SET Secure Electronic Transaction Specification

Book 3: Formal Protocol Definition

Version 1.0
May 31, 1997

PC

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.

Answers to Frequently Asked Questions about Today’s Cryptography, Paul Fahn, RSA Laboratories, 1993. (http://www.rsa.com/rsalabs/faq/)

Applied Cryptography, Second Edition, Bruce Schneier, John Wiley & Sons, Inc., 1996

“Asymmetric Encryption: Evolution and Enhancements,” Don B. Johnson and Stephen M. Matyas, *CryptoBytes,* volume 2, number 1, Spring 1996

BSAFE 2.1™, RSA Data Security, Inc., 1994. (http://www.rsa.com/rsa/prodspec/bsafe/rsa_bsaf.htm)

Data Encryption Standard, Federal Information Processing Standards Publication 46, 1977.

“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

The Internet for Everyone: A Guide for Users and Providers, Richard W. Wiggins, McGraw-Hill, Inc., 1995.

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

An Overview of the PKCS Standards, Burton S. Kaliski, Jr., RSA Laboratories, 1993. (http://www.rsa.com/pub/pkcs/doc or http://www.rsa.com/pub/pkcs/ps)

Public-Key Cryptography Standards (PKCS), RSA Data Security, Inc., Version 1.5, revised Nov. 1, 1993.

Extensions and Revisions to PKCS #7, RSA Data Security, Inc., May 13, 1997.

ITU Rec. X.509 (1993) | ISO/IEC 9594-8: 1995, including Draft Amendment 1: Certificate Extensions (Version 3 certificate).

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

PANData0 153
AcctData 154
RegFormOrReferral 155
Chapter 7 Certificate Management Messages 157
Overview 157
Certificate Initialization Pair - Cardholder 158
Certificate Initialization Pair - Merchant or Payment Gateway 160
Registration Form Pair - Cardholder Only 165
Certificate Request Pair 169
Certificate Inquiry Pair 177
Part II ASN.1 Code 179
Cross reference 239
Index 253

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 42
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

Table 53: AuthRevRes 110
Table 54: CapReq 112
Table 55: CapPayload 115
Table 56: CapRes 117
Table 57: CapResPayload 119
Table 58: CapRevOrCredReqData 122
Table 59: CapRevOrCredResData 125
Table 60: CapRevOrCredResPayload 127
Table 61: CapRevReq 129
Table 62: CapRevRes 131
Table 63: CredReq 132
Table 64: CredRes 134
Table 65: CredRevReq 135
Table 66: CredRevRes 137
Table 67: PCertReq 139
Table 68: PCertRes 140
Table 69: BatchAdminReq 141
Table 70: BatchAdminRes 144
Table 71: IDData 148
Table 72: RequestType 149
Table 73: EE (End Entity types) 150
Table 74: CA (CA types) 150
Table 75: BrandCRLIdentifier 151
Table 76: PANData0 153
Table 77: AcctData 154
Table 78: RegFormOrReferral 155
Table 79: CardCInitReq 158
Table 80: CardCInitRes 159
Table 81: Me-AqCInitReq 160
Table 82: Me-AqCInitRes 162
Table 83: RegFormReq 165
Table 84: RegFormRes 167
Table 85: CertReq 169
Table 86: CertInqReq 177
Table 87: CertInqRes 177

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:

Chapter	Title	Page
1	Cryptography	4
2	Message Encapsulation	24
3	Payment Message Components	32
4	Payment Messages	71
5	Payment Gateway Certificate Request and Batch Administration	138
6	Certificate Management Payload Components	147
7	Certificate Management Messages	157

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.

Concept	**Notation**	**Definition**	
Tuple	**{A, B, C}**	A grouping of zero or more data elements. This notation means "the tuple containing ***A***, ***B***, and ***C***," which may, themselves, be tuples.	
Component	**T = {A, B, C}**	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.	
Ordered concatenation	**A | B | C**	This notation means that an explicit, *ordered concatenation* of items ***A***, ***B***, and ***C*** is needed.	
Optional	**[A]**	This notation means that item ***A*** is *optional*.	Any other nesting of these brackets is permissible.
Selection	**<A, B, C>**	This notation means that exactly one of ***A***, ***B***, and ***C*** must appear. This is a *selection* notation.	
Optional selection	**[<A, B, C>]**	This notation means that the *selection* is *optional*; that is, that either nothing or exactly one of ***A***, ***B***, and ***C*** may appear.	
Multiple instances	**{A +}**	This notation means a tuple containing *one or more instances* of **A.** (Order may not be significant; refer to the specific description for details.)	
	{A *}	This notation means a tuple containing *zero or more instances* of ***A***.	
	{[A] +}	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).	
Exclusive-or	**A ⊕ B**	This symbol denotes a bit-wise *exclusive-or* (XOR) operation.	

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:

Topic	Page
Entities	5
Hashing and Hash-based operators	6
Signature Primitives	8
Encryption Primitives	9
Encapsulation Operations	12
Optimal Asymmetric Encryption Padding (OAEP)	15

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.

```
2948 CA ::= ENTITY-IDENTIFIER  -- Certifying Authority
2944 C  ::= ENTITY-IDENTIFIER  -- Cardholder
2945 M  ::= ENTITY-IDENTIFIER  -- Merchant
2946 P  ::= ENTITY-IDENTIFIER  -- Payment Gateway
2949 P1 ::= ENTITY-IDENTIFIER  -- Gateway One
2950 P2 ::= ENTITY-IDENTIFIER  -- Gateway Two
```

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.

r*, *s	In this chapter: • ***r*** represents a receiver, identified through an *encryption* or *key-exchange* certificate; and • ***s*** represents a sender, identified through a *signature* certificate. 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 ***t1*** and ***t2*** such that **H(*t1*) = H(*t2*)**.

Table 3: Hash - H

```
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 })
```

DigestedData

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

Table 4: DigestedData - DD

```
DD { ToBeHashed } ::= DetachedDigest
   (CONSTRAINED BY { -- digest of the DER representation, including --
                     -- the tag and length octets, of -- ToBeHashed })
```

Continued on next page

Hashing and Hash-based operators, continued

Linkage

L(*t1*, *t2*)	Shorthand for **{*t1*, DD(*t2*)}**, an augmentation of ***t1*** to provide linkage *from* ***t1*** *to* ***t2***. 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

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

Keyed hash mechanism

HMAC(*t*, *k*)	A derivation of HMAC-MD5 using the SHA-1 algorithm. **HMAC(*t*, *k*) = H((*k* ⊕ opad) | H((*k* ⊕ ipad) | *t*))** where • **ipad** is the byte 0x36 repeated 64 times, and • **opad** is the byte 0x5c repeated 64 times. Note, ⊕ denotes XOR.

Table 6: Keyed hash mechanism - HMAC

```
HMAC { ToBeHashed, Key } ::= Digest
  (CONSTRAINED BY { -- HMAC keyed digest of -- ToBeHashed,
                                                  -- 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

```
SO { SIGNER, ToBeSigned } ::= SignedData             -- Detached content
   (CONSTRAINED BY { SIGNER, -- signs -- ToBeSigned })
   (WITH COMPONENTS { ..., contentInfo
       (WITH COMPONENTS{
                  ..., content ABSENT }) } ^
    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

```
S { SIGNER, ToBeSigned } ::= SignedData
   (CONSTRAINED BY { SIGNER, -- signs -- ToBeSigned })
   (WITH COMPONENTS { ..., contentInfo
        (WITH COMPONENTS {
                  ..., content PRESENT }) } ^
    WITH COMPONENTS { ..., signerInfos (SIZE(1..2)) })
```

Encryption Primitives

Asymmetric encryption

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

E(r, t)	*Asymmetric Encryption* to entity ***r*** of tuple ***t***, corresponding to the standard PKCS #7 *EnvelopedData*.

Step	Action
1	Encrypt ***t*** with a DES key, **k**.
2	Insert **k** in a PKCS #7 envelope for entity ***r*** under OAEP.

Table 9: Asymmetric Encryption - E

```
E { RECIPIENT, ToBeEnveloped } ::= EnvelopedData
   (CONSTRAINED BY { ToBeEnveloped, -- is encrypted, and the --
                      -- session key is encrypted using the --
                      -- public key of -- RECIPIENT } )
   (WITH COMPONENTS {..., encryptedContentInfo
                 (WITH COMPONENTS { ..., encryptedContent PRESENT }) } ^
    WITH COMPONENTS { ..., recipientInfos (SIZE(1)) })
```

Integrity encryption

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

EH(r, t)	*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.

Table 10: Integrity Encryption - EH

```
EH { RECIPIENT, ToBeEnveloped } ::= E {
   RECIPIENT,
   ToBeEnveloped
} (CONSTRAINED BY { -- H(ToBeEnveloped) included in the OAEP block -- })
```

Continued on next page

Encryption Primitives, continued

Extra encryption

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

EX(*r*, *t*, *p*)	***r*** is the receiver, and ***t*** and ***p*** are the parts of a two-part message: • ***t*** is the tuple to be linked to ***p*** and subjected to symmetric encryption. • ***p*** is a *parameter* subject to "extra" processing. 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:

Step	Action
1	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.
2	Let **m = L(*t*, *p*)**, that is, ***t*** linked to ***p***.
3	Encrypt **m** with a DES key **k** and let **OAEP({k, *p*})** be the RSA envelope for entity ***r***.

