-basicConstraints
2250 }
2251
2252 BasicConstraintsSyntax ::= SEQUENCE {
2253    cA                 BOOLEAN  DEFAULT FALSE,
2254    pathLenConstraint  INTEGER (0..MAX)  OPTIONAL
2255 }
2256
2257 -- Basic CRL extensions --
2258
2259 cRLNumber EXTENSION ::= {
2260    SYNTAX         CRLNumber
2261    IDENTIFIED BY id-ce-cRLNumber
2262 }
2263
2264 CRLNumber ::= INTEGER (0..MAX)
2265
2266 -- Set protocol private extensions --
2267
2268 hashedRootKey EXTENSION ::= {
2269    SYNTAX         HashedRootKeySyntax
2270    CRITICAL       TRUE
2271    IDENTIFIED BY id-set-hashedRootKey
2272 }
2273
2274 HashedRootKeySyntax ::= RootKeyThumb
2275
2276 RootKeyThumb ::= SEQUENCE {
2277    rootKeyThumbprint  DD { SubjectPublicKeyInfo{{SupportedAlgorithms}} }
2278 }
2279
2280 certificateType EXTENSION ::= {
2281    SYNTAX         CertificateTypeSyntax
2282    CRITICAL       TRUE
2283    IDENTIFIED BY id-set-certificateType
2284 }
CertificateTypeSyntax ::= BIT STRING {
    card (0),
    mer (1),
    pgwy (2),
    cca (3),
    mca (4),
    pca (5),
    gca (6),
    bca (7),
    rca (8),
    acq (9)
}

merchantData EXTENSION ::= {
    SYNTAX MerchantDataSyntax
    IDENTIFIED BY id-set-merchantData
}

MerchantDataSyntax ::= SEQUENCE {
    merID               MerchantID,
    merAcquirerBIN      BIN,
    merNameSeq          MerNameSeq,
    merCountry          CountryCode,
    merAuthFlag         BOOLEAN DEFAULT TRUE
}

MerNameSeq ::= SEQUENCE SIZE(1..32) OF MerNames

MerNames ::= SEQUENCE {
    language       [0] Language OPTIONAL,
    name           [1] EXPLICIT SETString { ub-merName },
    city           [2] EXPLICIT SETString { ub-cityName },
    stateProvince   [3] EXPLICIT SETString { ub-stateProvince } OPTIONAL,
    postalCode     [4] EXPLICIT SETString { ub-postalCode } OPTIONAL,
    countryName    [5] EXPLICIT SETString { ub-countryName }
}

cardCertRequired EXTENSION ::= {
    SYNTAX Boolean
    IDENTIFIED BY id-set-cardCertRequired
}

tunneling EXTENSION ::= {
    SYNTAX TunnelingSyntax
    IDENTIFIED BY id-set-tunneling
}

TunnelingSyntax ::= SEQUENCE {
    tunneling   BOOLEAN DEFAULT TRUE,
    tunnelAlgIDs  TunnelAlg
}

TunnelAlg ::= SEQUENCE OF OBJECT IDENTIFIER

setExtensions EXTENSION ::= {
    SYNTAX SETExtensionsSyntax
    IDENTIFIED BY id-set-setExtensions
}
SETExtensionsSyntax ::= SEQUENCE OF OBJECT IDENTIFIER

-- Upper bounds of SETString{} types

ub-countryName INTEGER ::= 50
ub-cityName INTEGER ::= 50
ub-merName INTEGER ::= 25
ub-postalCode INTEGER ::= 14
ub-stateProvince INTEGER ::= 50
ub-terseStatement INTEGER ::= 2048

-- Object identifiers

id-ce OBJECT IDENTIFIER ::= { 2 5 29 }
id-ce-keyUsage OBJECT IDENTIFIER ::= { id-ce 15 }
id-ce-privateKeyUsagePeriod OBJECT IDENTIFIER ::= { id-ce 16 }
id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
id-ce-issuerAltName OBJECT IDENTIFIER ::= { id-ce 18 }
id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }
id-ce-cRLNumber OBJECT IDENTIFIER ::= { id-ce 20 }
id-ce-certificatePolicies OBJECT IDENTIFIER ::= { id-ce 32 }
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }

id-set OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) internationalRA(23) set(42) }

-- Content type

id-set-contentType OBJECT IDENTIFIER ::= { id-set 0 }
id-set-msgExt OBJECT IDENTIFIER ::= { id-set 1 }
id-set-field OBJECT IDENTIFIER ::= { id-set 2 }
id-set-attribute OBJECT IDENTIFIER ::= { id-set 3 }
id-set-algorithm OBJECT IDENTIFIER ::= { id-set 4 }
id-set-policy OBJECT IDENTIFIER ::= { id-set 5 }
id-set-module OBJECT IDENTIFIER ::= { id-set 6 }
id-set-certificate OBJECT IDENTIFIER ::= { id-set 7 }
id-set-brand OBJECT IDENTIFIER ::= { id-set 8 }
id-set-vendor OBJECT IDENTIFIER ::= { id-set 9 }
id-set-national OBJECT IDENTIFIER ::= { id-set 10 }

-- Content type

id-set-content-PANData OBJECT IDENTIFIER ::= { id-set-contentType 0 }
id-set-content-PANToken OBJECT IDENTIFIER ::= { id-set-contentType 1 }
id-set-content-PANOnly OBJECT IDENTIFIER ::= { id-set-contentType 2 }
id-set-content-OIData OBJECT IDENTIFIER ::= { id-set-contentType 3 }
id-set-content-PI OBJECT IDENTIFIER ::= { id-set-contentType 4 }
id-set-content-PIData OBJECT IDENTIFIER ::= { id-set-contentType 5 }
id-set-content-PIDataUnsigned OBJECT IDENTIFIER ::= { id-set-contentType 6 }
id-set-content-HODInput OBJECT IDENTIFIER ::= { id-set-contentType 7 }
id-set-content-AuthResBaggage OBJECT IDENTIFIER ::= { id-set-contentType 8 }
id-set-content-AuthRevReqBaggage OBJECT IDENTIFIER ::= { id-set-contentType 9 }
id-set-content-AuthRevResBaggage OBJECT IDENTIFIER ::= { id-set-contentType 10 }
id-set-content-CapTokenSeq OBJECT IDENTIFIER ::= { id-set-contentType 11 }
id-set-content-PInitResData OBJECT IDENTIFIER ::= { id-set-contentType 12 }
2401 id-set-content-PI-TBS  
2402 id-set-content-PResData  
2403 id-set-content-InqReqData  
2404 id-set-content-AuthReqTBS  
2405 id-set-content-AuthResTBS  
2406 id-set-content-AuthResTBSX  
2407 id-set-content-AuthTokenTBS  
2408 id-set-content-CapTokenData  
2409 id-set-content-CapTokenTBS  
2410 id-set-content-AcqCardCodeMsg  
2411 id-set-content-AUTHRevReqTBS  
2412 id-set-content-AUTHRevResData  
2413 id-set-content-AUTHRevResTBS  
2414 id-set-content-CapReqTBS  
2415 id-set-content-CapReqTBSX  
2416 id-set-content-CapResData  
2417 id-set-content-CapRevReqTBS  
2418 id-set-content-CapRevReqTBSX  
2419 id-set-content-CapRevResData  
2420 id-set-content-CapRevResTBS  
2421 id-set-content-CredReqTBS  
2422 id-set-content-CredReqTBSX  
2423 id-set-content-CredRevReqTBS  
2424 id-set-content-CredRevReqTBSX  
2425 id-set-content-CredRevResData  
2426 id-set-content-PCertReqData  
2427 id-set-content-PCertResTBS  
2428 id-set-content-BatchAdminReqData  
2429 id-set-content-BatchAdminResData  
2430 id-set-content-CardCInitResTBS  
2431 id-set-content-Me-AcqCInitResTBS  
2432 id-set-content-RegFormResTBS  
2433 id-set-content-CertReqData  
2434 id-set-content-CertReqTBS  
2435 id-set-content-CertResData  
2436 id-set-content-CertInqReqTBS  
2437 id-set-content-ErrorTBS  
2438 id-set-content-PIDualSignedTBE  
2439 id-set-content-PIUnsignedTBE  
2440 id-set-content-AuthReqTBE  
2441 id-set-content-AuthResTBE  
2442 id-set-content-AuthResTBEX  
2443 id-set-content-AuthTokenTBE  
2444 id-set-content-CapTokenTBE  
2445 id-set-content-CapTokenTBEX  
2446 id-set-content-AcqCardCodeMsgTBE  
2447 id-set-content-AUTHRevReqTBE  
2448 id-set-content-AUTHRevResTBE  
2449 id-set-content-AUTHRevResTBEB  
2450 id-set-content-CapReqTBE  
2451 id-set-content-CapReqTBEX  
2452 id-set-content-CapResTBE  
2453 id-set-content-CapRevReqTBE  
2454 id-set-content-CapRevReqTBEX  
2455 id-set-content-CapRevResTBE  
2456 id-set-content-CredReqTBE  
2457 id-set-content-CredReqTBEX  
2458 id-set-content-CredResTBE  

OID ::= { id-set-contentType 13 }  
OID ::= { id-set-contentType 14 }  
OID ::= { id-set-contentType 15 }  
OID ::= { id-set-contentType 16 }  
OID ::= { id-set-contentType 17 }  
OID ::= { id-set-contentType 18 }  
OID ::= { id-set-contentType 19 }  
OID ::= { id-set-contentType 20 }  
OID ::= { id-set-contentType 21 }  
OID ::= { id-set-contentType 22 }  
OID ::= { id-set-contentType 23 }  
OID ::= { id-set-contentType 24 }  
OID ::= { id-set-contentType 25 }  
OID ::= { id-set-contentType 26 }  
OID ::= { id-set-contentType 27 }  
OID ::= { id-set-contentType 28 }  
OID ::= { id-set-contentType 29 }  
OID ::= { id-set-contentType 30 }  
OID ::= { id-set-contentType 31 }  
OID ::= { id-set-contentType 32 }  
OID ::= { id-set-contentType 33 }  
OID ::= { id-set-contentType 34 }  
OID ::= { id-set-contentType 35 }  
OID ::= { id-set-contentType 36 }  
OID ::= { id-set-contentType 37 }  
OID ::= { id-set-contentType 38 }  
OID ::= { id-set-contentType 39 }  
OID ::= { id-set-contentType 40 }  
OID ::= { id-set-contentType 41 }  
OID ::= { id-set-contentType 42 }  
OID ::= { id-set-contentType 43 }  
OID ::= { id-set-contentType 44 }  
OID ::= { id-set-contentType 45 }  
OID ::= { id-set-contentType 46 }  
OID ::= { id-set-contentType 47 }  
OID ::= { id-set-contentType 48 }  
OID ::= { id-set-contentType 49 }  
OID ::= { id-set-contentType 50 }  
OID ::= { id-set-contentType 51 }  
OID ::= { id-set-contentType 52 }  
OID ::= { id-set-contentType 53 }  
OID ::= { id-set-contentType 54 }  
OID ::= { id-set-contentType 55 }  
OID ::= { id-set-contentType 56 }  
OID ::= { id-set-contentType 57 }  
OID ::= { id-set-contentType 58 }  
OID ::= { id-set-contentType 59 }  
OID ::= { id-set-contentType 60 }  
OID ::= { id-set-contentType 61 }  
OID ::= { id-set-contentType 62 }  
OID ::= { id-set-contentType 63 }  
OID ::= { id-set-contentType 64 }  
OID ::= { id-set-contentType 65 }  
OID ::= { id-set-contentType 66 }  
OID ::= { id-set-contentType 67 }  
OID ::= { id-set-contentType 68 }  
OID ::= { id-set-contentType 69 }  
OID ::= { id-set-contentType 70 }
id-set-content-CredRevReqTBE
  OID ::= { id-set-contentType 71 }

id-set-content-CredRevReqTBEX
  OID ::= { id-set-contentType 72 }

id-set-content-BatchAdminReqTBE
  OID ::= { id-set-contentType 74 }

id-set-content-BatchAdminResTBE
  OID ::= { id-set-contentType 75 }

id-set-content-RegFormReqTBE
  OID ::= { id-set-contentType 76 }

id-set-content-CertReqTBE
  OID ::= { id-set-contentType 77 }

id-set-content-CertReqTBEX
  OID ::= { id-set-contentType 78 }

id-set-content-CertResTBE
  OID ::= { id-set-contentType 79 }

id-set-content-CRLNotificationTBS
  OID ::= { id-set-contentType 80 }

id-set-content-CRLNotificationResTBS
  OID ::= { id-set-contentType 81 }

id-set-content-BCIDistributionTBS
  OID ::= { id-set-contentType 82 }

-- Message extensions
None currently defined

-- Fields

id-set-fullName
  OID ::= { id-set-field 0 }

id-set-givenName
  OID ::= { id-set-field 1 }

id-set-familyName
  OID ::= { id-set-field 2 }

id-set-birthFamilyName
  OID ::= { id-set-field 3 }

id-set-placeName
  OID ::= { id-set-field 4 }

id-set-identificationNumber
  OID ::= { id-set-field 5 }

id-set-month
  OID ::= { id-set-field 6 }

id-set-date
  OID ::= { id-set-field 7 }

id-set-address
  OID ::= { id-set-field 8 }

id-set-telephone
  OID ::= { id-set-field 9 }

id-set-amount
  OID ::= { id-set-field 10 }

id-set-accountNumber
  OID ::= { id-set-field 11 }

id-set-passPhrase
  OID ::= { id-set-field 12 }

-- Attributes

id-set-attribute-cert
  OID ::= { id-set-attribute 0 }

id-set-rootKeyThumb
  OID ::= { id-set-attribute-cert 0 }

id-set-additionalPolicy
  OID ::= { id-set-attribute-cert 1 }

-- Algorithms
None currently defined

-- Policy

id-set-policy-root
  OID ::= { id-set-policy 0 }

-- SET private certificate extensions

id-set-hashedRootKey
  OID ::= { id-set-certExt 0 }

id-set-certificateType
  OID ::= { id-set-certExt 1 }

id-set-merchantData
  OID ::= { id-set-certExt 2 }

id-set-cardCertRequired
  OID ::= { id-set-certExt 3 }

id-set-tunneling
  OID ::= { id-set-certExt 4 }

id-set-setExtensions
  OID ::= { id-set-certExt 5 }

id-set-setQualifier
  OID ::= { id-set-certExt 6 }

-- Brands

id-set-IATA-ATA
  OID ::= { id-set-brand 1 }

-- contact: rfcrum@air-travel-card.com

id-set-Diners
  OID ::= { id-set-brand 30 }

-- contact: william.burnett@citicorp.com

id-set-AmericanExpress
  OID ::= { id-set-brand 34 }
id-set-JCB  OID ::= { id-set-brand 35 }  
   -- contact: david.armes@aexp.com
id-set-Visa  OID ::= { id-set-brand 4 }  
   -- contact: tlewis@visa.com
id-set-MasterCard  OID ::= { id-set-brand 5 }  
   -- contact: paul_hollis@mastercard.com
id-set-Novus  OID ::= { id-set-brand 6011 }  
   -- contact: gallman@novus.net.com