Table 11: Extra Encryption - EX

```
EX { RECIPIENT, ToBeEnveloped, Parameter } ::= E {
   RECIPIENT,
   L { ToBeEnveloped, Parameter }
}(CONSTRAINED BY { Parameter -- data is included in the OAEP block -- })
```

Continued on next page

Encryption Primitives, continued

Extra encryption with integrity

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

EXH(*r*, *t*, *p*)	Like **EX**, except that • a hash of ***t*** is included in the PKCS #7 envelope and • the processing software shall check the hash of ***t***, as with **EH**.

Table 12: Extra Encryption with Integrity - EXH

```
EXH { RECIPIENT, ToBeEnveloped, Parameter } ::= EX {
   RECIPIENT,
   ToBeEnveloped,
   Parameter
} (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

```
EK { KeyData, ToBeEnveloped } ::= EncryptedData
   (CONSTRAINED BY { ToBeEnveloped, -- encrypted with -- KeyData } )
   (WITH COMPONENTS { ..., encryptedContentInfo
                  (WITH COMPONENTS { ..., encryptedContent PRESENT}) })
```

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.

Enc(*s*, *r*, *t*)	*Simple Encapsulation with Signature* Shorthand for **E(*r,* S(*s*, *t*))**. Corresponds to an instance of PKCS #7 *SignedData* encapsulated in *EnvelopedData*.
EncK(*kd*, *s*, *t*)	*Simple Encapsulation with Signature and Provided Key Data (algorithm and key)* **EncK(*kd*, *s*, *t*) = EK(*kd*, S(*s*, *t*))**.

Table 14: Simple Encapsulations with Signature - Enc, EncK

```
Enc { SIGNER, RECIPIENT, T } ::= E {
   RECIPIENT,
   S { SIGNER, T }
}
```

```
EncK { KeyData, SIGNER, T } ::= EK {
   KeyData,
   S { SIGNER, T }
}
```

Continued on next page

Encapsulation Operations, continued

Extra encapsulation with signature

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.

***p* is processed in two distinct ways:**	**...which require two different formats for *p*:**	**...and are indicated as:**
It is incorporated into the OAEP data.	See "Extra encryption" on page 10.	**OAEP(*p*)**
It is included in the **SO** signature described below.	The **SO** signature is computed over the DER-encoded form of ***p***.	**DER(*p*)**

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**:

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

Table 15: Extra Encapsulation with Signature - EncX

```
EncX { SIGNER, RECIPIENT, T, Parameter } ::= E {
   RECIPIENT,
   SEQUENCE {
       t  T,
       s  SO { SIGNER, SEQUENCE { t  T, p  Parameter } }
    }
} (CONSTRAINED BY { Parameter -- data, which shall contain a fresh --
                     -- nonce 'n', is included in the OAEP block.  -- } )
```

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.

Step	Action
1	Link the baggage to the main message.
2	Sign and encrypt the linked object.
3	Append the baggage to the end of the encrypted message.
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 16: Encapsulations with External, Encrypted Baggage - EncB, EncBX

```
EncB { SIGNER, RECIPIENT, T, Baggage } ::= SEQUENCE {
   enc      Enc { SIGNER, RECIPIENT, L { T, Baggage } },
   baggage  Baggage
}

EncBX { SIGNER, RECIPIENT, T, Baggage, Parameter } ::= SEQUENCE {
   encX     EncX { SIGNER, RECIPIENT, L { T, Baggage }, Parameter },
   baggage  Baggage
}
```

Optimal Asymmetric Encryption Padding (OAEP)

OAEP block format

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.

Data:	**...is carried in PKCS #7 field:**
OAEP block (including "extra" encryption)	**RecipientInfo.encryptedKey**
Symmetrically encrypted data	**EnvelopedData.encryptedContentInfo**

Item	Description	Length
R	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.	128
I	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.	1
PDB	The Padded Data Block, the concatenation of two parts: **A** and **B**. **PDB = A** | **B**	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)** ⊕ **DB**	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) | ... **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)

Item	Description	Length
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)

Item	Description	Length
ADB	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:	102

Value of BC	Encryption primitive	Fields in ADB
0 (zero)	**E**	**DEK**
> 0 and < 80 hex	**EX**	**DEK** | **X**
80 hex	**EH**	**DEK** | **HD**
> 81 hex	**EXH**	**DEK** | **HD** | **X**

Fill any unused space with zero bytes.

Item	Description	Length
DEK	An 8-byte DES encryption key stored at the start of **ADB**. This key is used to encrypt **D**.	8
HD	A 20-byte SHA-1 hash of the data prior to encryption: **H(D)**.	20
D	The data that will be symmetrically encrypted under the DES key **DEK**.	varies
X	“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.	varies
B	**B** is the **XOR** of **E-Salt** with the **H2** hash of **A**: $\mathbf{B} = \mathbf{E\text{-}Salt} \oplus \mathbf{H2}(\mathbf{A})$ **B** is the same length as **E-Salt**.	16
H2(t)	**H2** returns the trailing 16 bytes of the SHA-1 hash of its argument, **t**.	16

Table 17: OAEP Block Format, continued

Continued on next page

Optimal Asymmetric Encryption Padding (OAEP), continued

Field lengths

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.

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 **EncX**, **EX**, or **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.

Extra Encryption Type	Space Available with 128 Byte RSA Block
EncX	94 bytes
EX	94 bytes
EXH	74 bytes

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:

ASN.1 Type	DB Encoding
VisibleString	ASCII string, first character in lowest-numbered position, padded with blanks (0x20).
NumericString	ASCII string, first character in lowest-numbered position, padded with blanks (0x20).
OCTET STRING	Binary byte string in lowest-numbered position, padded with bytes of zero (0x00).

Continued on next page

Optimal Asymmetric Encryption Padding (OAEP), continued

DB fields

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

Field Name	Length	Format
I	1	OCTET STRING
A	111	OCTET STRING
B	16	OCTET STRING
BT	1	OCTET STRING
BC	1	OCTET STRING
V	7	OCTET STRING
DEK	8	OCTET STRING
HD	20	OCTET STRING
X	varies	depends upon content

PANData

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

Field Name	Length	Format
PAN	19	Numeric String
CardExpiry	6	NumericString - YYYYMM
PANSecret	20	OCTET STRING
EXNonce	20	OCTET STRING

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

```
PANData ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   panSecret   Secret,
   exNonce     Nonce
}

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

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

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:

Field Name	Length	Format
PAN	19	Numeric String
CardExpiry	6	NumericString - YYYYMM
CardSecret	20	OCTET STRING
EXNonce	20	OCTET STRING

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

```
PANData0 ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   cardSecret  Secret,
   exNonce     Nonce
}

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

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

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:

Field Name	Length	Format
PAN	19	Numeric String
CardExpiry	6	NumericString - YYYYMM
EXNonce	20	OCTET STRING

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.

Field Name	Length	Format
PAN	19	Numeric String
EXNonce	20	OCTET STRING

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

```
552 PANOnly ::= SEQUENCE {
553    pan      PAN,
554    exNonce  Nonce
555 }

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

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

AcctData

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

Field Name	Length	Format
AcctIdentification	74	VisibleString
EXNonce	20	OCTET STRING

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

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

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

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

Chapter 2
Message Encapsulation

Organization

This chapter describes:

- MessageWrapper
- Error Message

MessageWrapper

MessageWrapper

MessageWrapper	**{MessageHeader, Message, [MWExtensions]}**
MessageHeader	**{Version, Revision, Date, [MessageIDs], [RRPID], SWIdent}**
Message	**<** **PInitReq, PInitRes,** **PReq, PRes,** **InqReq, InqRes,** **AuthReq, AuthRes,** **AuthRevReq, AuthRevRes,** **CapReq, CapRes,** **CapRevReq, CapRevRes,** **CredReq, CredRes,** **CredRevReq, CredRevRes,** **PCertReq, PCertRes,** **BatchAdminReq, BatchAdminRes,** **CardCInitReq, CardCInitRes,** **Me-AqCInitReq, Me-AqCInitRes,** **RegFormReq, RegFormRes,** **CertReq, CertRes,** **CertInqReq, CertInqRes,** **Error** **>**
MWExtensions	*Appropriate where:* • *the data in the extension is general purpose information about SET messages, or* • *the contents of the message are encrypted and the extension contains non-financial data that does not require confidentiality.* *Note: The message wrapper is not encrypted so this extension must not contain confidential information.*

Table 18: MessageWrapper

Continued on next page

MessageWrapper, continued

MessageWrapper (continued)

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

Table 18: MessageWrapper, continued

Continued on next page

MessageWrapper, continued

MessageWrapper (continued)

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

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
}
```

Continued on next page

MessageWrapper, continued

MessageWrapper (continued)

```
Message ::= CHOICE {

   purchaseInitRequest          [ 0] EXPLICIT PInitReq,
   purchaseInitResponse         [ 1] EXPLICIT PInitRes,

   purchaseRequest              [ 2] EXPLICIT PReq,
   purchaseResponse             [ 3] EXPLICIT PRes,

   inquiryRequest               [ 4] EXPLICIT InqReq,
   inquiryResponse              [ 5] EXPLICIT InqRes,

   authorizationRequest         [ 6] EXPLICIT AuthReq,
   authorizationResponse        [ 7] EXPLICIT AuthRes,

   authReversalRequest          [ 8] EXPLICIT AuthRevReq,
   authReversalResponse         [ 9] EXPLICIT AuthRevRes,

   captureRequest               [10] EXPLICIT CapReq,
   captureResponse              [11] EXPLICIT CapRes,

   captureReversalRequest       [12] EXPLICIT CapRevReq,
   captureReversalResponse      [13] EXPLICIT CapRevRes,