-- Vendors
id-set-GlobeSet  OID ::= { id-set-vendor 0 }  
   -- contact: terence@globeset.com
id-set-IBM  OID ::= { id-set-vendor 1 }  
   -- contact: mepeters@raleigh.ibm.com
id-set-Cybercash  OID ::= { id-set-vendor 2 }  
   -- contact: dee@cybercash.com
id-set-Terisa  OID ::= { id-set-vendor 3 }  
   -- contact: briank@terisa.com
id-set-RSADSI  OID ::= { id-set-vendor 4 }  
   -- contact: baldwin@rsa.com
id-set-VeriFone  OID ::= { id-set-vendor 5 }  
   -- contact: trong@vfi.com
id-set-Trintech  OID ::= { id-set-vendor 6 }  
   -- contact: doneill@trintech.com
id-set-BankGate  OID ::= { id-set-vendor 7 }  
   -- contact: johnv@bankgate.com
id-set-GTE  OID ::= { id-set-vendor 8 }  
   -- contact: jeanegorman@gsc.gte.com
id-set-CompuSource  OID ::= { id-set-vendor 9 }  
   -- contact: simonr@compusource.co.za
id-set-Griffin  OID ::= { id-set-vendor 10 }  
   -- contact: asnl@mindspring.com
id-set-Certicom  OID ::= { id-set-vendor 11 }  
   -- contact: sshannon@certicom.ca
id-set-OSS  OID ::= { id-set-vendor 12 }  
   -- contact: baoas@oss.com
id-set-TenthMountain  OID ::= { id-set-vendor 13 }  
   -- contact: dapkus@tenthmountain.com
id-set-Antares  OID ::= { id-set-vendor 14 }  
   -- contact: bzcd0@toraag.com
id-set-ECC  OID ::= { id-set-vendor 15 }  
   -- contact: beattie@ecconsultants.com
id-set-Maithean  OID ::= { id-set-vendor 16 }  
   -- contact: sullivan@maithean.com
id-set-Netscape  OID ::= { id-set-vendor 17 }  
   -- contact: rich@netscape.com
id-set-VeriSign  OID ::= { id-set-vendor 18 }  
   -- contact: simpson@verisign.com
id-set-BlueMoney  OID ::= { id-set-vendor 19 }  
   -- contact: jeremy@bluemoney.com
id-set-Lacerte  OID ::= { id-set-vendor 20 }  
   -- contact: lacerte@lacerte.com
id-set-Fujitsu  OID ::= { id-set-vendor 21 }  
   -- contact: sfuruta@inet.mmp.fujitsu.co.jp
id-set-eLab  OID ::= { id-set-vendor 22 }  
   -- contact: rah@shipwright.com
id-set-Entrust  OID ::= { id-set-vendor 23 }  
   -- contact: davidarmes@aexp.com

Version 1.0
SetCRL

{ joint-iso-itut(2) internationalra(23) set(42) module(6) 5 }
DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- This module defines types for Certificate Revocation List support.

EXPORTS All;

IMPORTS

AlgorithmIdentifier(), Name, SignatureAlgorithms
FROM SetAttribute

CertificateSerialNumber, SIGNED {} FROM SetCertificate
Extensions
FROM SetCertificateExtensions;

UnsignedCertificateRevocationList ::= SEQUENCE {
    version          INTEGER { crlVer2(1) } ( crlVer2 ),
    signature        AlgorithmIdentifier {{SignatureAlgorithms}},
    issuer           Name,
    thisUpdate       UTCTime,
    nextUpdate       UTCTime,
    revokedCertificates CRLEntryList OPTIONAL,
    crlExtensions    [0] Extensions OPTIONAL
}

CRLEntryList ::= SEQUENCE OF CRLEntry

CRLEntry ::= SEQUENCE{
    userCertificate  CertificateSerialNumber,
    revocationDate   UTCTime,
    crlEntryExtensions Extensions OPTIONAL
}

EncodedCRL ::= TYPE-IDENTIFIER.&Type (UnsignedCertificateRevocationList)

CRL ::= SIGNED {
    EncodedCRL
} (CONSTRAINED BY { -- Validate Or Issue CRL -- })

END

SetPKCS7Plus
-- joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 6
DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- This module defines types for manipulating RSA PKCS #7 Cryptographic
-- Messages, as well as SET-specific messages which contain these types.
-- Note that SET uses definitions for PKCS-7 version 1.6.

IMPORTS

ALGORITHM-IDENTIFIER, AlgorithmIdentifier {}, ATTRIBUTE,
Attribute {}, Name
FROM SetAttribute

Certificate, CertificateSerialNumber
FROM SetCertificate

CRL
FROM SetCRL

CardExpiry, PAN
FROM SetMessage;

CRLSequence ::= SEQUENCE OF CRL

IssuerAndSerialNumber ::= SEQUENCE { -- Uniquely identifies certificate
  issuer        Name,
  serialNumber  CertificateSerialNumber
}

CONTENTS ::= TYPE-IDENTIFIER

Contents CONTENTS ::= {
  SignedData IDENTIFIED BY signedData },

ContentInfo ::= SEQUENCE {
  contentType  ContentType,
  content      [0] EXPLICIT CONTENTS.&Type({Contents}
  (@contentType)) OPTIONAL
}

ContentType ::= CONTENTS.&id({Contents})

SignedData ::= SEQUENCE {                                      -- PKCS#7
  sdVersion         INTEGER { sdVer2(2) } (sdVer2),
  digestAlgorithms  DigestAlgorithmIdentifiers,
  contentInfo       ContentInfo,
  certificates      [2] IMPLICIT Certificates  OPTIONAL,
  crls              [3] IMPLICIT CRLSequence  OPTIONAL,
  signerInfos       SignerInfos
}

SignerInfos ::= SEQUENCE OF SignerInfo

Authenticated ATTRIBUTE ::={
  { WITH SYNTAX ContentType ID content Type } |
  { WITH SYNTAX MessageDigest ID messageDigest } ,
  ...
}

MessageDigest ::= Digest

Digests ::= SEQUENCE OF Digest
Digest ::= OCTET STRING (SIZE(1..20))

Certificates ::= SEQUENCE OF Certificate

DigestAlgorithmIdentifiers ::= SEQUENCE OF AlgorithmIdentifier { {DigestAlgorithms} }

DigestAlgorithms ALGORITHM-IDENTIFIER ::= {
    { NULL IDENTIFIED BY id-sha1 }, ...
}

DigestEncryptionAlgorithms ALGORITHM-IDENTIFIER ::= {
    { NULL IDENTIFIED BY id-rsaEncryption }, ...
}

EncryptedData ::= SEQUENCE {
    version          INTEGER { enVer0(0) } (enVer0),
    encryptedContentInfo  EncryptedContentInfo
}

EnvelopedData ::= SEQUENCE {
    edVersion         INTEGER { edVer1(1) } (edVer1),
    recipientInfos    RecipientInfos,
    encryptedContentInfo  EncryptedContentInfo
}

RecipientInfos ::= SEQUENCE OF RecipientInfo

EncryptedContentInfo ::= SEQUENCE {
    contentType       ContentType,
    contentEncryptionAlgorithm
                        AlgorithmIdentifier {{ContentEncryptionAlgorithms}},
    encryptedContent  [0] IMPLICIT EncryptedContent  OPTIONAL
}

EncryptedContent ::= OCTET STRING

ContentEncryptionAlgorithms ALGORITHM-IDENTIFIER ::= {
    { CBC8Parameter IDENTIFIED BY id-desCMDF } |
    { CBC8Parameter IDENTIFIED BY id-desCBC }, ...
}

CBC8Parameter ::= OCTET STRING (SIZE(8))

RecipientInfo ::= SEQUENCE {
    riVersion          INTEGER { riVer0(0) } (riVer0),
    issuerAndSerialNumber  IssuerAndSerialNumber,
    keyEncryptionAlgorithm AlgorithmIdentifier {{KeyEncryptionAlgorithms}},
    encryptedKey       EncryptedKey
}

KeyEncryptionAlgorithms ALGORITHM-IDENTIFIER ::= {
    { NULL IDENTIFIED BY rsaOAEPEncryptionSET }, ...
}
When using the algorithm rsaOAEPEncryptionSET, the OAEP block is encrypted using the recipient’s public key and the result carried in EncryptedKey.

EncryptedKey ::= OCTET STRING (SIZE(1..128))

DigestedData ::= SEQUENCE {
  ddVersion        INTEGER { ddVer0(0) } (ddVer0),
  digestAlgorithm  AlgorithmIdentifier {{DigestAlgorithms}},
  contentInfo      ContentInfo,
  digest           Digest
}

EncryptedDigest ::= OCTET STRING

AttributeSeq { ATTRIBUTE:InfoObjectSet } ::= 
  SEQUENCE OF Attribute { {InfoObjectSet} }

Cryptographic Parameterized Types --

L { T1, T2 } ::= SEQUENCE {                     -- Linkage from t1 to t2
  t1  T1,
  t2  DD { T2 }                                  -- PKCS#7 DigestedData
}

DD { ToBeHashed } ::= DetachedDigest
  (CONSTRAINED BY { -- digest of the DER representation, including --
    -- the tag and length octets, of -- ToBeHashed })

DetachedDigest ::= DigestedData                          -- No parameter
  (WITH COMPONENTS {..., contentInfo (WITH COMPONENTS
                   {..., content ABSENT})})

H { ToBeHashed } ::= OCTET STRING (SIZE(1..20)) (CONSTRAINED BY {
  -- HASH is an n-byte value, which is the results --
  -- of the application of a valid digest procedure --
  -- applied to -- ToBeHashed })

HMAC { ToBeHashed, Key } ::= Digest
  (CONSTRAINED BY { -- HMAC keyed digest of -- ToBeHashed, -- using -- Key })

HMACPanData ::= SEQUENCE {           -- For HMAC, unique cardholder data
  pan         PAN,
  cardExpiry  CardExpiry
}

S { SIGNER, ToBeSigned } ::= SignedData
  (CONSTRAINED BY { SIGNER, -- signs -- ToBeSigned })

S { SIGNER, ToBeSigned } ::= SignedData
  (CONSTRAINED BY { SIGNER, -- signs -- ToBeSigned })

SO { SIGNER, ToBeSigned } ::= SignedData -- Detached content
  (CONSTRAINED BY { SIGNER, -- signs -- ToBeSigned })

(WITH COMPONENTS { ..., contentInfo
2859     (WITH COMPONENTS{
2860        ..., content ABSENT }) ) ^
2861     WITH COMPONENTS { ..., signerInfos (SIZE(1..2)) })
2862
2863
2864 -- Set Encapsulation Types
2865
2866
2867 -- Simple Encapsulation with Signature --
2868
2869 Enc { SIGNER, RECIPIENT, T } ::= E {
2870     RECIPIENT,
2871     S { SIGNER, T }
2872 }
2873
2874
2875 -- Simple Encapsulation with Signature and a Provided Key --
2876
2877 EncK { KeyData, SIGNER, T } ::= EK {
2878     KeyData,
2879     S { SIGNER, T }
2880 }
2881
2882
2883 -- Extra Encapsulation with Signature --
2884
2885 EncX { SIGNER, RECIPIENT, T, Parameter } ::= E {
2886     RECIPIENT,
2887     SEQUENCE {
2888        t  T,
2889        s  SO { SIGNER, SEQUENCE { t  T, p  Parameter } }
2890    }
2891 } (CONSTRAINED BY { Parameter -- data, which shall contain a fresh --
2892                    -- nonce 'n', is included in the OAEP block. -- })
2893
2894
2895 -- Simple Encapsulation with Signature and Baggage --
2896
2897 EncB { SIGNER, RECIPIENT, T, Baggage } ::= SEQUENCE {
2898     enc  Enc { SIGNER, RECIPIENT, L { T, Baggage } },
2899     baggage  Baggage
2900 }
2901
2902
2903 -- Extra Encapsulation with Signature and Baggage --
2904
2905 EncBX { SIGNER, RECIPIENT, T, Baggage, Parameter } ::= SEQUENCE {
2906     encX  EncX { SIGNER, RECIPIENT, L { T, Baggage }, Parameter },
2907     baggage  Baggage
2908 }
2909
2910
2911 -- Other Cryptographic Messages --
2912
2913 E { RECIPIENT, ToBeEnveloped } ::= EnvelopedData
2914 (CONSTRAINED BY { ToBeEnveloped, -- is encrypted, and the --
2915                    -- session key is encrypted using the --
2916                    -- public key of -- RECIPIENT })
(WITH COMPONENTS { ..., encryptedContentInfo
  (WITH COMPONENTS { ..., encryptedContent PRESENT }) } ^
  WITH COMPONENTS { ..., recipientInfos (SIZE(1)) })