   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
}
```

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

Error	**< SignedError, UnsignedError >**
SignedError	**S(EE, ErrorTBS)**
UnsignedError	**ErrorTBS** *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).*
ErrorTBS	**{ErrorCode, ErrorNonce, [ErrorOID], [ErrorThumb], ErrorMsg}**
ErrorCode	*Enumerated code.*
ErrorNonce	*A nonce to ensure the signature is generated over unpredictable data.*
ErrorOID	*The object identifier of an object (extension, content type, attribute, etc.) that caused the error.*
ErrorThumb	*The thumbprint of the certificate, CRL or BrandCRLIdentifier that caused the error.*
ErrorMsg	**<MessageHeader, BadWrapper>**
MessageHeader	*The message header of the message that produced the error.*
BadWrapper	*The message wrapper of the message that produced the error (up to 20,000 bytes).*

Table 19: Error Message

```
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
 }
```

Continued on next page

Error Message, continued

Error message (continued)

```
164 ErrorCode ::= ENUMERATED {
165    unspecifiedFailure      (1),
166    messageNotSupported     (2),
167    decodingFailure         (3),
168    invalidCertificate      (4),
169    expiredCertificate      (5),
170    revokedCertificate      (6),
171    missingCertificate      (7),
172    signatureFailure        (8),
173    badMessageHeader        (9),
174    wrapperMsgMismatch     (10),
175    versionTooOld          (11),
176    versionTooNew          (12),
177    unrecognizedExtension  (13),
178    messageTooBig          (14),
179    signatureRequired      (15),
180    messageTooOld          (16),
181    messageTooNew          (17),
182    thumbsMismatch         (18),
183    unknownRRPID           (19),
184    unknownXID             (20),
185    unknownLID             (21),
186    challengeMismatch      (22)
187  }

159 ErrorMsg ::= CHOICE {                              -- Either the
160    messageHeader  [0] EXPLICIT MessageHeader,       -- MessageHeader or a
161    badWrapper     [1] OCTET STRING (SIZE(1..20000)) -- copy of the message
162 }

 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 }
```

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:

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

TransIDs

TransIDs

TransIDs	**{LID-C, [LID-M], XID, PReqDate, [PaySysID], Language }**
LID-C	*Local ID; convenience label generated by and for Cardholder system. This field has the same value as in the MessageWrapper; see page 25.*
LID-M	*Local ID; convenience label generated by and for Merchant system. This field has the same value as in the MessageWrapper; see page 25.*
XID	*Globally unique ID. This field has the same value as in the MessageWrapper; see page 25.*
PReqDate	*Date of purchase request; generated by Merchant in* **PInitRes** *or by Cardholder in* **PReq***.*
PaySysID	*Used by some payment card brands to label transaction from time of authorization onward*
Language	*Cardholder's natural language*

Table 20: TransIDs

```
TransIDs ::= SEQUENCE {
   lid-C      LocalID,
   lid-M      [0] LocalID  OPTIONAL,
   xid        XID,
   pReqDate   Date,
   paySysID   [1] PaySysID  OPTIONAL,
   language   Language          -- Cardholder requested session language
}

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

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

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

RRTags

RRTags

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

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)

PI (Payment Instruction)

There are three alternative forms of PI:

PI form	Created by...	For...
PIUnsigned	Cardholder	Sending **PReqUnsigned** message
PIDualSigned	Cardholder	Sending **PReqDualSigned** message
AuthToken	Payment Gateway	Merchants to redeem in subsequent **AuthReqs**

Table 22: PI Variants

PI	**< PIUnsigned, PIDualSigned, AuthToken >** *Cardholder creates* **PIUnsigned** *or* **PIDualSigned**. *Payment gateway creates* **AuthToken** *to support split shipments or installment/recurring payments.* *Merchant shall retain the* **PI** *for later incorporation into* **AuthReq**.
PIUnsigned	**EXH(P, PI-OILink, PANToken)** *See page 46 for* **PANToken**.
PIDualSigned	**{PISignature, EX(P, PI-OILink, PANData)}** *See page 45 for* **PANData**.
AuthToken	*See page 40.*
PI-OILink	**L(PIHead, OIData)** *See page 37 for* **PIHead**. *See page 81 for* **OIData**.
PISignature	**SO(C, PI-TBS)**
PI-TBS	**{HPIData, HOIData}**
HPIData	**DD(PIData)**
HOIData	**DD(OIData)** *See page 81 for* **OIData**.
PIData	**{PIHead, PANData}** *See page 37 for* **PIHead**. *See page 45 for* **PANData**.

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

PIHead	**{TransIDs, Inputs, MerchantID, [InstallRecurData], TransStain, SWIdent, [AcqBackKeyData], [PIExtensions]}**
TransIDs	*See page 33.*
Inputs	**{HOD, PurchAmt}**
MerchantID	*Copied from Merchant signature certificate*
InstallRecurData	*See page 42.*
TransStain	**HMAC(XID, CardSecret)**
SWIdent	*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.*
AcqBackKeyData	**{AcqBackAlg, AcqBackKey}**
PIExtensions	*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.*

Table 24: PIHead

Continued on next page

PI (Payment Instruction), continued

PIHead (continued)

HOD	*The same value as placed in* **OIData***. See “OIData” on page 81*
PurchAmt	*The amount of the transaction as specified by the Cardholder*
XID	*Copied from* **TransIDs***; see page 33*
CardSecret	*See* “PANData0*” on page 21.*
AcqBackAlg	*Selected from Encryption IDs in Payment Gateway certificate.*
AcqBackKey	*Key for* **AcqCardMsg** *of an appropriate length for* **AcqBackAlg**

Table 24: PIHead, continued

```
PIHead ::= SEQUENCE {
   transIDs          TransIDs,
   inputs            Inputs,
   merchantID        MerchantID,
   installRecurData  [0] InstallRecurData  OPTIONAL,
   transStain        TransStain,
   swIdent           SWIdent,
   acqBackKeyData    [1] EXPLICIT BackKeyData  OPTIONAL,
   piExtensions      [2] MsgExtensions {{PIExtensionsIOS}} OPTIONAL
}
```

```
TransIDs ::= SEQUENCE {
   lid-C      LocalID,
   lid-M      [0] LocalID  OPTIONAL,
   xid        XID,
   pReqDate   Date,
   paySysID   [1] PaySysID  OPTIONAL,
   language   Language          -- Cardholder requested session language
}
```

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.

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

Table 25: AuthToken

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

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)

```
PANToken ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   exNonce     Nonce
}

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

MerchantID ::= SETString { ub-MerchantID }

InstallRecurData ::= SEQUENCE {
   installRecurInd  InstallRecurInd,
   irExtensions     [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL
}
```

InstallRecurData

InstallRecurData Specifies information about installment or recurring payments.

InstallRecurData	**{InstallRecurInd, [IRExtensions]}**
InstallRecurInd	**< InstallTotalTrans, Recurring >**
IRExtensions	*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.*
InstallTotalTrans	*Cardholder specifies a maximum number of permitted Authorizations for installment payments.*
Recurring	**{RecurringFrequency, RecurringExpiry}**
RecurringFrequency	*The minimum number of days between Authorizations (a frequency of monthly is indicated by a value of 28), and…*
RecurringExpiry	*a final date, after which no further Authorizations are permitted.*

Table 26: InstallRecurData

```
InstallRecurData ::= SEQUENCE {
   installRecurInd  InstallRecurInd,
   irExtensions     [0] MsgExtensions {{IRExtensionsIOS}} OPTIONAL
}

InstallRecurInd ::= CHOICE {
   installTotalTrans  [0] INTEGER (2..MAX),
   recurring          [1] Recurring
}

Recurring ::= SEQUENCE {
   recurringFrequency  INTEGER (1..ub-recurringFrequency),
   recurringExpiry     Date
}
```

AcqCardMsg

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.

AcqCardMsg	**EncK(AcqBackKeyData, P, AcqCardCodeMsg)** **AcqBackKeyData** *is supplied by the Cardholder in the* **PI**. *The encrypted message is destined to the Cardholder.*
AcqBackKeyData	*Copied from* **PIHead.AcqBackKeyData***; see page 37.*
AcqCardCodeMsg	**{AcqCardCode, AcqCardMsgData}**
AcqCardCode	*Enumerated code*
AcqCardMsgData	**{[AcqCardText], [AcqCardURL], [AcqCardPhone]}**
AcqCardText	*Textual message to be displayed to Cardholder*
AcqCardURL	*URL referencing HTML message to be displayed to Cardholder*
AcqCardPhone	*Phone number to be presented to the Cardholder*

Table 27: AcqCardMsg

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

AcqCardCodeMsg ::= SEQUENCE {
   acqCardCode     AcqCardCode,
   acqCardMsgData  AcqCardMsgData
}

AcqCardCode ::= ENUMERATED {
   messageOfDay         (0),
   accountInfo          (1),
   callCustomerService  (2)
}

AcqCardMsgData ::= SEQUENCE {
   acqCardText   [0] EXPLICIT SETString { ub-acqCardText }  OPTIONAL,
   acqCardURL    [1] URL  OPTIONAL,
   acqCardPhone  [2] EXPLICIT SETString { ub-acqCardPhone }  OPTIONAL
}
```

CapToken

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

CapToken	**< Enc(P1, P2, CapTokenData), EncX(P1, P2, CapTokenData, PANToken), {} >** **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.*
CapTokenData	**{AuthRRPID, AuthAmt, TokenOpaque}**
PANToken	*See page 46.*
AuthRRPID	*The RRPID that appeared in the corresponding* **AuthReq** *or* **AuthRevReq**
AuthAmt	*Actual amount authorized, which may differ from Cardholder's* **PurchAmt**
TokenOpaque	*Opaque data defined by the generating Payment Gateway*

Table 28: CapToken

```
CapToken ::= CHOICE {
   encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
   enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
   null  [2] EXPLICIT NULL
}

CapTokenData ::= SEQUENCE {
   authRRPID    RRPID,
   authAmt      CurrencyAmount,
   tokenOpaque  TokenOpaque
}

PANToken ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   exNonce     Nonce
}

TokenOpaque ::= TYPE-IDENTIFIER.&Type       -- Gateway-defined data
```

PANData

PANData

PANData	**{PAN, CardExpiry, PANSecret, EXNonce}** *Always in the extra (OAEP) slot of an encapsulation operator*
PAN	*Primary Account Number; typically, the account number on the card*
CardExpiry	*Expiration date on the card*
PANSecret	*Secret value shared among Cardholder, Payment Gateway, and Cardholder CA; prevents guessing attacks on* **PAN** *in the Cardholder certificate.*
EXNonce	*A fresh nonce to foil dictionary attacks on* **PANData**

Table 29: PANData

```
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

PANToken

PANToken	**{PAN, CardExpiry, EXNonce}** *Always in the extra (OAEP) slot of an encapsulation operator*
PAN	*Primary Account Number; typically, the account number on the card*
CardExpiry	*Expiration date on the card*
EXNonce	*A fresh nonce to foil dictionary attacks on* **PANToken**

Table 30: PANToken

```
PANToken ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   exNonce     Nonce
}
```

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

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

BatchStatus

BatchStatus

BatchStatus	**{OpenDateTime, [ClosedWhen], BatchDetails, [BatchExtensions]}**
OpenDateTime	*The date and time the batch was opened*
ClosedWhen	**{CloseStatus, CloseDateTime}**
BatchDetails	**{BatchTotals, [BrandBatchDetailsSeq]}**
BatchExtensions	*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.*
CloseStatus	*Enumerated code indicating status of batch close*
CloseDateTime	*The date and time the batch was closed*
BatchTotals	**{TransactionCountCredit, TransactionTotalAmtCredit, TransactionCountDebit, TransactionTotalAmtDebit, [BatchTotalExtensions]}**
BrandBatchDetailsSeq	**{BrandBatchDetails +}**
TransactionCountCredit	*The number of transactions that resulted in a credit to the Merchant's account*
TransactionTotalAmtCredit	*The total amount credited to the Merchant's account*
TransactionCountDebit	*The number of transactions that resulted in a debit to the Merchant's account*
TransactionTotalAmtDebit	*The total amount debited from the Merchant's account*

Table 31: BatchStatus

Continued on next page

BatchStatus, continued

BatchStatus (continued)

BatchTotalExtensions	*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.*
BrandBatchDetails	**{BrandID, BatchTotals}**
BrandID	*Payment card brand (without product type)*

Table 31: BatchStatus, continued

```
BatchStatus ::= SEQUENCE {
   openDateTime     Date,
   closedWhen       [0] ClosedWhen  OPTIONAL,
   batchDetails     BatchDetails,
   batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL
}

ClosedWhen ::= SEQUENCE {
   closeStatus    CloseStatus,
   closeDateTime  Date
}

BatchDetails ::= SEQUENCE {
   batchTotals           BatchTotals,
   brandBatchDetailsSeq  BrandBatchDetailsSeq  OPTIONAL
}

CloseStatus ::= ENUMERATED {
   closedbyMerchant  (0),
   closedbyAcquirer  (1)
}

BatchTotals ::= SEQUENCE {
   transactionCountCredit     INTEGER (0..MAX),
   transactionTotalAmtCredit  CurrencyAmount,
   transactionCountDebit      INTEGER (0..MAX),
   transactionTotalAmtDebit   CurrencyAmount,
   batchTotalExtensions       [0] MsgExtensions {{BTExtensionsIOS}} OPTIONAL
}
```

Continued on next page

BatchStatus, continued

BatchStatus (continued)

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

BrandBatchDetails ::= SEQUENCE {
   brandID      BrandID,
   batchTotals  BatchTotals
}

BrandID ::= SETString { ub-BrandID }
```

Continued on next page

BatchStatus, continued

TransactionDetail

TransactionDetail	**{TransIDs, AuthRRPID, BrandID, BatchSequenceNum, [ReimbursementID], TransactionAmt, TransactionAmtType, [TransactionStatus], [TransExtensions]}**
TransIDs	*The transaction identifiers from the authorization/capture processing of the item*
AuthRRPID	*The RRPID that appeared in the corresponding* **AuthReq** *or* **AuthRevReq**
BrandID	*Payment card brand (without product type)*
BatchSequenceNum	*The sequence number of this item within the batch*
ReimbursementID	*Enumerated code indicating the type of reimbursement for the item*
TransactionAmt	*The amount for the item of the type indicated by* **TransactionAmtType**. *The amount is always specified as a positive value.*
TransactionAmtType	*Enumerated code indicating the type of amount (credit or debit)*
TransactionStatus	*Enumerated code indicating the result of passing the transaction to the next upstream system.*
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)

```
TransactionDetail ::= SEQUENCE {
   transIDs            TransIDs,
   authRRPID           RRPID,
   brandID             BrandID,
   batchSequenceNum    BatchSequenceNum,
   reimbursementID     ReimbursementID  OPTIONAL,
   transactionAmt      CurrencyAmount,
   transactionAmtType  AmountType,
   transactionStatus   [0] TransactionStatus  OPTIONAL,
   transExtensions     [1] MsgExtensions {{TransExtensionsIOS}} OPTIONAL
}

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

BrandID ::= SETString { ub-BrandID }

BatchSequenceNum ::= INTEGER (1..MAX)

ReimbursementID ::= ENUMERATED {
   unspecified     (0),
   standard        (1),
   keyEntered      (2),
   electronic      (3),
   additionalData  (4),
   enhancedData    (5),
   marketSpecific  (6)
}

TransactionStatus ::= ENUMERATED {
   success            (0),
   unspecifiedFailure  (1)
}
```

Location

Location

Location	**{CountryCode, [City], [StateProvince], [PostalCode], [LocationID]}**
CountryCode	*The ISO 3166 country code for the location.*
City	*The city name of the location.*
StateProvince	*The name or abbreviation of the state or province.*
PostalCode	*The postal code of the location.*
LocationID	*An identifier that the Merchant uses to specify one of its locations*

Location (continued)

```
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
}

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

SaleDetail

SaleDetail

SaleDetail	**{[BatchID], [BatchSequenceNum], [PayRecurInd], [MerOrderNum], [AuthCharInd], [MarketSpecSaleData], [CommercialCardData], [OrderSummary], [CustomerReferenceNumber], [CustomerServicePhone], OKtoPrintPhoneInd, [SaleExtensions]}** *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.*
BatchID	*Identification of the settlement batch for merchant-acquirer accounting*
BatchSequenceNum	*The sequence number of this item within the batch*
PayRecurInd	*Enumerated transaction type*
MerOrderNum	*Merchant order number*
AuthCharInd	*Copied from* **AuthResPayload***; see page 101*
MarketSpecSaleData	**{[MarketSpecDataID], [MarketSpecCapData]}**
CommercialCardData	*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.*
OrderSummary	*A summary description of the order.*
CustomerReferenceNumber	*A reference number assigned to the order by the Cardholder.*
CustomerServicePhone	*The Merchant's customer service telephone number*
OKtoPrintPhoneInd	*A Boolean value indicating if the Issuer may print the customer service telephone number on the Cardholder's statement.*

Table 33: SaleDetail

Continued on next page

SaleDetail, continued

SaleDetail (continued)

SaleExtensions	*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.*
MarketSpecDataID	*Copied from* **AuthResPayload***; see page 101*
MarketSpecCapData	**< MarketAutoCap, MarketHotelCap, MarketTransportCap >** *Market-specific capture data*
MarketAutoCap	*Automobile rental charge description. See page 62.*
MarketHotelCap	*Hotel charge description. See page 66.*
MarketTransportCap	*Passenger transport data. See page 69.*

Table 33: SaleDetail, continued

```
SaleDetail ::= SEQUENCE {
   batchID                  [ 0] BatchID  OPTIONAL,
   batchSequenceNum         [ 1] BatchSequenceNum  OPTIONAL,
   payRecurInd              [ 2] PayRecurInd  OPTIONAL,
   merOrderNum              [ 3] MerOrderNum  OPTIONAL,
   authCharInd              [ 4] AuthCharInd  OPTIONAL,
   marketSpecSaleData       [ 5] MarketSpecSaleData  OPTIONAL,
   commercialCardData       [ 6] CommercialCardData  OPTIONAL,
   orderSummary             [ 7] EXPLICIT SETString { ub-summary }  OPTIONAL,
   customerReferenceNumber  [ 8] EXPLICIT SETString { ub-reference }  OPTIONAL,
   customerServicePhone     [ 9] EXPLICIT Phone  OPTIONAL,
   okToPrintPhoneInd        [10] BOOLEAN DEFAULT TRUE,
   saleExtensions           [11] MsgExtensions {{SaleExtensionsIOS}}  OPTIONAL
}
```

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,
1926    marketSpecSaleData       [ 5] MarketSpecSaleData  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,
1932    saleExtensions           [11] MsgExtensions {{SaleExtensionsIOS}}  OPTIONAL
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)

```
MarketSpecDataID ::= ENUMERATED {
  failedEdit  (0),
  auto        (1),
  hotel       (2),
  transport   (3)
}

MarketSpecCapData ::= CHOICE {
   auto-rental  [0] MarketAutoCap,
   hotel        [1] MarketHotelCap,
   transport    [2] MarketTransportCap
}


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
}

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
}
```