2920

2921 EH { RECIPIENT, ToBeEnveloped } ::= E {
2922    RECIPIENT,
2923    ToBeEnveloped
2924 } (CONSTRAINED BY { -- H(ToBeEnveloped) included in the OAEP block -- })
2925
2926 EX { RECIPIENT, ToBeEnveloped, Parameter } ::= E {
2927    RECIPIENT,
2928    L { ToBeEnveloped, Parameter }
2929 } (CONSTRAINED BY { Parameter -- data is included in the OAEP block -- })
2930
2931 EXH { RECIPIENT, ToBeEnveloped, Parameter } ::= EX {
2932    RECIPIENT,
2933    ToBeEnveloped,
2934    Parameter
2935 } (CONSTRAINED BY { -- H(ToBeEnveloped) included in the OAEP block -- })
2936
2937 EK { KeyData, ToBeEnveloped } ::= EncryptedData
2938    (CONSTRAINED BY { ToBeEnveloped, -- encrypted with -- KeyData })
2939    (WITH COMPONENTS { ..., encryptedContentInfo
2940       (WITH COMPONENTS { ..., encryptedContent PRESENT}) })
2941
2942 ENTITY-IDENTIFIER ::= TYPE-IDENTIFIER             -- Generic placeholder
2943
2944 C ::= ENTITY-IDENTIFIER  -- Cardholder
2945 M ::= ENTITY-IDENTIFIER  -- Merchant
2946 P ::= ENTITY-IDENTIFIER  -- Payment Gateway
2947 EE ::= ENTITY-IDENTIFIER  -- End Entity
2948 CA ::= ENTITY-IDENTIFIER  -- Certifying Authority
2949 P1 ::= ENTITY-IDENTIFIER  -- Gateway One
2950 P2 ::= ENTITY-IDENTIFIER  -- Gateway Two
2951
2952 -- Object Identifiers --
2953
2954 secsig OBJECT IDENTIFIER ::= {
2955   iso(1) identified-organization(3) oiw(14) secsig(3) }
2956
2957 pkcs-1 OBJECT IDENTIFIER ::= {
2958   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }
2959
2960 rsaOAEPEncryptionSET OBJECT IDENTIFIER ::= { pkcs-1 6 }
2961
2962 id-rsaEncryption OBJECT IDENTIFIER ::= { pkcs-1 1 }
2963
2964 id-shal-with-rsa-signature OBJECT IDENTIFIER ::= { pkcs-1 5 }
2965
2966 id-shal OBJECT IDENTIFIER ::= { secsig 2 26 }
2967
2968 id-desCBC OBJECT IDENTIFIER ::= { secsig 2 7 }
2969
2970 id-desCDMF OBJECT IDENTIFIER ::= {
2971   iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) 10}
2972
2973 pkcs-7 OBJECT IDENTIFIER ::= {
2974   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 7 }

Version 1.0
data OBJECT IDENTIFIER ::= { pkcs-7 1 }
signedData OBJECT IDENTIFIER ::= { pkcs-7 2 }
envelopedData OBJECT IDENTIFIER ::= { pkcs-7 3 }
digestedData OBJECT IDENTIFIER ::= { pkcs-7 5 }
pkcs-9 OBJECT IDENTIFIER ::= {
  iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 9 }
contentType OBJECT IDENTIFIER ::= { pkcs-9 3 }
messageDigest OBJECT IDENTIFIER ::= { pkcs-9 4 }
END

SetAttribute
{ joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 7 }
DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- This module defines types from ISO/IEC 9594-2:1995(E), Annex B, known
-- as the Information Framework. A minimal number of types have been
-- copied in order to constrain certificate names in SET. Specific SET
-- implementations may wish to copy additional X.501 types as necessary
-- to facilitate directory manipulation. National language support is
-- achieved through the DirectoryString type, copied from the X-500
-- series SelectedAttributeTypes module, and restricted for use in SET.

attributes

commonName ATTRIBUTE ::= {
  WITH SYNTAX DirectoryString { ub-common-name }
  ID id-at-commonName
}
countryName ATTRIBUTE ::= {
  WITH SYNTAX PrintableString( SIZE(2) )
  ID id-at-countryName
}
organizationName ATTRIBUTE ::= {
  WITH SYNTAX DirectoryString { ub-organization-name }
  ID id-at-organizationName
}
organizationalUnitName ATTRIBUTE ::= {
  WITH SYNTAX DirectoryString { ub-organizational-unit-name }
  ID id-at-organizationalUnitName
}
Attribute { ATTRIBUTE:InfoObjectSet } ::= SEQUENCE {
    type    ATTRIBUTE.&id({InfoObjectSet}),
    values  SET SIZE(1) OF ATTRIBUTE.&Type({InfoObjectSet}@type)
}

AttributeTypeAndValue ::= SEQUENCE {
    type   ATTRIBUTE.&id((SupportedAttributes)),
    value  ATTRIBUTE.&Type((SupportedAttributes}@type)
}

SupportedAttributes ATTRIBUTE ::= {
    countryName             |
    organizationName        |
    organizationalUnitName  |
    commonName
}

ALGORITHM-IDENTIFIER ::= TYPE-IDENTIFIER

AlgorithmIdentifier { ALGORITHM-IDENTIFIER:InfoObjectSet } ::= SEQUENCE {
    algorithm  ALGORITHM-IDENTIFIER.&id({InfoObjectSet}),
    parameters  ALGORITHM-IDENTIFIER.&Type({InfoObjectSet}@algorithm) OPTIONAL
}

SupportedAlgorithms ALGORITHM-IDENTIFIER ::= {
    ...,
    KeyEncryptionAlgorithms |
    SignatureAlgorithms
}

SignatureAlgorithms ALGORITHM-IDENTIFIER ::= {
    sha1-with-rsa-signature,
    ...
}

sha1-with-rsa-signature ALGORITHM-IDENTIFIER ::= {
    NULL IDENTIFIED BY id-sha1-with-rsa-signature }

-- naming data types

Name ::= CHOICE {
    -- only one possibility for now --
    distinguishedName RDNSequence }

RDNSequence ::= SEQUENCE SIZE (1..5) OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET SIZE(1) OF AttributeTypeAndValue

ATTRIBUTE ::= CLASS {
    &derivation         ATTRIBUTE OPTIONAL,
    &Type               OPTIONAL,  -- &Type or &derivation required
    &equality-match     MATCHING-RULE OPTIONAL,
    &ordering-match     MATCHING-RULE OPTIONAL,
    &substrings-match   MATCHING-RULE OPTIONAL,
\$single-valued\$ BOOLEAN DEFAULT FALSE,
\$collective\$ BOOLEAN DEFAULT FALSE,
-- operational extensions
\$no-user-modification\$ BOOLEAN DEFAULT FALSE,
\$usage\$ AttributeUsage DEFAULT userApplications,
\$id\$ OBJECT IDENTIFIER UNIQUE
}
WITH SYNTAX {
-- [ SUBTYPE OF                \$derivation \]           --
-- [ -- WITH SYNTAX            \$Type -- ] --
-- [ EQUALITY MATCHING RULE    \$equality-match \]       --
-- [ ORDERING MATCHING RULE    \$ordering-match \]       --
-- [ SUBSTRINGS MATCHING RULE  \$substrings-match \]     --
-- [ SINGLE VALUE              \$single-valued \]        --
-- [ COLLECTIVE                \$collective \]           --
-- [ NO USER MODIFICATION      \$no-user-modification \] --
ID \$id
}

AttributeUsage ::= ENUMERATED {
  userApplications    (0),
directoryOperation   (1),
distributedOperation (2),
dSAOperation         (3)
}

MATCHING-RULE ::= CLASS {
  \$AssertionType\$ OPTIONAL,
  \$id\$ OBJECT IDENTIFIER UNIQUE
}
WITH SYNTAX {
  \$AssertionType\$
  ID \$id
}

DirectoryString { INTEGER:maxSIZE } ::= CHOICE {
  printableString  PrintableString (SIZE(1..maxSIZE)),
bmpString        BMPString (SIZE(1..maxSIZE))
}

SETString { INTEGER:maxSIZE } ::= CHOICE {
  visibleString  VisibleString (SIZE(1..maxSIZE)),
bmpString      BMPString (SIZE(1..maxSIZE))
}

-- Upper bounds of type Name components

ub-common-name       INTEGER ::=  64
ub-organization-name INTEGER ::=  64
ub-organizational-unit-name INTEGER ::=  64

ds OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) ds(5) }

id-at OBJECT IDENTIFIER ::= { ds 4 }
id-at-commonName OBJECT IDENTIFIER ::= { id-at 3 }
id-at-countryName OBJECT IDENTIFIER ::= { id-at 6 }
3146 id-at-organizationName OCTET STRING ::= { id-at 10 }
3147 id-at-organizationalUnitName OCTET STRING ::= { id-at 11 }
3148
3149 END

3150 SetMarketData
3151   { joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 8 }
3152       DEFINITIONS IMPLICIT TAGS ::= BEGIN
3153
3154 -- EXPORTS All;
3155
3156 IMPORTS
3157
3158    Date, DateTime, Distance, Location, Phone
3159       FROM SetMessage
3160
3161    CurrencyAmount, FloatingPoint, ub-merType
3162       FROM SetPayMsgs
3163
3164    SETString
3165       FROM SetAttribute;
3166
3167 CommercialCardData ::= SEQUENCE {
3168    chargeInfo [0] ChargeInfo OPTIONAL,
3169    merchantLocation [1] Location OPTIONAL,
3170    shipFrom [2] Location OPTIONAL,
3171    shipTo [3] Location OPTIONAL,
3172    itemSeq [4] ItemSeq OPTIONAL
3173 }
3174
3175 ChargeInfo ::= SEQUENCE {
3176    totalFreightShippingAmount [ 0] CurrencyAmount OPTIONAL,
3177    totalDutyTariffAmount [ 1] CurrencyAmount OPTIONAL,
3178    dutyTariffReference [ 2] EXPLICIT SETString { ub-reference } OPTIONAL,
3179    totalNationalTaxAmount [ 3] CurrencyAmount OPTIONAL,
3180    totalLocalTaxAmount [ 4] CurrencyAmount OPTIONAL,
3181    totalOtherTaxAmount [ 5] CurrencyAmount OPTIONAL,
3182    totalTaxAmount [ 6] CurrencyAmount OPTIONAL,
3183    merchantTaxID [ 7] EXPLICIT SETString { ub-taxID } OPTIONAL,
3184    merchantDutyTariffRef [ 8] EXPLICIT SETString { ub-reference } OPTIONAL,
3185    customerDutyTariffRef [ 9] EXPLICIT SETString { ub-reference } OPTIONAL,
3186    summaryCommodityCode [10] EXPLICIT SETString { ub-commCode } OPTIONAL,
3187    merchantType [11] EXPLICIT SETString { ub-merType } OPTIONAL
3188 }
3189
3190 ItemSeq ::= SEQUENCE SIZE(1..ub-items) OF Item
3191
3192 Item ::= SEQUENCE {
3193    quantity INTEGER (1..MAX) DEFAULT 1,
3194    unitOfMeasureCode [0] EXPLICIT SETString { ub-unitMeasure } OPTIONAL,
3195    descriptor SETString { ub-description },
3196    commodityCode [1] EXPLICIT SETString { ub-commCode } OPTIONAL,
unitCost  [3] CurrencyAmount  OPTIONAL,
netCost    [4] CurrencyAmount  OPTIONAL,
discountInd BOOLEAN DEFAULT FALSE,
discountAmount [5] CurrencyAmount  OPTIONAL,
nationalTaxAmount [6] CurrencyAmount  OPTIONAL,
nationalTaxRate  [7] FloatingPoint  OPTIONAL,
nationalTaxType  [8] EXPLICIT SETString { ub-taxType }  OPTIONAL,
localTaxAmount  [9] CurrencyAmount  OPTIONAL,
otherTaxAmount [10] CurrencyAmount  OPTIONAL,
itemTotalCost  CurrencyAmount
}

MarketAutoCap ::= SEQUENCE {
renterName  [0] EXPLICIT SETString { ub-renterName }  OPTIONAL,
rentalLocation [1] Location  OPTIONAL,
rentalDateTime DateTime,
autoNoShow  [2] AutoNoShow  OPTIONAL,
rentalAgreementNumber [3] EXPLICIT SETString { ub-rentalNum }  OPTIONAL,
referenceNumber [4] EXPLICIT SETString { ub-rentalRefNum }  OPTIONAL,
insuranceType  [5] EXPLICIT SETString { ub-insuranceType }  OPTIONAL,
autoRateInfo  [6] AutoRateInfo  OPTIONAL,
returnLocation [7] Location  OPTIONAL,
returnDateTime DateTime,
autoCharges  AutoCharges
}

AutoNoShow ::= ENUMERATED {
normalVehicle  (0),
specialVehicle  (1)
}

AutoRateInfo ::= SEQUENCE {
autoApplicableRate  AutoApplicableRate,
lateReturnHourlyRate [0] CurrencyAmount  OPTIONAL,
distanceRate  [1] CurrencyAmount  OPTIONAL,
freeDistance  [2] Distance  OPTIONAL,
vehicleClassCode  [3] EXPLICIT SETString { ub-vehicleClass }  OPTIONAL,
corporateID  [4] EXPLICIT SETString { ub-corpID }  OPTIONAL
}

AutoApplicableRate ::= CHOICE {
dailyRentalRate  [0] CurrencyAmount,
weeklyRentalRate  [1] CurrencyAmount
}

AutoCharges ::= SEQUENCE {
regularDistanceCharges  CurrencyAmount,
lateReturnCharges  [0] CurrencyAmount  OPTIONAL,
totalDistance  [1] Distance  OPTIONAL,
extraDistanceCharges  [2] CurrencyAmount  OPTIONAL,
insuranceCharges  [3] CurrencyAmount  OPTIONAL,
fuelCharges  [4] CurrencyAmount  OPTIONAL,
autoTowingCharges  [5] CurrencyAmount  OPTIONAL,
oneWayDropOffCharges  [6] CurrencyAmount  OPTIONAL,
telephoneCharges  [7] CurrencyAmount  OPTIONAL,
violationsCharges  [8] CurrencyAmount  OPTIONAL,
deliveryCharges  [9] CurrencyAmount  OPTIONAL,
parkingCharges  [10] CurrencyAmount  OPTIONAL,
otherCharges          [11] CurrencyAmount  OPTIONAL,
totalTaxAmount       [12] CurrencyAmount  OPTIONAL,
auditAdjustment      [13] CurrencyAmount  OPTIONAL
}

MarketHotelCap ::= SEQUENCE {
  arrivalDate       Date,
  hotelNoShow       [0] HotelNoShow  OPTIONAL,
  departureDate     Date,
  durationOfStay    [1] INTEGER (0..99)  OPTIONAL,
  folioNumber       [2] EXPLICIT SETString { ub-hotelFolio }  OPTIONAL,
  propertyPhone     [3] Phone  OPTIONAL,
  customerServicePhone [4] Phone  OPTIONAL,
  programCode       [5] EXPLICIT SETString { ub-programCode }  OPTIONAL,
  hotelRateInfo     [6] HotelRateInfo  OPTIONAL,
  hotelCharges      HotelCharges
}

HotelNoShow ::= ENUMERATED {
  guaranteedLateArrival  (0)
}

HotelRateInfo ::= SEQUENCE {
  dailyRoomRate  CurrencyAmount,
  dailyTaxRate   CurrencyAmount  OPTIONAL
}