Continued on next page

SaleDetail, continued

SaleDetail (continued)

```
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
}
```

Continued on next page

SaleDetail, continued

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

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

Table 34: CommercialCardData

Continued on next page

SaleDetail, continued

CommercialCardData (continued)

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

Table 34: CommercialCardData, continued

Continued on next page

SaleDetail, continued

CommercialCardData (continued)

```
CommercialCardData ::= SEQUENCE {
   chargeInfo        [0] ChargeInfo  OPTIONAL,
   merchantLocation  [1] Location  OPTIONAL,
   shipFrom          [2] Location  OPTIONAL,
   shipTo            [3] Location  OPTIONAL,
   itemSeq           [4] ItemSeq  OPTIONAL
}

ChargeInfo ::= SEQUENCE {
   totalFreightShippingAmount  [ 0] CurrencyAmount  OPTIONAL,
   totalDutyTariffAmount       [ 1] CurrencyAmount  OPTIONAL,
   dutyTariffReference         [ 2] EXPLICIT SETString { ub-reference } OPTIONAL,
   totalNationalTaxAmount      [ 3] CurrencyAmount  OPTIONAL,
   totalLocalTaxAmount         [ 4] CurrencyAmount  OPTIONAL,
   totalOtherTaxAmount         [ 5] CurrencyAmount  OPTIONAL,
   totalTaxAmount              [ 6] CurrencyAmount  OPTIONAL,
   merchantTaxID               [ 7] EXPLICIT SETString { ub-taxID }  OPTIONAL,
   merchantDutyTariffRef       [ 8] EXPLICIT SETString { ub-reference } OPTIONAL,
   customerDutyTariffRef       [ 9] EXPLICIT SETString { ub-reference } OPTIONAL,
   summaryCommodityCode        [10] EXPLICIT SETString { ub-commCode }  OPTIONAL,
   merchantType                [11] EXPLICIT SETString { ub-merType }  OPTIONAL
}

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
}

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

ItemSeq ::= SEQUENCE SIZE(1..ub-items) OF Item
```

Continued on next page

SaleDetail, continued

CommercialCardData (continued)

```
Item ::= SEQUENCE {
   quantity           INTEGER (1..MAX) DEFAULT 1,
   unitOfMeasureCode  [ 0] EXPLICIT SETString { ub-unitMeasure }  OPTIONAL,
   descriptor         SETString { ub-description  },
   commodityCode      [ 1] EXPLICIT SETString { ub-commCode }  OPTIONAL,
   productCode        [ 2] EXPLICIT SETString { ub-productCode }  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
}
```

```
Item ::= SEQUENCE {
   quantity           INTEGER (1..MAX) DEFAULT 1,
   unitOfMeasureCode  [ 0] EXPLICIT SETString { ub-unitMeasure }  OPTIONAL,
   descriptor         SETString { ub-description  },
   commodityCode      [ 1] EXPLICIT SETString { ub-commCode }  OPTIONAL,
   productCode        [ 2] EXPLICIT SETString { ub-productCode }  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
}
```

Continued on next page

SaleDetail, continued

MarketAutoCap

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

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

Table 35: MarketAutoCap

Continued on next page

SaleDetail, continued

MarketAutoCap (continued)

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

Table 35: MarketAutoCap, continued

Continued on next page

SaleDetail, continued

MarketAutoCap (continued)

```
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
}

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
}
```

Continued on next page

SaleDetail, continued

MarketAutoCap (continued)

```
AutoApplicableRate ::= CHOICE {
   dailyRentalRate   [0] CurrencyAmount,
   weeklyRentalRate  [1] CurrencyAmount
}

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.

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

Table 36: MarketHotelCap

Continued on next page

SaleDetail, continued

MarketHotelCap (continued)

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

Table 36: MarketHotelCap, continued

```
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
}
```

Continued on next page

SaleDetail, continued

MarketHotelCap (continued)

```
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
}
```

Continued on next page

SaleDetail, continued

MarketTransportCap

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

MarketTransportCap	**{PassengerName, DepartureDate, OrigCityAirport, [TripLegSeq], [TicketNumber], [TravelAgencyCode], [TravelAgencyName], [Restrictions]}**
PassengerName	*The name of the passenger to whom the tickets were issued.*
DepartureDate	*The departure date.*
OrigCityAirport	*The city of origin for the trip.*
TripLegSeq	**{TripLeg +}** *1 to 16* **TripLeg** *records*
TicketNumber	*The ticket number.*
TravelAgencyCode	*The travel agency code.*
TravelAgencyName	*The travel agency name.*
Restrictions	*Enumerated code indicating restrictions on refunds or changes.*
TripLeg	**{DateOfTravel, CarrierCode, ServiceClass, StopOverCode, DestCityAirport, [FareBasisCode], [DepartureTax]}**
DateOfTravel	*The date of travel for this trip leg.*
CarrierCode	*The carrier code for this trip leg.*
ServiceClass	*The class of service for this trip leg.*
StopOverCode	*Enumerated code indicating whether stopovers are permitted for this trip leg.*
DestCityAirport	*The destination city for this trip leg.*
FareBasisCode	*The fare basis code for this trip leg.*
DepartureTax	*The departure tax for this trip leg.*

Table 37: MarketTransportCap

```
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
}
```

Continued on next page

SaleDetail, continued

MarketTransportCap (continued)

```
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)
}
```

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:

Topic	Page
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

Purchase Initialization Pair

PInitReq

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**.

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

Table 38: PInitReq

```
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,
   piRqExtensions  [2] MsgExtensions {{PIRqExtensionsIOS}}  OPTIONAL
}
```

Continued on next page

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 }
```

Continued on next page

Purchase Initialization Pair, continued

PInitRes

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

Table 39: PInitRes

```
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
}
```

Continued on next page

Purchase Initialization Pair, continued

PInitRes (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

191 BrandCRLIdentifier ::= SIGNED {
192    EncodedBrandCRLID
193 } ( CONSTRAINED 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 }
```

Purchase Pair

PReq

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.

PReq	**< PReqDualSigned, PReqUnsigned >**
PReqDualSigned	*See page 77.*
PReqUnsigned	*See page 79.*

Table 40: PReq

```
PReq ::= CHOICE {
   pReqDualSigned  [0] EXPLICIT PReqDualSigned,
   pReqUnsigned    [1] EXPLICIT PReqUnsigned
}

PReqDualSigned ::= SEQUENCE {
   piDualSigned  PIDualSigned,
   oiDualSigned  OIDualSigned
}

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

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**.

PReqDualSigned	**{PIDualSigned, OIDualSigned}**
PIDualSigned	*See "PI (Payment Instruction)" on page 35.*
OIDualSigned	**L(OIData, PIData)**
OIData	*See page 81.*
PIData	**{PIHead, PANData}** *See page 37 for* **PIHead**. *See page 45 for* **PANData**.

Table 41: PReqDualSigned

```
PReqDualSigned ::= SEQUENCE {
   piDualSigned  PIDualSigned,
   oiDualSigned  OIDualSigned
}

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

OIDualSigned ::= L { OIData, PIData }
```

Continued on next page

Purchase Pair, continued

PReqDualSigned (continued)

```
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
}

PIData ::= SEQUENCE {
   piHead   PIHead,
   panData  PANData
}
```

Continued on next page

Purchase Pair, continued

PReqUnsigned Sent by Cardholders without certificates.

PReqUnsigned	**{PIUnsigned, OIUnsigned}**
PIUnsigned	*See "PI (Payment Instruction)" on page 35.*
OIUnsigned	**L(OIData, PIDataUnsigned)**
OIData	*See page 81.*
PIDataUnsigned	**{PIHead, PANToken}** *See page 37 for* **PIHead**. *See page 46 for* **PANToken**.

Table 42: PReqUnsigned

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

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

OIUnsigned ::= L { OIData, PIDataUnsigned }

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
}

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

Continued on next page

Purchase Pair, continued

PReqUnsigned (continued)

```
PIHead ::= SEQUENCE {
   transIDs          TransIDs,
   inputs            Inputs,
   merchantID        MerchantID,
   installRecurData  [0] InstallRecurData  OPTIONAL,
   transStain        TransStain,
   swIdent           SWIdent,
   acqBackKeyData    [1] EXPLICIT BackKeyData  OPTIONAL,
   piExtensions      [2] MsgExtensions {{PIExtensionsIOS}} OPTIONAL
}
```

```
PANToken ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   exNonce     Nonce
}
```

Continued on next page

Purchase Pair, continued

OIData

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

Table 43: OIData

Continued on next page

Purchase Pair, continued

OIData (continued)

HODInput	**{OD, PurchAmt, ODSalt, [InstallRecurData], [ODExtensions]}**
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

```
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
}
```

Continued on next page

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                                  -- Order description

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

Continued on next page

Purchase Pair, continued

PRes

PRes	**S(M, PResData)**
PResData	**{TransIDs, RRPID, Chall-C, [BrandCRLIdentifier], PResPayloadSeq}**
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

```
PRes ::= S { M, PResData }

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

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

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

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

Continued on next page

Purchase Pair, continued

PRes

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

PResPayload ::= SEQUENCE {
   completionCode  CompletionCode,
   results         Results  OPTIONAL,
   pRsExtensions   [0] MsgExtensions {{PRsExtensionsIOS}} OPTIONAL
}
```

Continued on next page

Purchase Pair, continued

PResPayload

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

Table 45: PResPayload

Continued on next page

Purchase Pair, continued

PResPayload (continued)

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

Table 45: PResPayload, continued

```
PResPayload ::= SEQUENCE {
   completionCode  CompletionCode,
   results         Results  OPTIONAL,
   pRsExtensions   [0] MsgExtensions {{PRsExtensionsIOS}} OPTIONAL
}
```

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    installRecurMismatch    (13),
1157    captureNotSupported     (14),
1158    signatureRequired       (15),
1159    cardMerchBrandMismatch  (16)
1160 }
```

Continued on next page

Purchase Pair, continued

PResPayload (continued)

```
CreditStatus ::= SEQUENCE {
   creditDate   Date,
   creditCode   CapRevOrCredCode,
   creditRatio  FloatingPoint
}
```

```
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)
}
```

Purchase Inquiry Pair

InqReq

InqReq	**< InqReqSigned, InqReqData >**
InqReqSigned	**S(C, InqReqData)**
InqReqData	**{TransIDs, RRPID, Chall-C2, [InqRqExtensions]}**
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.*

Table 46: InqReq

```
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
}

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

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

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

Table 48: AuthReq

Continued on next page

Authorization Pair, continued

AuthReq (continued)

HOIData	**DD(OIData)** *See page 81 for the definition of* **OIData**. *An independent hash computed by Merchant. Payment Gateway compares with Cardholder-produced copy in* **PI** *to verify linkage from* **PI** *to* **OIData**.
HOD2	**DD(HODInput)** *See "OIData" on page 81 for definition of* **HODInput**. *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.*

Table 48: AuthReq, continued

```
AuthReq ::= EncB { M, P, 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 } )

PI ::= CHOICE {
   piUnsigned    [0] EXPLICIT PIUnsigned,
   piDualSigned  [1] EXPLICIT PIDualSigned,
   authToken     [2] EXPLICIT AuthToken
}

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

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,
1926    marketSpecSaleData       [ 5] MarketSpecSaleData  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,
1932    saleExtensions           [11] MsgExtensions {{SaleExtensionsIOS}}  OPTIONAL
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    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 }

 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

AuthReqPayload	**{SubsequentAuthInd, AuthReqAmt, [AVSData], [SpecialProcessing], [CardSuspect], RequestCardTypeInd, [InstallRecurData], [MarketSpecAuthData], MerchData, [ARqExtensions]}**
SubsequentAuthInd	*Boolean indicating Merchant requests an additional authorization because of a split shipment*
AuthReqAmt	*May differ from* **PurchAmt***; acquirer policy may place limitations on the permissible difference*
AVSData	**{[StreetAddress], Location}** *Cardholder billing address; contents are received from cardholder using an out-of-band mechanism* *See page 52 for definition of* **Location***.*
SpecialProcessing	*Enumerated field indicating the type of special processing requested.*
CardSuspect	*Enumerated code indicating that Merchant is suspicious of the Cardholder and the reason for the suspicion*
RequestCardTypeInd	*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.*
InstallRecurData	*See page 42.*
MarketSpecAuthData	**< MarketAutoAuth, MarketHotelAuth, MarketTransportAuth >** *Market-specific authorization data*
MerchData	**{ [MerchCatCode], [MerchGroup]}**
ARqExtensions	*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.*
StreetAddress	*The street address of the cardholder*

Table 49: AuthReqPayload

Continued on next page

Authorization Pair, continued

AuthReqPayload (continued)

MarketAutoAuth	**{Duration}**
MarketHotelAuth	**{Duration, [Prestige]}**
MarketTransportAuth	**{}** *There is currently no authorization data for this market segment.*
MerchCatCode	*Four-byte code (defined in ANSI X9.10) describing Merchant's type of business, product, or service*
MerchGroup	*Enumerated code identifying the general category of the merchant*
Duration	*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.*
Prestige	*Enumerated type of prestigious property; the meaning of the various levels are defined by the payment card brand*

Table 49: AuthReqPayload, continued

```
AuthReqPayload ::= SEQUENCE {
   subsequentAuthInd   BOOLEAN DEFAULT FALSE,
   authReqAmt          CurrencyAmount,        -- May differ from PurchAmt
   avsData             [0] AVSData  OPTIONAL,
   specialProcessing   [1] SpecialProcessing  OPTIONAL,
   cardSuspect         [2] CardSuspect  OPTIONAL,
   requestCardTypeInd  BOOLEAN DEFAULT FALSE,
   installRecurData    [3] InstallRecurData  OPTIONAL,
   marketSpecAuthData  [4] EXPLICIT MarketSpecAuthData  OPTIONAL,
   merchData           MerchData,
   aRqExtensions       [5] MsgExtensions {{ARqExtensionsIOS}} OPTIONAL
}
```

Continued on next page

Authorization Pair, continued

AuthReqPayload (continued)

```
AVSData ::= SEQUENCE {
   streetAddress  SETString { ub-AVSData } OPTIONAL,
   location       Location
}

SpecialProcessing ::= ENUMERATED {
   directMarketing    (0),
   preferredCustomer  (1)
}

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

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

MarketSpecAuthData ::= CHOICE {
   auto-rental  [0] MarketAutoAuth,
   hotel        [1] MarketHotelAuth,
   transport    [2] MarketTransportAuth
}

MerchData ::= SEQUENCE {
   merchCatCode  MerchCatCode  OPTIONAL,
   merchGroup    MerchGroup  OPTIONAL
}

MarketAutoAuth ::= SEQUENCE {
   duration  Duration
}
```

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

AuthRes

AuthRes	**< EncB(P, M, AuthResData, AuthResBaggage), EncBX(P, M, AuthResData, AuthResBaggage, PANToken) >**
AuthResData	**{AuthTags, [BrandCRLIdentifier], [PEThumb], AuthResPayload}**
AuthResBaggage	**{[CapToken], [AcqCardMsg], [AuthToken]}**
PANToken	*See page 46. Sent if Merchant certificate indicates Merchant is entitled to the information.*
AuthTags	*Copied from corresponding* **AuthReq**; **TransIDs** *and* **AuthRetNum** *may be updated with current information*
BrandCRLIdentifier	*List of current CRLs for all CAs under a Brand CA. See page 151.*
PEThumb	*Thumbprint of Payment Gateway certificate provided if* **AuthReq.MThumbs** *indicates Merchant needs one*
AuthResPayload	*See page 101.*
CapToken	*See page 44.*
AcqCardMsg	*If Cardholder included* **AcqBackKeyData** *in* **PIHead**, *the Payment Gateway may send this field to the Merchant containing a message (encrypted using the key data) for the Cardholder. The Merchant is required to copy* **AcqCardMsg** *to any subsequent* **PRes** *or* **InqRes** *sent to the Cardholder. See page 43.*
AuthToken	*Merchant uses as the* **PI** *in a subsequent* **AuthReq**. *See page 40.*

Table 50: AuthRes

```
AuthRes ::= CHOICE {
   encB   [0] EXPLICIT EncB { P, M, AuthResData, AuthResBaggage },
   encBX  [1] EXPLICIT EncBX { P, M, AuthResData, AuthResBaggage, PANToken }
}
```

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

AuthResPayload	**{AuthHeader, [CapResPayload], [ARsExtensions]}**
AuthHeader	**{AuthAmt, AuthCode, ResponseData, [BatchStatus], [CurrConv]}**
CapResPayload	*See page 119.* *Returned if* **CaptureNow** *had a value of* **TRUE** *in* **AuthReq**.
ARsExtensions	*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.*
AuthAmt	*Copied from* **AuthReqPayload.AuthReqAmt**
AuthCode	*Enumerated code indicating outcome of payment authorization processing*
ResponseData	**{[AuthValCodes], [RespReason], [CardType], [AVSResult], [LogRefID]}**
BatchStatus	*See page 47.*
CurrConv	**{CurrConvRate, CardCurr}**
AuthValCodes	**{[ApprovalCode], [AuthCharInd], [ValidationCode], [MarketSpecDataID]}**
RespReason	*Enumerated code that indicates authorization service entity and (if appropriate) reason for decline*
CardType	*Enumerated code indicating the type of card used for the transaction*
AVSResult	*Enumerated Address Verification Service response code*
LogRefID	*Alphanumeric data assigned to the authorization transaction (used for matching to reversals)*

Table 51: AuthResPayload

Continued on next page

Authorization Pair, continued

AuthResPayload (continued)

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

Table 51: AuthResPayload, continued

```
AuthResPayload ::= SEQUENCE {
   authHeader     AuthHeader,
   capResPayload  CapResPayload  OPTIONAL,
   aRsExtensions  [0] MsgExtensions {{ARsExtensionsIOS}} OPTIONAL
}

AuthHeader ::= SEQUENCE {
   authAmt       CurrencyAmount,
   authCode      AuthCode,
   responseData  ResponseData,
   batchStatus   [0] BatchStatus  OPTIONAL,
   currConv      CurrConv  OPTIONAL          -- Merchant to cardholder
}

CapResPayload ::= SEQUENCE {
   capCode           CapCode,
   capAmt            CurrencyAmount,
   batchID           [0] BatchID  OPTIONAL,
   batchSequenceNum  [1] BatchSequenceNum  OPTIONAL,
   cRsPayExtensions  [2] MsgExtensions {{CRsPayExtensionsIOS}} OPTIONAL
}
```

Continued on next page

Authorization Pair, continued

AuthResPayload (continued)

```
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
}

BatchStatus ::= SEQUENCE {
   openDateTime     Date,
   closedWhen       [0] ClosedWhen  OPTIONAL,
   batchDetails     BatchDetails,
   batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL
}