HotelCharges ::= SEQUENCE {
  roomCharges            CurrencyAmount,
  roomTax                [ 0] CurrencyAmount  OPTIONAL,
  prepaidExpenses        [ 1] CurrencyAmount  OPTIONAL,
  foodBeverageCharges    [ 2] CurrencyAmount  OPTIONAL,
  roomServiceCharges     [ 3] CurrencyAmount  OPTIONAL,
  miniBarCharges         [ 4] CurrencyAmount  OPTIONAL,
  laundryCharges         [ 5] CurrencyAmount  OPTIONAL,
  telephoneCharges       [ 6] CurrencyAmount  OPTIONAL,
  businessCenterCharges  [ 7] CurrencyAmount  OPTIONAL,
  parkingCharges         [ 8] CurrencyAmount  OPTIONAL,
  movieCharges           [ 9] CurrencyAmount  OPTIONAL,
  healthClubCharges      [10] CurrencyAmount  OPTIONAL,
  giftShopPurchases      [11] CurrencyAmount  OPTIONAL,
  folioCashAdvances      [12] CurrencyAmount  OPTIONAL,
  otherCharges           [13] CurrencyAmount  OPTIONAL,
  totalTaxAmount         [14] CurrencyAmount  OPTIONAL,
  auditAdjustment        [15] CurrencyAmount  OPTIONAL
}

MarketTransportCap ::= SEQUENCE {
  passengerName     SETString { ub-passName },
  departureDate     Date,
  origCityAirport   SETString { ub-airportCode },
  tripLegSeq       [0] TripLegSeq  OPTIONAL,
  ticketNumber     [1] EXPLICIT SETString { ub-ticketNum }  OPTIONAL,
  travelAgencyCode [2] EXPLICIT SETString { ub-taCode }  OPTIONAL,
  travelAgencyName [3] EXPLICIT SETString { ub-taName }  OPTIONAL,
  restrictions     [4] Restrictions  OPTIONAL
}

Version 1.0
TripLegSeq ::= SEQUENCE SIZE(1..16) OF TripLeg

TripLeg ::= SEQUENCE {
    dateOfTravel     Date,
    carrierCode      SETString { ub-carrierCode },
    serviceClass     SETString { ub-serviceClass },
    stopOverCode     StopOverCode,
    destCityAirport  SETString { ub-airportCode },
    fareBasisCode    [0] SETString { ub-fareBasis }  OPTIONAL,
    departureTax     [1] CurrencyAmount  OPTIONAL
}

StopOverCode ::= ENUMERATED {
    noStopOverPermitted  (0),
    stopOverPermitted    (1)
}

Restrictions ::= ENUMERATED {
    unspecifiedRestriction  (0)
}

ub-airportCode    INTEGER ::=   3
ub-carrierCode    INTEGER ::=   2
ub-commCode       INTEGER ::=  15
ub-corpID         INTEGER ::=  12
ub-description    INTEGER ::=  35
ub-fareBasis      INTEGER ::=   6
ub-hotelFolio     INTEGER ::=  25
ub-insuranceType  INTEGER ::=   1
ub-items          INTEGER ::= 999
ub-passName       INTEGER ::=  20
ub-phone          INTEGER ::=  20
ub-productCode    INTEGER ::=  12
ub-programCode    INTEGER ::=   2
ub-reference      INTEGER ::=  28
ub-rentalNum      INTEGER ::=  25
ub-rentalRefNum   INTEGER ::=   8
ub-renterName     INTEGER ::=  40
ub-serviceClass   INTEGER ::=   1
ub-taCode         INTEGER ::=   8
ub-taName         INTEGER ::=  25
ub-taxID          INTEGER ::=  10
ub-taxType        INTEGER ::=   4
ub-ticketNum      INTEGER ::=  13
ub-vehicleClass   INTEGER ::=   2
ub-unitMeasure    INTEGER ::=  12

SetPKCS10

{ joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 9 }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

IMPORTS
Attribute {}, ATTRIBUTE, Name, SupportedAlgorithms
    FROM SetAttribute

SIGNED {}, SubjectPublicKeyInfo {}
    FROM SetCertificate

AdditionalPolicy, CertificateTypeSyntax, GeneralNames, id-ce-keyUsage,
id-ce-privateKeyUsagePeriod, id-ce-subjectAltName,
id-set-additionalPolicy, id-set-certificateType, id-set-tunneling,
KeyUsage, PrivateKeyUsagePeriod, TunnelingSyntax
    FROM SetCertificateExtensions;

AttributeSet { ATTRIBUTE:InfoObjectSet } ::= 
    SET OF Attribute { {InfoObjectSet} }

EncodedCertificationRequestInfo ::= 
    TYPE-IDENTIFIER.&Type (CertificationRequestInfo)

CertificationRequest ::= SIGNED {
    EncodedCertificationRequestInfo
} { CONSTRAINED BY { -- Verify Or Sign CertificationRequest -- } }

CertificationRequestInfo ::= SEQUENCE {
    version               INTEGER { criVer1(0) } (criVer1),
    subject               Name,
    subjectPublicKeyInfo  SubjectPublicKeyInfo {{SupportedAlgorithms}},
    attributes            [0] IMPLICIT AttributeSet {{SupportedCRIAttributes}}
}

SupportedCRIAttributes ATTRIBUTE ::= {
    -- Attributes corresponding to standard X.509v3 extensions
    -- Attributes corresponding to SET private extensions
    -- Attributes corresponding to certificate policy
}

END
## Cross reference

<table>
<thead>
<tr>
<th>AcctData</th>
<th>397 394</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcctIdentification</td>
<td>402 398</td>
</tr>
<tr>
<td>AcctInfo</td>
<td>392 575 589</td>
</tr>
<tr>
<td>AcqBackKey</td>
<td>1102 1104</td>
</tr>
<tr>
<td>AcqCardCode</td>
<td>1114 1110</td>
</tr>
<tr>
<td>AcqCardCodeMsg</td>
<td>1109 1104 1107</td>
</tr>
<tr>
<td>AcqCardCodeMsgTBE</td>
<td>1107</td>
</tr>
<tr>
<td>AcqCardMsg</td>
<td>1104 934 1098</td>
</tr>
<tr>
<td>AcqCardMsgData</td>
<td>1120 1111</td>
</tr>
<tr>
<td>AcquirerBusinessID</td>
<td>419 416</td>
</tr>
<tr>
<td>AcquirerID</td>
<td>414 406</td>
</tr>
<tr>
<td>AdditionalPolicies</td>
<td>2208 2205</td>
</tr>
<tr>
<td>AdditionalPolicy</td>
<td>2210 2208 3414</td>
</tr>
<tr>
<td>ALGORITHM-IDENTIFIER</td>
<td>3051 244 2751 2756 2783 2798 3054 3055 3059 3065 3070</td>
</tr>
<tr>
<td>AlgorithmIdentifier</td>
<td>3053 255 331 2003 2029 2041 2636 2726 2729 2749 2777 2794 2809</td>
</tr>
<tr>
<td>AmountType</td>
<td>1765 1686 1758</td>
</tr>
<tr>
<td>ApprovalCode</td>
<td>1215 1171</td>
</tr>
<tr>
<td>ARqExtensionsIOS</td>
<td>1028 1025</td>
</tr>
<tr>
<td>ARsExtensionsIOS</td>
<td>1132 1129</td>
</tr>
<tr>
<td>ARvRqExtensionsIOS</td>
<td>1245 1242</td>
</tr>
<tr>
<td>ARvRsExtensionsIOS</td>
<td>1288 1285</td>
</tr>
<tr>
<td>ATTRIBUTE</td>
<td>3082 2734 3012 3017 3022 3027 3035 3036 3040 3041 3044 3083 3399</td>
</tr>
<tr>
<td>Attribute</td>
<td>3034 2817 3383</td>
</tr>
<tr>
<td>AttributeSeq</td>
<td>2816 2728 2731</td>
</tr>
<tr>
<td>AttributeSet</td>
<td>3382 3396</td>
</tr>
<tr>
<td>AttributeTypeAndValue</td>
<td>3039 3080</td>
</tr>
<tr>
<td>AttributeUsage</td>
<td>3107 3092</td>
</tr>
<tr>
<td>AuthCharInd</td>
<td>1217 1172 1925</td>
</tr>
<tr>
<td>AuthCode</td>
<td>1142 942 1136</td>
</tr>
<tr>
<td>Authenticated</td>
<td>2734 2728</td>
</tr>
<tr>
<td>AuthHeader</td>
<td>1134 1127</td>
</tr>
<tr>
<td>AuthorityKeyIdentifier</td>
<td>2124 2120</td>
</tr>
<tr>
<td>authorizationKeyIdentifier</td>
<td>2119 2090</td>
</tr>
<tr>
<td>AuthReq</td>
<td>983 86</td>
</tr>
<tr>
<td>AuthReqData</td>
<td>990 983 988 1239</td>
</tr>
<tr>
<td>AuthReqItem</td>
<td>998 991 1352</td>
</tr>
<tr>
<td>AuthReqPayload</td>
<td>1015 1001</td>
</tr>
<tr>
<td>AuthReqTBE</td>
<td>986</td>
</tr>
<tr>
<td>AuthReqTBS</td>
<td>988 986</td>
</tr>
<tr>
<td>AuthRes</td>
<td>1069 87</td>
</tr>
<tr>
<td>AuthResBaggage</td>
<td>1096 1070 1071 1082</td>
</tr>
<tr>
<td>AuthResData</td>
<td>1089 1070 1071 1082</td>
</tr>
<tr>
<td>AuthResDataNew</td>
<td>1303 1284</td>
</tr>
<tr>
<td>AuthResPayload</td>
<td>1126 1093 1240 1305 1353</td>
</tr>
<tr>
<td>AuthResTBE</td>
<td>1075</td>
</tr>
<tr>
<td>AuthResTBS</td>
<td>1082 1075 1078 1085</td>
</tr>
<tr>
<td>AuthResTBSX</td>
<td>1084 1079</td>
</tr>
<tr>
<td>AuthRetNum</td>
<td>1259 1007 1254</td>
</tr>
<tr>
<td>AuthRevCode</td>
<td>1290 1279</td>
</tr>
<tr>
<td>AuthRevReq</td>
<td>1229 89</td>
</tr>
<tr>
<td>AuthRevReqBaggage</td>
<td>1247 1229 1234</td>
</tr>
<tr>
<td>AuthRevReqData</td>
<td>1236 1229 1234</td>
</tr>
<tr>
<td>AuthRevReqTBE</td>
<td>1232</td>
</tr>
<tr>
<td>AuthRevReqTBS</td>
<td>1234 1232</td>
</tr>
<tr>
<td>AuthRevRes</td>
<td>1261 90</td>
</tr>
<tr>
<td>AuthRevResBaggage</td>
<td>1273 1262 1271</td>
</tr>
<tr>
<td>AuthRevResData</td>
<td>1278 1262 1263 1267 1271</td>
</tr>
<tr>
<td>AuthRevResTBE</td>
<td>1267</td>
</tr>
<tr>
<td>Token</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>AuthRevResTBE</td>
<td>1269</td>
</tr>
<tr>
<td>AuthRevResTBS</td>
<td>1271 1269</td>
</tr>
<tr>
<td>AuthRevRRTags</td>
<td>1257 1253</td>
</tr>
<tr>
<td>AuthRevTags</td>
<td>1252 1237 1280</td>
</tr>
<tr>
<td>AuthStatus</td>
<td>940 935</td>
</tr>
<tr>
<td>AuthTags</td>
<td>1004 999 1090</td>
</tr>
<tr>
<td>AuthToken</td>
<td>1787 825 1099 1275</td>
</tr>
<tr>
<td>AuthTokenData</td>
<td>1800 1787 1791 1796</td>
</tr>
<tr>
<td>AuthTokenTBE</td>
<td>1790</td>
</tr>
<tr>
<td>AuthTokenTBS</td>
<td>1795 1792</td>
</tr>
<tr>
<td>AuthValCodes</td>
<td>1170 1163</td>
</tr>
<tr>
<td>AutoApplicableRate</td>
<td>3238 3230</td>
</tr>
<tr>
<td>AutoCharges</td>
<td>3243 3221</td>
</tr>
<tr>
<td>AutoNoShow</td>
<td>3224 3214</td>
</tr>
<tr>
<td>AutoRateInfo</td>
<td>3229 3218</td>
</tr>
<tr>
<td>AVSData</td>
<td>1030 1018</td>
</tr>
<tr>
<td>AVSResult</td>
<td>1205 1166</td>
</tr>
<tr>
<td>BackKey</td>
<td>248 245</td>
</tr>
<tr>
<td>BackKeyData</td>
<td>243 603 692 840 1102 1804</td>
</tr>
<tr>
<td>BARqExtensionsIOS</td>
<td>1638 1635</td>
</tr>
<tr>
<td>BARsExtensionsIOS</td>
<td>1674 1671</td>
</tr>
<tr>
<td>basicConstraints</td>
<td>2246 2096</td>
</tr>
<tr>
<td>BasicConstraintsSyntax</td>
<td>2252 2247</td>
</tr>
<tr>
<td>BASTatus</td>
<td>1691 1666</td>
</tr>
<tr>
<td>BatchAdminReq</td>
<td>1621 107</td>
</tr>
<tr>
<td>BatchAdminReqData</td>
<td>1626 1621 1624</td>
</tr>
<tr>
<td>BatchAdminReqTBE</td>
<td>1624</td>
</tr>
<tr>
<td>BatchAdminRes</td>
<td>1658 108</td>
</tr>
<tr>
<td>BatchAdminResData</td>
<td>1663 1658 1661</td>
</tr>
<tr>
<td>BatchAdminResTBE</td>
<td>1661</td>
</tr>
<tr>
<td>BatchID</td>
<td>1812 1387 1426 1457 1628 1665 1921</td>
</tr>
<tr>
<td>BatchOperation</td>
<td>1640 1630</td>
</tr>
<tr>
<td>BatchSequenceNum</td>
<td>1814 1388 1458 1755 1922</td>
</tr>
<tr>
<td>BatchStatus</td>
<td>1718 1138 1633 1667 1716</td>
</tr>
<tr>
<td>BatchStatusSeq</td>
<td>1716 1369 1439</td>
</tr>
<tr>
<td>BatchTotals</td>
<td>1739 1728 1736</td>
</tr>
<tr>
<td>BCIDistribution</td>
<td>225</td>
</tr>
<tr>
<td>BCIDistributionTBS</td>
<td>227 225</td>
</tr>
<tr>
<td>BIN</td>
<td>250 410 415 763 861 1589 2306</td>
</tr>
<tr>
<td>BIT STRING</td>
<td>ASN.1 2030 2038 2042 2139 2286</td>
</tr>
<tr>
<td>BMPString</td>
<td>ASN.1 3127 3132</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>ASN.1 269 466 467 993 1016 1021 1429 1631 1649 1931 2075 2253 2309 2324</td>
</tr>
<tr>
<td></td>
<td>2334 3088 3089 3091 3200</td>
</tr>
<tr>
<td>BrandAndBIN</td>
<td>1587 1585</td>
</tr>
<tr>
<td>BrandAndBINSeq</td>
<td>1585 1579 1629</td>
</tr>
<tr>
<td>BrandBatchDetails</td>
<td>1734 1732</td>
</tr>
<tr>
<td>BrandBatchDetailsSeq</td>
<td>1732 1729</td>
</tr>
<tr>
<td>BrandCRLIdentifier</td>
<td>191 229 502 531 568 650 777 909 1091 1281 1367 1437 1617</td>
</tr>
<tr>
<td>BrandCRLIdentifierSeq</td>
<td>1617 1597</td>
</tr>
<tr>
<td>BrandID</td>
<td>232 220 490 514 762 860 1588 1650 1735 1754</td>
</tr>
<tr>
<td>BSExtensionsIOS</td>
<td>1725 1722</td>
</tr>
<tr>
<td>BTExtensionsIOS</td>
<td>1747 1744</td>
</tr>
<tr>
<td>C</td>
<td>2944 811 968</td>
</tr>
<tr>
<td>CA</td>
<td>2948 209 216 225 494 519 537 557 575 576 636 637 641</td>
</tr>
<tr>
<td>CACKey</td>
<td>692 637</td>
</tr>
<tr>
<td>CAMsg</td>
<td>684 658</td>
</tr>
<tr>
<td>CapCode</td>
<td>1394 949 1385</td>
</tr>
<tr>
<td>CapItem</td>
<td>1343 1341</td>
</tr>
<tr>
<td>CapItemSeq</td>
<td>1341 1333</td>
</tr>
<tr>
<td>CapPayload</td>
<td>1349 1346 1425</td>
</tr>
<tr>
<td>CapReq</td>
<td>1310 92</td>
</tr>
<tr>
<td>CapReqData</td>
<td>1330 1311 1312 1323</td>
</tr>
<tr>
<td>CapReqTBE</td>
<td>1316</td>
</tr>
<tr>
<td>Capability</td>
<td>Value</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>CapReqTBEX</td>
<td>1318</td>
</tr>
<tr>
<td>CapReqTBS</td>
<td>1323</td>
</tr>
<tr>
<td>CapReqTBSX</td>
<td>1325</td>
</tr>
<tr>
<td>CapRes</td>
<td>1360</td>
</tr>
<tr>
<td>CapResData</td>
<td>1365</td>
</tr>
<tr>
<td>CapResItem</td>
<td>1376</td>
</tr>
<tr>
<td>CapResItemSeq</td>
<td>1376</td>
</tr>
<tr>
<td>CapResPayload</td>
<td>1384</td>
</tr>
<tr>
<td>CapResTBE</td>
<td>1363</td>
</tr>
<tr>
<td>CapRevData</td>
<td>1501</td>
</tr>
<tr>
<td>CapRevOrCredCode</td>
<td>1464</td>
</tr>
<tr>
<td>CapRevOrCredReqData</td>
<td>1411</td>
</tr>
<tr>
<td>CapRevOrCredReqItem</td>
<td>1422</td>
</tr>
<tr>
<td>CapRevOrCredReqItemSeq</td>
<td>1420</td>
</tr>
<tr>
<td>CapRevOrCredResData</td>
<td>1435</td>
</tr>
<tr>
<td>CapRevOrCredResItem</td>
<td>1448</td>
</tr>
<tr>
<td>CapRevOrCredResItemSeq</td>
<td>1446</td>
</tr>
<tr>
<td>CapRevOrCredResPayload</td>
<td>1454</td>
</tr>
<tr>
<td>CapRevReq</td>
<td>1481</td>
</tr>
<tr>
<td>CapRevReqTBE</td>
<td>1487</td>
</tr>
<tr>
<td>CapRevReqTBEX</td>
<td>1489</td>
</tr>
<tr>
<td>CapRevReqTBS</td>
<td>1494</td>
</tr>
<tr>
<td>CapRevReqTBSX</td>
<td>1496</td>
</tr>
<tr>
<td>CapRevRes</td>
<td>1503</td>
</tr>
<tr>
<td>CapRevResData</td>
<td>1508</td>
</tr>
<tr>
<td>CapRevResTBE</td>
<td>1506</td>
</tr>
<tr>
<td>CapRRTags</td>
<td>1339</td>
</tr>
<tr>
<td>CapStatus</td>
<td>947</td>
</tr>
<tr>
<td>CapToken</td>
<td>1816</td>
</tr>
<tr>
<td>CapTokenData</td>
<td>1835</td>
</tr>
<tr>
<td>CapTokenSeq</td>
<td>1841</td>
</tr>
<tr>
<td>CapTokenTBE</td>
<td>1823</td>
</tr>
<tr>
<td>CapTokenTBEX</td>
<td>1825</td>
</tr>
<tr>
<td>CapTokenTBS</td>
<td>1830</td>
</tr>
<tr>
<td>CardCertRequired</td>
<td>2323</td>
</tr>
<tr>
<td>CardCInitReq</td>
<td>486</td>
</tr>
<tr>
<td>CardCInitRes</td>
<td>494</td>
</tr>
<tr>
<td>CardCInitResTBS</td>
<td>496</td>
</tr>
<tr>
<td>CardExpiry</td>
<td>252</td>
</tr>
<tr>
<td>CardSuspect</td>
<td>1040</td>
</tr>
<tr>
<td>CardType</td>
<td>1186</td>
</tr>
<tr>
<td>CBC8Parameter</td>
<td>2789</td>
</tr>
<tr>
<td>CERT-POLICY-QUALIFIER</td>
<td>2189</td>
</tr>
<tr>
<td>Certificate</td>
<td>2023</td>
</tr>
<tr>
<td>certificatePolicies</td>
<td>2160</td>
</tr>
<tr>
<td>CertificatePoliciesSyntax</td>
<td>2166</td>
</tr>
<tr>
<td>Certificates</td>
<td>2746</td>
</tr>
<tr>
<td>CertificateSerialNumber</td>
<td>2015</td>
</tr>
<tr>
<td>certificateType</td>
<td>2280</td>
</tr>
<tr>
<td>CertificateTypeSyntax</td>
<td>2286</td>
</tr>
<tr>
<td>CertificateVersion</td>
<td>2013</td>
</tr>
<tr>
<td>CertificationRequest</td>
<td>3388</td>
</tr>
<tr>
<td>CertificationRequestInfo</td>
<td>3392</td>
</tr>
<tr>
<td>CertReq</td>
<td>696</td>
</tr>
<tr>
<td>CertReqTBS</td>
<td>698</td>
</tr>
<tr>
<td>CertReqRes</td>
<td>705</td>
</tr>
<tr>
<td>CertPolicyId</td>
<td>2174</td>
</tr>
<tr>
<td>CertReq</td>
<td>574</td>
</tr>
<tr>
<td>CertReqData</td>
<td>592</td>
</tr>
<tr>
<td>CertReqTBE</td>
<td>580</td>
</tr>
<tr>
<td>CertReqTBEX</td>
<td>582</td>
</tr>
<tr>
<td>CertReqTBS</td>
<td>587</td>
</tr>
<tr>
<td>CertRes</td>
<td>635</td>
</tr>
<tr>
<td>CertResData</td>
<td>643</td>
</tr>
<tr>
<td>Identifier</td>
<td>Value</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CertResTBE</td>
<td>641</td>
</tr>
<tr>
<td>CertStatus</td>
<td>654-648</td>
</tr>
<tr>
<td>CertStatusCode</td>
<td>669-655</td>
</tr>
<tr>
<td>Challenge</td>
<td>259-489-499-511-524-526-545-562-564-595-597-646-701-761-775</td>
</tr>
<tr>
<td>ChargeInfo</td>
<td>3175-3168</td>
</tr>
<tr>
<td>CheckDigests</td>
<td>1010-1000</td>
</tr>
<tr>
<td>CHOICE</td>
<td>ASN.1-75-144-159-392-404-442-574-618-635-787-822-963-1069-1261</td>
</tr>
<tr>
<td>CONTENTS</td>
<td>1310-1481-1512-1543-1816-1878-1884-1952-2232-3075-3125-3130-3238</td>
</tr>
<tr>
<td>CLASS</td>
<td>ASN.1-2073-2189-3082-3116</td>
</tr>
<tr>
<td>ClosedWhen</td>
<td>1706-1720</td>
</tr>
<tr>
<td>CloseStatus</td>
<td>1711-1707</td>
</tr>
<tr>
<td>CommercialCardData</td>
<td>3167-1927</td>
</tr>
<tr>
<td>commonName</td>
<td>3012-3048</td>
</tr>
<tr>
<td>CompletionCode</td>
<td>923-916</td>
</tr>
<tr>
<td>ContentEncryptionAlgorithms</td>
<td>2783-244-2777</td>
</tr>
<tr>
<td>ContentInfo</td>
<td>2702-2713-2810</td>
</tr>
<tr>
<td>CONTENTS</td>
<td>2695-2697-2704-2708</td>
</tr>
<tr>
<td>Contents</td>
<td>2697-2704-2708</td>
</tr>
<tr>
<td>ContentType</td>
<td>2708-2703-2735-2775</td>
</tr>
<tr>
<td>contentType</td>
<td>2984-2735</td>
</tr>
<tr>
<td>CountryCode</td>
<td>261-287-2308</td>
</tr>
<tr>
<td>countryName</td>
<td>3017-3045</td>
</tr>
<tr>
<td>CPayExtensionsIOS</td>
<td>1358-1355</td>
</tr>
<tr>
<td>CreditStatus</td>
<td>955-953</td>
</tr>
<tr>
<td>CreditStatusSeq</td>
<td>953-937</td>
</tr>
<tr>
<td>CredReq</td>
<td>1512-98</td>
</tr>
<tr>
<td>CredReqTBE</td>
<td>1532-1513-1514-1525</td>
</tr>
<tr>
<td>CredReqTBEX</td>
<td>1518-1520</td>
</tr>
<tr>
<td>CredReqTBS</td>
<td>1525-1518-1521-1528</td>
</tr>
<tr>
<td>CredReqTBSX</td>
<td>1527-1522</td>
</tr>
<tr>
<td>CredRes</td>
<td>1534-99</td>
</tr>
<tr>
<td>CredResData</td>
<td>1539-1534-1537</td>
</tr>
<tr>
<td>CredResTBE</td>
<td>1537-</td>
</tr>
<tr>
<td>CredRevReq</td>
<td>1543-101</td>
</tr>
<tr>
<td>CredRevReqData</td>
<td>1563-1544-1545-1556</td>
</tr>
<tr>
<td>CredRevReqTBE</td>
<td>1549-</td>
</tr>
<tr>
<td>CredRevReqTBEX</td>
<td>1551-</td>
</tr>
<tr>
<td>CredRevReqTBS</td>
<td>1556-1549-1552-1559</td>
</tr>
<tr>
<td>CredRevReqTBSX</td>
<td>1558-1553</td>
</tr>
<tr>
<td>CredRevRes</td>
<td>1565-102</td>
</tr>
<tr>
<td>CredRevResData</td>
<td>1570-1565-1568</td>
</tr>
<tr>
<td>CredRevResTBE</td>
<td>1568-</td>
</tr>
<tr>
<td>CRL</td>
<td>2654-2668</td>
</tr>
<tr>
<td>CRLEntry</td>
<td>2646-2644</td>
</tr>
<tr>
<td>CRLEntryList</td>
<td>2644-2640</td>
</tr>
<tr>
<td>CRLIdentifier</td>
<td>236-234</td>
</tr>
<tr>
<td>CRLIdentifierSeq</td>
<td>234-203</td>
</tr>
<tr>
<td>CRLNotification</td>
<td>209-</td>
</tr>
<tr>
<td>CRLNotificationRes</td>
<td>216-</td>
</tr>
<tr>
<td>CRLNotificationResTBS</td>
<td>218-216</td>
</tr>
<tr>
<td>CRLNotificationTBS</td>
<td>211-209</td>
</tr>
<tr>
<td>CRLNumber</td>
<td>2264-2260</td>
</tr>
<tr>
<td>cRLNumber</td>
<td>2259-2097</td>
</tr>
<tr>
<td>CRLSequence</td>
<td>2688-2715</td>
</tr>
<tr>
<td>CRqExtensionsIOS</td>
<td>1337-1334</td>
</tr>
<tr>
<td>CRsExtensionsIOS</td>
<td>1374-1371</td>
</tr>
<tr>
<td>CRsPayExtensionsIOS</td>
<td>1392-1389</td>
</tr>
<tr>
<td>CRvRqExtensionsIOS</td>
<td>1418-1415</td>
</tr>
<tr>
<td>CRvRqItemExtensionsIOS</td>
<td>1433-1430</td>
</tr>
<tr>
<td>CRvRsExtensionsIOS</td>
<td>1444-1441</td>
</tr>
<tr>
<td>CRvRsPayExtensionsIOS</td>
<td>1462-1459</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CurrConv</td>
<td>1853</td>
</tr>
<tr>
<td>Currency</td>
<td>205</td>
</tr>
<tr>
<td>CurrencyAmount</td>
<td>1843</td>
</tr>
<tr>
<td>Date</td>
<td>205</td>
</tr>
<tr>
<td>DateTime</td>
<td>205</td>
</tr>
<tr>
<td>DD</td>
<td>205</td>
</tr>
<tr>
<td>DetachedDigest</td>
<td>205</td>
</tr>
<tr>
<td>Digest</td>
<td>205</td>
</tr>
<tr>
<td>DigestAlgorithmIdentifiers</td>
<td>205</td>
</tr>
<tr>
<td>DigestAlgorithms</td>
<td>205</td>
</tr>
<tr>
<td>DigestedData</td>
<td>205</td>
</tr>
<tr>
<td>digestedData</td>
<td>205</td>
</tr>
<tr>
<td>DigestEncryptionAlgorithms</td>
<td>205</td>
</tr>
<tr>
<td>Digits</td>
<td>205</td>
</tr>
<tr>
<td>DirectoryString</td>
<td>205</td>
</tr>
<tr>
<td>Distance</td>
<td>205</td>
</tr>
<tr>
<td>DistanceScale</td>
<td>205</td>
</tr>
<tr>
<td>Duration</td>
<td>205</td>
</tr>
<tr>
<td>E</td>
<td>205</td>
</tr>
<tr>
<td>EE</td>
<td>205</td>
</tr>
<tr>
<td>EH</td>
<td>205</td>
</tr>
<tr>
<td>EK</td>
<td>205</td>
</tr>
<tr>
<td>Enc</td>
<td>205</td>
</tr>
<tr>
<td>EncB</td>
<td>205</td>
</tr>
<tr>
<td>EncBX</td>
<td>205</td>
</tr>
<tr>
<td>EncK</td>
<td>205</td>
</tr>
<tr>
<td>EncryptedBrandCRLID</td>
<td>205</td>
</tr>
<tr>
<td>EncryptedCertificate</td>
<td>205</td>
</tr>
<tr>
<td>EncryptedCertificationRequestInfo</td>
<td>205</td>
</tr>
<tr>
<td>EncryptedCRL</td>
<td>205</td>
</tr>
<tr>
<td>EncryptedContent</td>
<td>205</td>
</tr>
<tr>
<td>EncryptedContentInfo</td>
<td>205</td>
</tr>
<tr>
<td>EncryptedData</td>
<td>205</td>
</tr>
<tr>
<td>EncryptedDigest</td>
<td>205</td>
</tr>
<tr>
<td>EncryptedKey</td>
<td>205</td>
</tr>
<tr>
<td>encryptionAlgorithm</td>
<td>205</td>
</tr>
<tr>
<td>EncX</td>
<td>205</td>
</tr>
<tr>
<td>ENTITY-IDENTIFIER</td>
<td>205</td>
</tr>
<tr>
<td>ENUMERATED</td>
<td>205</td>
</tr>
<tr>
<td>EnvelopedData</td>
<td>205</td>
</tr>
<tr>
<td>envelopedData</td>
<td>205</td>
</tr>
<tr>
<td>Error</td>
<td>205</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>205</td>
</tr>
<tr>
<td>ErrorMsg</td>
<td>205</td>
</tr>
<tr>
<td>ErrorTBS</td>
<td>205</td>
</tr>
<tr>
<td>EX</td>
<td>205</td>
</tr>
<tr>
<td>EXH</td>
<td>205</td>
</tr>
<tr>
<td>EXTENSION</td>
<td>205</td>
</tr>
<tr>
<td>Extension</td>
<td>205</td>
</tr>
<tr>
<td>Extensions</td>
<td>205</td>
</tr>
<tr>
<td>ExtensionSet</td>
<td>205</td>
</tr>
<tr>
<td>FailedItem</td>
<td>205</td>
</tr>
</tbody>
</table>