CurrConv ::= SEQUENCE {
   currConvRate  FloatingPoint,
   cardCurr      Currency
}

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

Continued on next page

Authorization Pair, continued

AuthResPayload (continued)

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

CardType ::= ENUMERATED {
   unavailable           ( 0),
   classic               ( 1),
   gold                  ( 2),
   platinum              ( 3),
   premier               ( 4),
   debit                 ( 5),
   pinBasedDebit         ( 6),
   atm                   ( 7),
   electronicOnly        ( 8),
   unspecifiedConsumer   ( 9),
   corporateTravel       (10),
   purchasing            (11),
   business              (12),
   unspecifiedCommercial  (13),
   privateLabel          (14),
   proprietary           (15)
}

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)
}
```

Continued on next page

Authorization Pair, continued

AuthResPayload (continued)

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

MarketSpecDataID ::= ENUMERATED {
  failedEdit  (0),
  auto        (1),
  hotel       (2),
  transport   (3)
}
```

Authorization Reversal Pair

AuthRevReq

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

AuthRevReq	**EncB(M, P, AuthRevReqData, AuthRevReqBaggage)**
AuthRevReqData	**{AuthRevTags, [MThumbs], [AuthReqData], [AuthResPayload], AuthNewAmt, [ARvRqExtensions]}**
AuthRevReqBaggage	**{PI, [CapToken]}**
AuthRevTags	**{AuthRevRRTags, [AuthRetNum]}**
MThumbs	*Thumbprints of certificates, CRLs, and Brand CRL Identifiers currently held in Merchant's cache*
AuthReqData	*Copied from prior, corresponding* **AuthReq.** *Not required in message if* **CapToken** *generated by Payment Gateway contains all relevant data.*
AuthResPayload	*Copied from prior, corresponding* **AuthRes.** *Not required in message if* **CapToken** *generated by Payment Gateway contains all relevant data.*
AuthNewAmt	*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.*
ARvRqExtensions	*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.*

Continued on next page

Authorization Reversal Pair, continued

AuthRevReq (continued)

PI	*Copied from prior, corresponding* **AuthReq**
CapToken	*Copied from prior, corresponding* **AuthRes**
AuthRevRRTags	**RRTags,** *see page 34.* *Fresh* **RRPID** *and* **Date** *for* **AuthRev** *pair*
AuthRetNum	*Identification of the authorization request used within the financial network*

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,
1240    authResPayload   [2] AuthResPayload  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

AuthRevRes	**< EncB(P, M, AuthRevResData, AuthRevResBaggage), Enc(P, M, AuthRevResData) >**
AuthRevResData	**{AuthRevCode, AuthRevTags, [BrandCRLIdentifier], [PEThumb], AuthNewAmt, AuthResDataNew, [ARvRsExtensions]}**
AuthRevResBaggage	**{[CapTokenNew], [AuthTokenNew]}**
AuthRevCode	*Enumerated code indicating outcome of payment authorization reversal processing*
AuthRevTags	*Copied from corresponding* **AuthRevReq**
BrandCRLIdentifier	*List of current CRLs for all CAs under a Brand CA. See page 151.*
PEThumb	*Thumbprint of Payment Gateway certificate provided if* **AuthRevReq.MThumbs** *indicates Merchant needs one*
AuthNewAmt	*Copied from corresponding* **AuthRevReq**
AuthResDataNew	**{TransIDs, [AuthResPayloadNew]}** *If* **AuthNewAmt** *is not 0, Payment Gateway creates a new instance of* **AuthResData** *(see "AuthRes" on page 99).*
ARvRsExtensions	*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.*
CapTokenNew	*New Capture Token (with updated fields), if* **AuthNewAmt** *is not 0. This replaces the* **CapToken** *returned in the corresponding* **AuthRes**.

Continued on next page

Authorization Reversal Pair, continued

AuthRevRes (continued)

AuthTokenNew	*New Authorization Token (with updated fields). Merchant uses as the* **PI** *in a subsequent* **AuthReq**. *See "AuthToken" on page 40.*
TransIDs	*Copied from corresponding* **AuthRevReq**
AuthResPayloadNew	*Formally identical to* **AuthResPayload** *(see page 101); if* **AuthNewAmt** *is not 0.*

Table 53: AuthRevRes

```
AuthRevRes ::= CHOICE {
   encB  [0] EXPLICIT EncB { P, M, AuthRevResData, AuthRevResBaggage },
   enc   [1] EXPLICIT Enc { P, M, AuthRevResData }
}

AuthRevResData ::= SEQUENCE {
   authRevCode         AuthRevCode,
   authRevTags         AuthRevTags,
   brandCRLIdentifier  [0] EXPLICIT BrandCRLIdentifier  OPTIONAL,
   peThumb             [1] EXPLICIT CertThumb  OPTIONAL,
   authNewAmt          CurrencyAmount,                    -- May be zero
   authResDataNew      AuthResDataNew,
   aRvRsExtensions     [2] MsgExtensions {{ARvRsExtensionsIOS}} OPTIONAL
}

AuthRevResBaggage ::= SEQUENCE {
   capTokenNew   CapToken  OPTIONAL,
   authTokenNew  AuthToken  OPTIONAL
}
```

Continued on next page

Authorization Reversal Pair, continued

AuthRevRes (continued)

```
AuthRevTags ::= SEQUENCE {
   authRevRRTags  AuthRevRRTags,
   authRetNum     AuthRetNum  OPTIONAL
}

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

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

TransIDs ::= SEQUENCE {
   lid-C      LocalID,
   lid-M      [0] LocalID  OPTIONAL,
   xid        XID,
   pReqDate   Date,
   paySysID   [1] PaySysID  OPTIONAL,
   language   Language          -- Cardholder requested session language
}
```

Capture Pair

CapReq

CapReq	**< EncB(M, P, CapReqData, CapTokenSeq), EncBX(M, P, CapReqData, CapTokenSeq, PANToken) >** **CapTokenSeq** *is external "baggage".* *If* **PANToken** *is included, it must correspond to a single* **CapItem** *and a single* **CapToken** *in* **CapTokenSeq**.
CapReqData	**{CapRRTags, [MThumbs], CapItemSeq, [CRqExtensions]}**
CapTokenSeq	**{[CapToken] +}** *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.
PANToken	*See page 46*
CapRRTags	**RRTags,** *see page 34.* *Fresh* **RRPID** *and* **Date**
MThumbs	*Thumbprints of certificates, CRLs, and Brand CRL Identifiers currently held in Merchant's cache*
CapItemSeq	**{CapItem +}** *One or more* **CapItem** *in an ordered array*

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

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

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

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

PANToken ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   exNonce     Nonce
}

CapRRTags ::= RRTags

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

Continued on next page

Capture Pair, continued

CapReq (continued)

```
CapToken ::= CHOICE {
   encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
   enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
   null  [2] EXPLICIT NULL
}

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

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

CapPayload ::= SEQUENCE {
   capDate         Date,
   capReqAmt       CurrencyAmount,
   authReqItem     [0] AuthReqItem  OPTIONAL,
   authResPayload  [1] AuthResPayload  OPTIONAL,
   saleDetail      [2] SaleDetail  OPTIONAL,
   cPayExtensions  [3] MsgExtensions {{CPayExtensionsIOS}} OPTIONAL
}
```

Continued on next page

Capture Pair, continued

CapPayload

CapPayload	**{CapDate, CapReqAmt, [AuthReqItem], [AuthResPayload], [SaleDetail], [CPayExtensions]}**
CapDate	*Date of capture; this is the Transaction Date that will appear on the Cardholder's statement*
CapReqAmt	*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*
AuthReqItem	*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*
AuthResPayload	*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*
SaleDetail	*See page 53.*
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

```
CapPayload ::= SEQUENCE {
   capDate         Date,
   capReqAmt       CurrencyAmount,
   authReqItem     [0] AuthReqItem  OPTIONAL,
   authResPayload  [1] AuthResPayload  OPTIONAL,
   saleDetail      [2] SaleDetail  OPTIONAL,
   cPayExtensions  [3] MsgExtensions {{CPayExtensionsIOS}} OPTIONAL
}
```

Continued on next page

Capture Pair, continued

CapPayload (continued)

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

AuthResPayload ::= SEQUENCE {
   authHeader     AuthHeader,
   capResPayload  CapResPayload  OPTIONAL,
   aRsExtensions  [0] MsgExtensions {{ARsExtensionsIOS}} OPTIONAL
}