**Version 1.0**
<table>
<thead>
<tr>
<th>FailedItemSeq</th>
<th>FieldName</th>
<th>FieldValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>FloatingPoint</td>
<td>GeneralizedTime</td>
<td>ASN.1</td>
</tr>
<tr>
<td>GeneralName</td>
<td>GeneralNames</td>
<td>2230 2126 2226 2240 3405</td>
</tr>
<tr>
<td>H</td>
<td></td>
<td>2835</td>
</tr>
<tr>
<td>HashedRootKey</td>
<td>HMAC</td>
<td>2840 851</td>
</tr>
<tr>
<td>HMACPanData</td>
<td>HOD</td>
<td>870 847 857 1012</td>
</tr>
<tr>
<td>HOIDInput</td>
<td>HOIData</td>
<td>820 815 1011</td>
</tr>
<tr>
<td>HotelCharges</td>
<td>HotelNoShow</td>
<td>3274 3263</td>
</tr>
<tr>
<td>HotelRateInfo</td>
<td>HPIData</td>
<td>818 814</td>
</tr>
<tr>
<td>IA5String</td>
<td>id-at</td>
<td>3143 3144 3145 3146 3147</td>
</tr>
<tr>
<td>id-at-commonName</td>
<td>id-at-countryName</td>
<td>3145 3019</td>
</tr>
<tr>
<td>id-at-organizationalUnitName</td>
<td>id-at-organizationName</td>
<td>3146 3024</td>
</tr>
<tr>
<td>id-ce</td>
<td>2368 2359 2360 2361 2362 2363 2364 2365 2366</td>
<td></td>
</tr>
<tr>
<td>id-ce-authorizationKeyIdentifier</td>
<td>id-ce-basicConstraints</td>
<td>2363 2249</td>
</tr>
<tr>
<td>id-ce-certificatePolicies</td>
<td>id-ce-cRLNumber</td>
<td>2364 2261</td>
</tr>
<tr>
<td>id-ce-issuerAltName</td>
<td>id-ce-keyUsage</td>
<td>2362 2241</td>
</tr>
<tr>
<td>id-ce-privkeyUsagePeriod</td>
<td>id-ce-privateKeyUsagePeriod</td>
<td>2360 2151 3404</td>
</tr>
<tr>
<td>id-ce-subjectAltName</td>
<td>id-desCBC</td>
<td>2968 2785</td>
</tr>
<tr>
<td>id-desCDMF</td>
<td>id-rsaEncryption</td>
<td>2962 2757</td>
</tr>
<tr>
<td>id-set</td>
<td>2368 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385</td>
<td></td>
</tr>
<tr>
<td>id-set-accountNumber</td>
<td>id-set-additionalPolicy</td>
<td>2487 2494 3414</td>
</tr>
<tr>
<td>id-set-address</td>
<td>id-set-algorithm</td>
<td>2379</td>
</tr>
<tr>
<td>id-set-AmericanExpress</td>
<td>id-set-amount</td>
<td>2486</td>
</tr>
<tr>
<td>id-set-Antares</td>
<td></td>
<td>2556</td>
</tr>
<tr>
<td>id-set-attribute</td>
<td>id-set-birthFamilyName</td>
<td>2479</td>
</tr>
<tr>
<td>id-set-birthFamilyName</td>
<td>id-set-BlueMoney</td>
<td>2566</td>
</tr>
<tr>
<td>id-set-brand</td>
<td>id-set-cardCertRequired</td>
<td>2506 2325</td>
</tr>
<tr>
<td>id-set-cardCertRequired</td>
<td>id-set-certExt</td>
<td>2382 2503 2504 2505 2506 2507 2508 2509</td>
</tr>
<tr>
<td>id-set-Certicom</td>
<td>id-set-certificateType</td>
<td>2504 2283 3409</td>
</tr>
<tr>
<td>id-set-CompuSource</td>
<td>id-set-Content-AcqCardCodeMsg</td>
<td>2546 2410</td>
</tr>
<tr>
<td>id-set-Content-AcqCardCodeMsgTBE</td>
<td>id-set-Content-AuthReqTBE</td>
<td>2446 2440</td>
</tr>
<tr>
<td>id-set-Content-AuthReqTBE</td>
<td>id-set-Content-AuthResBaggage</td>
<td>2404</td>
</tr>
<tr>
<td>id-set-Content-AuthResBaggage</td>
<td>id-set-Content-AuthResTBE</td>
<td>2396 2441</td>
</tr>
<tr>
<td>id-set-Content-AuthResTBE</td>
<td>id-set-Content-AuthResTBEX</td>
<td>2442</td>
</tr>
<tr>
<td>id-set-Content-AuthResTBEX</td>
<td>id-set-Content-AuthResTBS</td>
<td>2405</td>
</tr>
<tr>
<td>id-set-content-AuthResTBSX</td>
<td>2406</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthRevReqBaggage</td>
<td>2397</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthRevReqTBE</td>
<td>2447</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthRevReqTBS</td>
<td>2411</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthRevResBaggage</td>
<td>2398</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthRevResData</td>
<td>2412</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthRevResTBE</td>
<td>2448</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthRevResTBE</td>
<td>2449</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthRevResTBS</td>
<td>2413</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthTokenTBE</td>
<td>2443</td>
<td></td>
</tr>
<tr>
<td>id-set-content-AuthTokenTBS</td>
<td>2407</td>
<td></td>
</tr>
<tr>
<td>id-set-content-BatchAdminReqData</td>
<td>2428</td>
<td></td>
</tr>
<tr>
<td>id-set-content-BatchAdminReqTBE</td>
<td>2462</td>
<td></td>
</tr>
<tr>
<td>id-set-content-BatchAdminResData</td>
<td>2429</td>
<td></td>
</tr>
<tr>
<td>id-set-content-BatchAdminResTBE</td>
<td>2463</td>
<td></td>
</tr>
<tr>
<td>id-set-content-BCIDistributionTBS</td>
<td>2470</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapReqTBE</td>
<td>2450</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapReqTBEX</td>
<td>2451</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapReqTBS</td>
<td>2414</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapReqTBSX</td>
<td>2415</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapReqData</td>
<td>2416</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapResTBE</td>
<td>2452</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapRevReqTBE</td>
<td>2453</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapRevReqTBEX</td>
<td>2454</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapRevReqTBS</td>
<td>2417</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapRevReqTBSX</td>
<td>2418</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapRevResData</td>
<td>2419</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapRevResTBE</td>
<td>2455</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapTokenData</td>
<td>2408</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapTokenSeq</td>
<td>2399</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapTokenTBE</td>
<td>2444</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapTokenTBEX</td>
<td>2445</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapTokenTBS</td>
<td>2409</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CapTokenTBSX</td>
<td>2410</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CertInqReqTBS</td>
<td>2436</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CertReqData</td>
<td>2433</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CertReqTBE</td>
<td>2465</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CertReqTBEX</td>
<td>2466</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CertReqTBS</td>
<td>2434</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CertResData</td>
<td>2435</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CertResTBE</td>
<td>2467</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredReqTBE</td>
<td>2456</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredReqTBEX</td>
<td>2457</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredReqTBS</td>
<td>2420</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredReqTBSX</td>
<td>2421</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredResData</td>
<td>2422</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredResTBE</td>
<td>2458</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredRevReqTBE</td>
<td>2459</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredRevReqTBEX</td>
<td>2460</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredRevReqTBS</td>
<td>2423</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredRevReqTBSX</td>
<td>2424</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredRevResData</td>
<td>2425</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CredRevResTBE</td>
<td>2461</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CRLNotificationResTBS</td>
<td>2469</td>
<td></td>
</tr>
<tr>
<td>id-set-content-CRLNotificationTBS</td>
<td>2468</td>
<td></td>
</tr>
<tr>
<td>id-set-content-ErrorTBS</td>
<td>2437</td>
<td></td>
</tr>
<tr>
<td>id-set-content-HODInput</td>
<td>2395</td>
<td></td>
</tr>
<tr>
<td>id-set-content-InqReqData</td>
<td>2403</td>
<td></td>
</tr>
<tr>
<td>id-set-content-Me-AcCInitResTBS</td>
<td>2431</td>
<td></td>
</tr>
<tr>
<td>id-set-content-OfData</td>
<td>2391</td>
<td></td>
</tr>
<tr>
<td>id-set-content-PANData</td>
<td>2388</td>
<td></td>
</tr>
<tr>
<td>id-set-content-PANOnly</td>
<td>2390</td>
<td></td>
</tr>
<tr>
<td>id-set-content-PANtoken</td>
<td>2389</td>
<td></td>
</tr>
<tr>
<td>id-set-content-PCertReqData</td>
<td>2426</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PCertResTBS</td>
<td>2427</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PI</td>
<td>2392</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PI-TBS</td>
<td>2401</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PIData</td>
<td>2393</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PIDataSigned</td>
<td>2394</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PIDataUnsigned</td>
<td>2394</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PIInitResData</td>
<td>2438</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PInitResData</td>
<td>2400</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PIUnsignedTBE</td>
<td>2439</td>
<td></td>
</tr>
<tr>
<td>id-set-content:PResData</td>
<td>2402</td>
<td></td>
</tr>
<tr>
<td>id-set-content:RegFormReqTBE</td>
<td>2464</td>
<td></td>
</tr>
<tr>
<td>id-set-content:RegFormResTBS</td>
<td>2432</td>
<td></td>
</tr>
</tbody>
</table>

| id-set-contentType           | 2375 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 |
| id-set-date                  | 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 |
| id-set-Diners                | 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 |
| id-set-e-COMM                | 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 |
| id-set-ECC                   | 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 |
| id-set-espace-net            | 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 |

| id-set-Cybercash             | 2532 |
| id-set-data                 | 2482 |
| id-set-Diners               | 2514 |
| id-set-e-COMM               | 2602 |
| id-set-ECC                  | 2558 |
| id-set-eLab                 | 2572 |
| id-set-Entrust              | 2574 |
| id-set-espacenet            | 2590 |
| id-set-familyName           | 2478 |

<p>| id-set-field                | 2377 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 |
| id-set-Fujitsu              | 2570 |
| id-set-fullName             | 2476 |
| id-set-Gemplus              | 2604 |
| id-set-givenName            | 2477 |
| id-set-GlobeSet             | 2528 |
| id-set-Griffin              | 2548 |
| id-set-GTE                  | 2544 |
| id-set-hashedRootKey        | 2503 2271 |
| id-set-Hitachi              | 2592 |
| id-set-IATA-ATA             | 2512 |
| id-set-IBM                  | 2530 |
| id-set-identificationNumber | 2481 |
| id-set-III                 | 2578 |
| id-set-Intertrader          | 2584 |
| id-set-Japan                | 2609 |
| id-set-JCB                  | 2518 |
| id-set-Lacerte              | 2568 |
| id-set-Lexem                | 2582 |
| id-set-Mailthean           | 2560 |
| id-set-MasterCard           | 2522 |
| id-set-merchantData         | 2505 2301 |
| id-set-Microsoft            | 2594 |
| id-set-Mitsubishi           | 2598 |
| id-set-module               | 2381 |
| id-set-month                | 2482 |
| id-set-msgExt               | 2376 |
| id-set-NABLE                | 2588 |
| id-set-national             | 2385 2609 |
| id-set-NCR                  | 2600 |
| id-set-NEC                  | 2596 |
| id-set-Netscape             | 2562 |
| id-set-Novus                | 2524 |
| id-set-OpenMarket           | 2580 |
| id-set-OSS                  | 2552 |
| id-set-passPhrase           | 2488 |
| id-set-Persimmon            | 2586 |
| id-set-placeName            | 2480 |</p>
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id-set-policy</td>
<td>2380</td>
</tr>
<tr>
<td>id-set-policy-root</td>
<td>2500</td>
</tr>
<tr>
<td>id-set-rootKeyIdThumb</td>
<td>2493</td>
</tr>
<tr>
<td>id-set-RSADSI</td>
<td>2536</td>
</tr>
<tr>
<td>id-set-setExtensions</td>
<td>2508</td>
</tr>
<tr>
<td>id-set-setQualifier</td>
<td>2509</td>
</tr>
<tr>
<td>id-set-telephone</td>
<td>2485</td>
</tr>
<tr>
<td>id-set-TenthMountain</td>
<td>2554</td>
</tr>
<tr>
<td>id-set-Terisa</td>
<td>2534</td>
</tr>
<tr>
<td>id-set-Trintech</td>
<td>2540</td>
</tr>
<tr>
<td>id-set-tunneling</td>
<td>2507</td>
</tr>
<tr>
<td>id-set-vendor</td>
<td>2384</td>
</tr>
<tr>
<td>id-set-VeriFone</td>
<td>2538</td>
</tr>
<tr>
<td>id-set-VeriSign</td>
<td>2564</td>
</tr>
<tr>
<td>id-set-VIAnet</td>
<td>2576</td>
</tr>
<tr>
<td>id-set-Visa</td>
<td>2520</td>
</tr>
<tr>
<td>id-sha1</td>
<td>2966</td>
</tr>
<tr>
<td>id-sha1-with-rsa-signature</td>
<td>2964</td>
</tr>
<tr>
<td>IDData</td>
<td>404</td>
</tr>
<tr>
<td>identified-organization</td>
<td>ASN.1</td>
</tr>
<tr>
<td>Inputs</td>
<td>846</td>
</tr>
<tr>
<td>InqReq</td>
<td>963</td>
</tr>
<tr>
<td>InqReqData</td>
<td>970</td>
</tr>
<tr>
<td>InqReqSigned</td>
<td>968</td>
</tr>
<tr>
<td>InqRes</td>
<td>979</td>
</tr>
<tr>
<td>InstallRecurData</td>
<td>1945</td>
</tr>
<tr>
<td>InstallRecurInd</td>
<td>1952</td>
</tr>
<tr>
<td>INTEGER</td>
<td>ASN.1</td>
</tr>
<tr>
<td>InternationalRA</td>
<td>ASN.1</td>
</tr>
<tr>
<td>IssuerAltName</td>
<td>2239</td>
</tr>
<tr>
<td>IssuerAndSerialNumber</td>
<td>2690</td>
</tr>
<tr>
<td>Item</td>
<td>3192</td>
</tr>
<tr>
<td>ItemSeq</td>
<td>3190</td>
</tr>
<tr>
<td>Joint-iso-itu-t</td>
<td>1950</td>
</tr>
<tr>
<td>KeyEncryptionAlgorithms</td>
<td>2798</td>
</tr>
<tr>
<td>KeyIdentifier</td>
<td>2131</td>
</tr>
<tr>
<td>KeyUsage</td>
<td>2139</td>
</tr>
<tr>
<td>keyUsage</td>
<td>2133</td>
</tr>
<tr>
<td>Language</td>
<td>282</td>
</tr>
<tr>
<td>LocalID</td>
<td>284</td>
</tr>
<tr>
<td>Location</td>
<td>286</td>
</tr>
<tr>
<td>LogRefID</td>
<td>1213</td>
</tr>
<tr>
<td>MarketAutoAuth</td>
<td>1860</td>
</tr>
<tr>
<td>MarketAutoCap</td>
<td>3210</td>
</tr>
<tr>
<td>MarketHotelAuth</td>
<td>1864</td>
</tr>
<tr>
<td>MarketHotelCap</td>
<td>3261</td>
</tr>
</tbody>
</table>

**Version 1.0**
<table>
<thead>
<tr>
<th><strong>MarketSpecAuthData</strong></th>
<th>1878 1023</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MarketSpecCapData</strong></td>
<td>1884 1892</td>
</tr>
<tr>
<td><strong>MarketSpecDataID</strong></td>
<td>1897 1174 1891</td>
</tr>
<tr>
<td><strong>MarketSpecSaleData</strong></td>
<td>1890 1926</td>
</tr>
<tr>
<td><strong>MarketTransportAuth</strong></td>
<td>1895 1881</td>
</tr>
<tr>
<td><strong>MarketTransportCap</strong></td>
<td>3303 1887</td>
</tr>
<tr>
<td><strong>MATCHING-RULE</strong></td>
<td>3116 3085 3086 3087</td>
</tr>
<tr>
<td><strong>Me-AgClntReq</strong></td>
<td>508 113</td>
</tr>
<tr>
<td><strong>Me-AgClntRes</strong></td>
<td>519 114</td>
</tr>
<tr>
<td><strong>Me-AgClntResTBS</strong></td>
<td>521 519</td>
</tr>
<tr>
<td><strong>member-body</strong></td>
<td>ASN.1 2958 2971 2974 2982</td>
</tr>
<tr>
<td><strong>merchantData</strong></td>
<td>2299 2103</td>
</tr>
<tr>
<td><strong>MerchantDataSyntax</strong></td>
<td>2304 2300</td>
</tr>
<tr>
<td><strong>MerchantID</strong></td>
<td>294 411 836 1803 1907 2305</td>
</tr>
<tr>
<td><strong>MerCatCode</strong></td>
<td>1054 1050</td>
</tr>
<tr>
<td><strong>MerchantData</strong></td>
<td>1049 1024</td>
</tr>
<tr>
<td><strong>MerchantGroup</strong></td>
<td>1058 1051</td>
</tr>
<tr>
<td><strong>MerNames</strong></td>
<td>2314 2312</td>
</tr>
<tr>
<td><strong>MerNameSeq</strong></td>
<td>2312 2307</td>
</tr>
<tr>
<td><strong>MerOrderNum</strong></td>
<td>1904 1924</td>
</tr>
<tr>
<td><strong>MerTermIDs</strong></td>
<td>1906 1916</td>
</tr>
<tr>
<td><strong>MESSAGE</strong></td>
<td>73 45</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>75 45</td>
</tr>
<tr>
<td><strong>MessageDigest</strong></td>
<td>2740 2736</td>
</tr>
<tr>
<td><strong>messageDigest</strong></td>
<td>2986 2736</td>
</tr>
<tr>
<td><strong>MessageHeader</strong></td>
<td>58 44 160</td>
</tr>
<tr>
<td><strong>MessageIDs</strong></td>
<td>67 62</td>
</tr>
<tr>
<td><strong>MessageWrapper</strong></td>
<td>43</td>
</tr>
<tr>
<td><strong>MsgExtension</strong></td>
<td>137 135</td>
</tr>
<tr>
<td><strong>MsgExtensions</strong></td>
<td>134 46 765 780 841 863 877 918 974 1025 1129 1242 1285 1334 1355</td>
</tr>
<tr>
<td></td>
<td>1371 1389 1415 1430 1441 1459 1580 1598 1635 1661 1717 1722 1744 1760 1932</td>
</tr>
<tr>
<td><strong>NULL</strong></td>
<td>ASN.1 2189 1895 2752 2757 2799 3071</td>
</tr>
<tr>
<td><strong>NumericString</strong></td>
<td>ASN.1 2154 462 868 2074 2174 2190 2235 2338 2345 2358 2359 2360 2361 2362</td>
</tr>
<tr>
<td></td>
<td>2363 2364 2365 2366 2368 2373 2954 2957 2960 2962 2964 2966 2968 2970</td>
</tr>
<tr>
<td></td>
<td>2973 2976 2977 2978 2979 2981 2984 2986 3093 3118 3141 3143 3144 3145</td>
</tr>
<tr>
<td></td>
<td>3146 3147</td>
</tr>
<tr>
<td><strong>OD</strong></td>
<td>882 873</td>
</tr>
<tr>
<td><strong>ODExtensionsIOS</strong></td>
<td>880 877</td>
</tr>
<tr>
<td><strong>OID</strong></td>
<td>2373 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2388 2389 2390</td>
</tr>
<tr>
<td></td>
<td>2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404</td>
</tr>
<tr>
<td></td>
<td>2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418</td>
</tr>
<tr>
<td></td>
<td>2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432</td>
</tr>
<tr>
<td></td>
<td>2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446</td>
</tr>
<tr>
<td></td>
<td>2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460</td>
</tr>
<tr>
<td></td>
<td>2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2476 2477 2478 2479</td>
</tr>
<tr>
<td></td>
<td>2480 2481 2482 2483 2484 2485 2486 2487 2488 2491 2493 2494 2500 2503</td>
</tr>
<tr>
<td></td>
<td>2504 2505 2506 2507 2508 2509 2512 2514 2516 2518 2520 2522 2524 2528</td>
</tr>
<tr>
<td></td>
<td>2530 2532 2534 2536 2538 2540 2542 2544 2546 2548 2550 2552 2554 2556</td>
</tr>
<tr>
<td></td>
<td>2558 2560 2562 2564 2566 2568 2570 2572 2574 2576 2578 2580 2582 2584</td>
</tr>
<tr>
<td></td>
<td>2586 2588 2590 2592 2594 2596 2598 2600 2602 2604 2609</td>
</tr>
<tr>
<td><strong>OIData</strong></td>
<td>853 807 809 820 891</td>
</tr>
<tr>
<td><strong>OIDList</strong></td>
<td>868 862</td>
</tr>
<tr>
<td><strong>OIDSigned</strong></td>
<td>809 796</td>
</tr>
<tr>
<td><strong>OIDunsigned</strong></td>
<td>866 863</td>
</tr>
<tr>
<td><strong>OIExtensionsIOS</strong></td>
<td>881 888</td>
</tr>
<tr>
<td><strong>oiw</strong></td>
<td>ASN.1 2955</td>
</tr>
<tr>
<td>Reason</td>
<td>476 471</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>RecipientInfo</td>
<td>2791 2772</td>
</tr>
<tr>
<td>RecipientInfos</td>
<td>2772 2768</td>
</tr>
<tr>
<td>Recurring</td>
<td>1957 1954</td>
</tr>
<tr>
<td>ReferralData</td>
<td>470 444</td>
</tr>
<tr>
<td>ReferralURL</td>
<td>480 478</td>
</tr>
<tr>
<td>ReferralURLSeq</td>
<td>478 472</td>
</tr>
<tr>
<td>RegField</td>
<td>461 459 529</td>
</tr>
<tr>
<td>RegFieldSeq</td>
<td>459 456</td>
</tr>
<tr>
<td>RegForm</td>
<td>609 602</td>
</tr>
<tr>
<td>RegFormData</td>
<td>447 443</td>
</tr>
<tr>
<td>RegFormItem</td>
<td>611 609</td>
</tr>
<tr>
<td>RegFormOrReferral</td>
<td>442 528 567</td>
</tr>
<tr>
<td>RegFormReq</td>
<td>537 116</td>
</tr>
<tr>
<td>RegFormReqData</td>
<td>542 537 540</td>
</tr>
<tr>
<td>RegFormReqTBE</td>
<td>540</td>
</tr>
<tr>
<td>RegFormRes</td>
<td>557 117</td>
</tr>
<tr>
<td>RegFormResTBS</td>
<td>559 557</td>
</tr>
<tr>
<td>RegTemplate</td>
<td>452 448</td>
</tr>
<tr>
<td>ReimbursementID</td>
<td>1775 1756</td>
</tr>
<tr>
<td>RelativeDistinguishedName</td>
<td>3080 3078</td>
</tr>
<tr>
<td>RequestType</td>
<td>421 512 527 547 566 598</td>
</tr>
<tr>
<td>ResponseData</td>
<td>1162 1137</td>
</tr>
<tr>
<td>RespReason</td>
<td>1177 1164</td>
</tr>
<tr>
<td>Restrictions</td>
<td>3331 3311</td>
</tr>
<tr>
<td>Results</td>
<td>933 917</td>
</tr>
<tr>
<td>ReturnTransactionDetail</td>
<td>1646 1632</td>
</tr>
<tr>
<td>RootKeyThumb</td>
<td>2276 2274</td>
</tr>
<tr>
<td>RRPID</td>
<td>324 63 487 497 509 522 543 560 593 644 699 757 774 855 907</td>
</tr>
<tr>
<td></td>
<td>972 1345 1380 1424 1450 1753 1836 1915</td>
</tr>
<tr>
<td>RRTags</td>
<td>1914 1005 1257 1339 1412 1436 1577 1595 1627 1664</td>
</tr>
<tr>
<td>rsaOAEPEncryptionSET</td>
<td>2980 2599</td>
</tr>
<tr>
<td>s</td>
<td>2849 149 209 216 225 494 519 557 580 636 641 696 770 903 968</td>
</tr>
<tr>
<td></td>
<td>986 1075 1107 1232 1267 1289 1316 1363 1487 1506 1518 1537 1549 1568</td>
</tr>
<tr>
<td></td>
<td>1574 1592 1624 1661 1823 2871 2879</td>
</tr>
<tr>
<td>SaleDetail</td>
<td>1920 994 1354</td>
</tr>
<tr>
<td>SaleExtensionsIOS</td>
<td>1935 1932</td>
</tr>
<tr>
<td>Secret</td>
<td>326 303 310 851</td>
</tr>
<tr>
<td>seccig</td>
<td>2954 2955 2966 2968</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>ASN.1</td>
</tr>
<tr>
<td></td>
<td>267 272 286 300 307 314 330 337 397 409 414 447 452 459</td>
</tr>
<tr>
<td></td>
<td>461 470 478 486 496 508 521 542 559 582 587 592 609</td>
</tr>
<tr>
<td></td>
<td>611 623 643 654 662 664 684 698 756 772 794 913 813 828</td>
</tr>
<tr>
<td></td>
<td>833 846 853 868 872 886 903 915 933 940 947 953</td>
</tr>
<tr>
<td></td>
<td>955 970 990 998 1004 1010 1015 1030 1049 1077 1084 1089 1096 1109</td>
</tr>
<tr>
<td></td>
<td>1120 1126 1134 1162 1170 1236 1247 1292 1273 1278 1303 1318 1325 1330</td>
</tr>
<tr>
<td></td>
<td>1341 1343 1349 1365 1376 1378 1384 1411 1420 1422 1435 1446 1448 1454</td>
</tr>
<tr>
<td></td>
<td>1489 1496 1520 1527 1551 1558 1576 1585 1587 1594 1603 1605 1617 1626</td>
</tr>
<tr>
<td></td>
<td>1646 1653 1663 1684 1706 1716 1718 1727 1732 1734 1739 1749 1751 1790</td>
</tr>
<tr>
<td></td>
<td>1795 1800 1825 1830 1835 1841 1843 1853 1860 1864 1890 1906 1914 1920</td>
</tr>
<tr>
<td></td>
<td>1945 1957 2000 2027 2033 2040 2084 2110 2124 2154 2166 2168 2170 2176</td>
</tr>
<tr>
<td></td>
<td>2203 2208 2210 2216 2230 2252 2276 2304 2312 2314 2333 2338 2345 2364</td>
</tr>
<tr>
<td></td>
<td>2644 2646 2688 2690 2702 2710 2719 2723 2742 2746 2749 2761 2766 2772</td>
</tr>
<tr>
<td></td>
<td>2774 2791 2807 2817 2821 2844 2887 2899 2905 3034 3039 3053 3078</td>
</tr>
<tr>
<td></td>
<td>3167 3175 3190 3192 3210 3229 3243 3261 3278 3283 3303 3314 3316 3392</td>
</tr>
<tr>
<td>SET</td>
<td>ASN.1</td>
</tr>
<tr>
<td>set</td>
<td>ASN.1</td>
</tr>
<tr>
<td>setExtensions</td>
<td>2340 2106</td>
</tr>
<tr>
<td>SETExtensionsSyntax</td>
<td>2345 2341</td>
</tr>
<tr>
<td>SetPolicyQualifier</td>
<td>2203 2200</td>
</tr>
<tr>
<td>selfPolicyQualifier</td>
<td>2198 2185</td>
</tr>
<tr>
<td>SETQualifier</td>
<td>2216 2204 2212</td>
</tr>
<tr>
<td>SETString</td>
<td>3130 232 288 289 290 291 294 322 464 476 482 616 619 657 666</td>
</tr>
<tr>
<td>Field</td>
<td>Start Position</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>SettlementInfo</td>
<td>1684</td>
</tr>
<tr>
<td>sha1-with-rsa-signature</td>
<td>3070</td>
</tr>
<tr>
<td>SignatureAlgorithms</td>
<td>3065</td>
</tr>
<tr>
<td>SIGNED</td>
<td>2027</td>
</tr>
<tr>
<td>SignedData</td>
<td>2710</td>
</tr>
<tr>
<td>signedData</td>
<td>2977</td>
</tr>
<tr>
<td>SignedError</td>
<td>149</td>
</tr>
<tr>
<td>SignerInfo</td>
<td>2723</td>
</tr>
<tr>
<td>SignerInfos</td>
<td>2719</td>
</tr>
<tr>
<td>SO</td>
<td>2856</td>
</tr>
<tr>
<td>SpecialProcessing</td>
<td>1035</td>
</tr>
<tr>
<td>StopOverCode</td>
<td>3326</td>
</tr>
<tr>
<td>subjectAltName</td>
<td>2225</td>
</tr>
<tr>
<td>SubjectPublicKeyInfo</td>
<td>2040</td>
</tr>
<tr>
<td>SupportedAlgorithms</td>
<td>3059</td>
</tr>
<tr>
<td>SupportedAttributes</td>
<td>3044</td>
</tr>
<tr>
<td>SupportedCRIAttributes</td>
<td>3399</td>
</tr>
<tr>
<td>SupportedPolicyQualifiers</td>
<td>2184</td>
</tr>
<tr>
<td>SWIdent</td>
<td>328</td>
</tr>
<tr>
<td>Thumbs</td>
<td>330</td>
</tr>
<tr>
<td>TokenOpaque</td>
<td>1962</td>
</tr>
<tr>
<td>TransactionDetail</td>
<td>1751</td>
</tr>
<tr>
<td>TransactionDetailSeq</td>
<td>1749</td>
</tr>
<tr>
<td>TransactionStatus</td>
<td>1770</td>
</tr>
<tr>
<td>TransDetails</td>
<td>1653</td>
</tr>
<tr>
<td>TransExtensionsIOS</td>
<td>1763</td>
</tr>
<tr>
<td>TransIDs</td>
<td>337</td>
</tr>
<tr>
<td>TransmissionStatus</td>
<td>1676</td>
</tr>
<tr>
<td>TransStain</td>
<td>851</td>
</tr>
<tr>
<td>TripLeg</td>
<td>3316</td>
</tr>
<tr>
<td>TripLegSeq</td>
<td>3314</td>
</tr>
<tr>
<td>TunnelAlg</td>
<td>2338</td>
</tr>
<tr>
<td>tunneling</td>
<td>2328</td>
</tr>
<tr>
<td>TunnelingSyntax</td>
<td>2333</td>
</tr>
<tr>
<td>TYPE-IDENTIFIER</td>
<td>ASN.1</td>
</tr>
<tr>
<td>ub-acctidentification</td>
<td>709</td>
</tr>
<tr>
<td>ub-acqBusinessID</td>
<td>355</td>
</tr>
<tr>
<td>ub-acqCardPhone</td>
<td>1967</td>
</tr>
<tr>
<td>ub-acqCardText</td>
<td>1966</td>
</tr>
<tr>
<td>ub-airportCode</td>
<td>3335</td>
</tr>
<tr>
<td>ub-approvalCode</td>
<td>1968</td>
</tr>
<tr>
<td>ub-AVSData</td>
<td>1969</td>
</tr>
<tr>
<td>ub-BrandID</td>
<td>352</td>
</tr>
<tr>
<td>ub-cardholderMsg</td>
<td>710</td>
</tr>
<tr>
<td>ub-carrierCode</td>
<td>3336</td>
</tr>
<tr>
<td>ub-cityName</td>
<td>2350</td>
</tr>
<tr>
<td>ub-commCode</td>
<td>3337</td>
</tr>
<tr>
<td>ub-common-name</td>
<td>3137</td>
</tr>
<tr>
<td>ub-corpID</td>
<td>3338</td>
</tr>
<tr>
<td>ub-countryName</td>
<td>2349</td>
</tr>
<tr>
<td>ub-description</td>
<td>3339</td>
</tr>
<tr>
<td>ub-eeMessage</td>
<td>711</td>
</tr>
<tr>
<td>ub-fareBasis</td>
<td>3340</td>
</tr>
<tr>
<td>ub-FieldDesc</td>
<td>712</td>
</tr>
<tr>
<td>ub-FieldList</td>
<td>713</td>
</tr>
<tr>
<td>ub-FieldName</td>
<td>714</td>
</tr>
<tr>
<td>ub-FieldValue</td>
<td>715</td>
</tr>
<tr>
<td>ub-hotelFolio</td>
<td>3341</td>
</tr>
<tr>
<td>ub-insuranceType</td>
<td>3342</td>
</tr>
<tr>
<td>ub-items</td>
<td>3343</td>
</tr>
<tr>
<td>Tag</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
</tr>
<tr>
<td>ub-locationID</td>
<td>356 291</td>
</tr>
<tr>
<td>ub-logRefID</td>
<td>1970 1213</td>
</tr>
<tr>
<td>ub-MerchantID</td>
<td>353 294</td>
</tr>
<tr>
<td>ub-merchantName</td>
<td>2351 2316</td>
</tr>
<tr>
<td>ub-merchantOrderNum</td>
<td>1971 1904</td>
</tr>
<tr>
<td>ub-merchantType</td>
<td>1972 1054 3187</td>
</tr>
<tr>
<td>ub-organization-name</td>
<td>3138 3023</td>
</tr>
<tr>
<td>ub-organizational-unit-name</td>
<td>3139 3028</td>
</tr>
<tr>
<td>ub-passName</td>
<td>3344 3304</td>
</tr>
<tr>
<td>ub-paymentID</td>
<td>357 320</td>
</tr>
<tr>
<td>ub-phone</td>
<td>3345 322</td>
</tr>
<tr>
<td>ub-PolicyText</td>
<td>716 482</td>
</tr>
<tr>
<td>ub-postalCode</td>
<td>2352 290 2319</td>
</tr>
<tr>
<td>ub-productCode</td>
<td>3346 3197</td>
</tr>
<tr>
<td>ub-programCode</td>
<td>3347 3269</td>
</tr>
<tr>
<td>ub-Reason</td>
<td>717 476 666</td>
</tr>
<tr>
<td>ub-recurringFrequency</td>
<td>1973 1958</td>
</tr>
<tr>
<td>ub-reference</td>
<td>3348 1929 3178 3184 3185</td>
</tr>
<tr>
<td>ub-rentalNum</td>
<td>3349 3215</td>
</tr>
<tr>
<td>ub-rentalRefNum</td>
<td>3350 3216</td>
</tr>
<tr>
<td>ub-renterName</td>
<td>3351 3211</td>
</tr>
<tr>
<td>ub-RFC1766-language</td>
<td>358 282</td>
</tr>
<tr>
<td>ub-serviceClass</td>
<td>3352 3319</td>
</tr>
<tr>
<td>ub-settlementAccount</td>
<td>1974 1687</td>
</tr>
<tr>
<td>ub-stateProvince</td>
<td>2353 289 2318</td>
</tr>
<tr>
<td>ub-summary</td>
<td>1975 1928</td>
</tr>
<tr>
<td>ub-SWIdent</td>
<td>354 328</td>
</tr>
<tr>
<td>ub-taCode</td>
<td>3353 3309</td>
</tr>
<tr>
<td>ub-taName</td>
<td>3354 3310</td>
</tr>
<tr>
<td>ub-taxID</td>
<td>3355 3183</td>
</tr>
<tr>
<td>ub-taxType</td>
<td>3356 3204</td>
</tr>
<tr>
<td>ub-terminalID</td>
<td>1976 1908</td>
</tr>
<tr>
<td>ub-tenderStatement</td>
<td>2354 2218</td>
</tr>
<tr>
<td>ub-ticketNum</td>
<td>3357 3308</td>
</tr>
<tr>
<td>ub-unitMeasure</td>
<td>3359 3194</td>
</tr>
<tr>
<td>ub-URL</td>
<td>359 346</td>
</tr>
<tr>
<td>ub-validationCode</td>
<td>1977 1225</td>
</tr>
<tr>
<td>ub-vehicleClass</td>
<td>3358 3234</td>
</tr>
<tr>
<td>UniqueIdentifier</td>
<td>2038 2008 2009</td>
</tr>
<tr>
<td>UnsignedBrandCRLIdentifier</td>
<td>197 195</td>
</tr>
<tr>
<td>UnsignedCertificate</td>
<td>2000 2021</td>
</tr>
<tr>
<td>UnsignedCertificateRevocationList</td>
<td>2634 2652</td>
</tr>
<tr>
<td>URL</td>
<td>346 454 455 480 685 686 1122 2219 2220</td>
</tr>
<tr>
<td>us</td>
<td>ASN.1 2958 2971 2974 2982</td>
</tr>
<tr>
<td>userApplications</td>
<td>ASN.1 3092</td>
</tr>
<tr>
<td>UTCTime</td>
<td>ASN.1 2034 2035 2638 2639 2648</td>
</tr>
<tr>
<td>ValidationCode</td>
<td>1225 1173</td>
</tr>
<tr>
<td>Validity</td>
<td>2033 2005</td>
</tr>
<tr>
<td>VisibleString</td>
<td>ASN.1 1 282 320 328 346 402 1215 1225 1904 1908 3131</td>
</tr>
<tr>
<td>XID</td>
<td>348 70 340 851</td>
</tr>
</tbody>
</table>

Version 1.0
Index

---A---
AcctData, 154
AcqCardMsg, 43
Asymmetric Encryption - E, 9
AuthReq, 92
AuthReqPayload, 95
AuthRes, 99
AuthResPayload, 101
AuthRevReq, 107
AuthRevRes, 110
AuthToken, 40

---B---
BatchAdminReq, 141
BatchAdminRes, 144
BatchStatus, 47
BrandCRLIdentifier, 151

---C---
CA (CA types), 150
CapPayload, 115
CapReq, 112
CapRes, 117
CapResPayload, 119
CapRevOrCredReqData, 122
CapRevOrCredResData, 125
CapRevOrCredResPayload, 127
CapRevReq, 129
CapRevRes, 131
CapToken, 44
CardCInitReq, 158
CardCInitRes, 159
CertInqReq, 177
CertInqRes, 177
CertReq, 169
CommercialCardData, 58
CredReq, 132
CredRes, 134
CredRevReq, 135
CredRevRes, 137

---D---
DD, 6
DigestedData - DD, 6

---E---
E, 9
EE (End Entity types), 150
EH, 9
EK, 11
Enc, 12

Encapsulations with External, Encrypted Baggage - EncB,
EncB, 14
EncBX, 14
EncK, 12
EncX, 13
Entity Symbols, 5
Error Message, 30
EX, 10
EXH, 11
Extra Encapsulation with Signature - EncX, 13
Extra Encryption - EX, 10
Extra Encryption with Integrity - EXH, 11

---H---
H, 6
Hash - H, 6
HMAC, 7

---I---
IDData, 148
InqReq, 90
InqRes, 91
InstallRecurData, 42
IntegrityEncryption - EH, 9

---K---
Keyed hash mechanism - HMAC, 7

---L---
L, 7
Linkage - L, 7

---M---
MarketAutoCap, 62
MarketHotelCap, 66
MarketTransportCap, 69
Me-AcCInitReq, 160
Me-AcCInitRes, 162
MessageWrapper, 25

---N---
Notation, 3

---O---
OAEP Block Format, 15
OIData, 81
—P—
PANData, 45
PANData0, 153
PANToken, 46
PCertReq, 139
PCertRes, 140
PI, 35
PI Variants, 35
PIHead, 37
PInitReq, 72
PInitRes, 74
PReq, 76
PReqDualSigned, 77
PReqUnsigned, 79
PRes, 84
PResPayload, 86

—R—
RegFormOrReferral, 155

—S—
S, 8
SaleDetail, 53
Signature Only - SO, 8
Signed Message - S, 8
Simple Encapsulations with Signature - Enc, EncK, 12
SO, 8
Symmetric Encryption with Provided Key Data - EK, 11

—T—
TransactionDetail, 50
TransIDs, 33