SaleDetail ::= SEQUENCE {
   batchID                  [ 0] BatchID  OPTIONAL,
   batchSequenceNum         [ 1] BatchSequenceNum  OPTIONAL,
   payRecurInd              [ 2] PayRecurInd  OPTIONAL,
   merOrderNum              [ 3] MerOrderNum  OPTIONAL,
   authCharInd              [ 4] AuthCharInd  OPTIONAL,
   marketSpecSaleData       [ 5] MarketSpecSaleData  OPTIONAL,
   commercialCardData       [ 6] CommercialCardData  OPTIONAL,
   orderSummary             [ 7] EXPLICIT SETString { ub-summary }  OPTIONAL,
   customerReferenceNumber  [ 8] EXPLICIT SETString { ub-reference }  OPTIONAL,
   customerServicePhone     [ 9] EXPLICIT Phone  OPTIONAL,
   okToPrintPhoneInd        [10] BOOLEAN DEFAULT TRUE,
   saleExtensions           [11] MsgExtensions {{SaleExtensionsIOS}}  OPTIONAL
}
```

Continued on next page

Capture Pair, continued

CapRes

CapRes	**Enc(P, M, CapResData)**
CapResData	**{CapRRTags, [BrandCRLIdentifier], [PEThumb], [BatchStatusSeq], CapResItemSeq, [CRsExtensions]}**
CapRRTags	**RRTags** *(see page 34); copied from* **CapReq**
BrandCRLIdentifier	*List of current CRLs for all CAs under a Brand CA . See page 151.*
PEThumb	*Thumbprint of Payment Gateway certificate provided if* **CapReqData.MThumbs** *indicates Merchant needs one*
BatchStatusSeq	**{BatchStatus +}**
CapResItemSeq	**{CapResItem +}** *Order corresponds to* **CapReq**
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    peThumb             [1] EXPLICIT CertThumb  OPTIONAL,
1369    batchStatusSeq      [2] BatchStatusSeq  OPTIONAL,
1370    capResItemSeq       CapResItemSeq,
1371    cRsExtensions       [3] MsgExtensions {{CRsExtensionsIOS}} OPTIONAL
1372 }

1339 CapRRTags ::= RRTags

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

1716 BatchStatusSeq ::= SEQUENCE OF BatchStatus

1376 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,
1389    cRsPayExtensions  [2] MsgExtensions {{CRsPayExtensionsIOS}} OPTIONAL
1390 }
```

Continued on next page

Capture Pair, continued

CapResPayload

CapResPayload	**{CapCode, CapAmt, [BatchID], [BatchSequenceNum], [CRsPayExtensions]}**
CapCode	*Enumerated code indicating status of capture.*
CapAmt	*Copied from corresponding* **CapReq**
BatchID	*Identification of the settlement batch for merchant-acquirer accounting; copied from corresponding* **CapReq**
BatchSequenceNum	*The sequence number of this item within the batch; copied from corresponding* **CapReq**
CRsPayExtensions	*The data in an extension to the capture response payload 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 an individual item in the capture response; data related to the entire capture response message should be placed in an extension to CapResData.*

Table 57: CapResPayload

```
CapResPayload ::= SEQUENCE {
   capCode          CapCode,
   capAmt           CurrencyAmount,
   batchID          [0] BatchID  OPTIONAL,
   batchSequenceNum  [1] BatchSequenceNum  OPTIONAL,
   cRsPayExtensions  [2] MsgExtensions {{CRsPayExtensionsIOS}} OPTIONAL
}
```

Continued on next page

Capture Pair, continued

CapResPayload (continued)

```
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)
}

BatchID ::= INTEGER (0..MAX)

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:

Topic	Page
CapRevOrCredReqData	122
CapRevOrCredResData	125
CapRevOrCredResPayload	127

Continued on next page

Capture Reversal Or Credit, continued

CapRevOrCredReqData

CapRevOrCredReqData	**{CapRevOrCredRRTags, [MThumbs], CapRevOrCredReqItemSeq, [CRvRqExtensions]}**
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)

AuthRRPID	*The RRPID that appeared in the corresponding* **AuthReq** *or* **AuthRevReq**
CapPayload	*See page 115.*
NewBatchID	*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.*
CapRevOrCredReqDate	*The date the request is submitted.*
CapRevOrCredReqAmt	*In credit requests, the amount of credit requested, which may differ from* **AuthAmt** *in* **CapToken** *and* **CapReqAmt** *in* **CapPayload**.
NewAccountInd	*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.*
CRvRqItemExtensions	*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**.

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,
1353    authResPayload  [1] AuthResPayload  OPTIONAL,
1354    saleDetail      [2] SaleDetail  OPTIONAL,
1355    cPayExtensions  [3] MsgExtensions {{CPayExtensionsIOS}} OPTIONAL
1356 }
```

Continued on next page

Capture Reversal Or Credit, continued

CapRevOrCredResData

CapRevOrCredResData	**{CapRevOrCredRRTags, [BrandCRLIdentifier], [PEThumb], [BatchStatusSeq], CapRevOrCredResItemSeq, [CRvRsExtensions]}**
CapRevOrCredRRTags	**RRTags***(see page 34); copied* **CapRevOrCredRRTags** *from corresponding* **CapRevOrCredReqData**
BrandCRLIdentifier	*List of current CRLs for all CAs under a Brand CA. See page 151.*
PEThumb	*Thumbprint of Payment Gateway certificate provided if* **CapRevOrCredReq.MThumbs** *indicates Merchant needs one*
BatchStatusSeq	**{BatchStatus +}**
CapRevOrCredResItemSeq	**{CapRevOrCredResItem +}** *One or more* **CapRevOrCredResItem** *in an ordered array*
CRvRsExtensions	*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.*
BatchStatus	*See page 47.*
CapRevOrCredResItem	**{TransIDs, AuthRRPID, CapRevOrCredResPayload}**
TransIDs	*Copied from corresponding* **CapRevOrCredReqData.AuthReqData.AuthTags**
AuthRRPID	*The RRPID that appeared in the corresponding* **AuthReq** *or* **AuthRevReq**
CapRevOrCredResPayload	*See page 127.*

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,
1441    cRvRsExtensions         [3] MsgExtensions {{CRvRsExtensionsIOS}} OPTIONAL
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,
1459    cRvRsPayExtensions     [2] MsgExtensions {{CRvRsPayExtensionsIOS}} OPTIONAL
1460 }
```

Continued on next page

Capture Reversal Or Credit, continued

CapRevOrCredResPayload

CapRevOrCredResPayload	**{CapRevOrCredCode, CapRevOrCredActualAmt, [BatchID], [BatchSequenceNum], [CRvRsPayExtensions]}**
CapRevOrCredCode	*Enumerated code indicating capture reversal or credit status*
CapRevOrCredActualAmt	*Copied from corresponding* **CapRevOrCredReqItem**
BatchID	*Identification of the settlement batch for merchant-acquirer accounting*
BatchSequenceNum	*The sequence number of this item within the batch*
CRvRsPayExtensions	*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.*

Table 60: CapRevOrCredResPayload

```
CapRevOrCredResPayload ::= SEQUENCE {
   capRevOrCredCode       CapRevOrCredCode,
   capRevOrCredActualAmt  CurrencyAmount,
   batchID                [0] BatchID  OPTIONAL,
   batchSequenceNum       [1] BatchSequenceNum  OPTIONAL,
   cRvRsPayExtensions     [2] MsgExtensions {{CRvRsPayExtensionsIOS}} OPTIONAL
}
```

Continued on next page

Capture Reversal Or Credit, continued

CapRevOrCredResPayload (continued)

```
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)
}

BatchID ::= INTEGER (0..MAX)

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

Capture Reversal Pair

CapRevReq

CapRevReq	**< EncB(M, P, CapRevData, CapTokenSeq), EncBX(M, P, CapRevData, CapTokenSeq, PANToken) >** **CapTokenSeq** *is external "baggage".* *If* **PANToken** *is included, it must correspond to a single entry in* **CapRevData.CapRevOrCredReqItemSeq** *and a single* **CapToken** in **CapTokenSeq**
CapRevData	**CapRevOrCredReqData***; see page 122.*
CapTokenSeq	**{[CapToken] +}** *One or more* **CapTokens,** *in ordered one-to-one correspondence with* **CapRevOrCredReqItem** *sequence in* **CapRevOrCredReqData.CapRevOrCredReqItemSeq**. *Note: Any* **CapToken** *may be omitted; that is, may be* NULL.
PANToken	*See page 46*
CapToken	*Copied from corresponding* **AuthRes** *or* **AuthRevRes**

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)

```
PANToken ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   exNonce     Nonce
}

CapToken ::= CHOICE {
   encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
   enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
   null  [2] EXPLICIT NULL
}
```

Continued on next page

Capture Reversal Pair, continued

CapRevRes

CapRevRes	**Enc(P, M, CapRevResData)**
CapRevResData	**CapRevOrCredResData***; see page 125.*

Table 62: CapRevRes

```
1503 CapRevRes ::= Enc { P, M, CapRevResData }

1508 CapRevResData ::= [0] EXPLICIT CapRevOrCredResData
```

Credit Pair

CredReq

CredReq	**< EncB(M, P, CredReqData, CapTokenSeq), EncBX(M, P, CredReqData, CapTokenSeq, PANToken) >** **CapTokenSeq** *is external "baggage".* *If* **PANToken** *is included, it must correspond to a single entry in* **CredReqData.CapRevOrCredReqItemSeq** and a single **CapToken** in **CapTokenSeq**
CredReqData	**CapRevOrCredReqData***; see page 122.*
CapTokenSeq	**{[CapToken] +}** *One or more* **CapTokens** *in ordered one-to-one correspondence with* **CapRevOrCredReqItem** *sequence in* **CapRevOrCredReqData.CapRevOrCredReqItemSeq**. *Note: Any* **CapToken** *may be omitted; that is, may be* NULL.
PANToken	*See page 46*
CapToken	*Copied from corresponding* **AuthRes** *or* **AuthRevRes**.

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)

```
CapToken ::= CHOICE {
   encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
   enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
   null  [2] EXPLICIT NULL
}
```

Continued on next page

Credit Pair, continued

CredRes

CredRes	**Enc(P, M, CredResData)**
CredResData	**CapRevOrCredResData***; see page 125.*

Table 64: CredRes

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

1539 CredResData ::= [1] EXPLICIT CapRevOrCredResData
```

Credit Reversal Pair

CredRevReq

CredRevReq	**< EncB(M, P, CredRevReqData, CapTokenSeq), EncBX(M, P, CredRevReqData, CapTokenSeq, PANToken) >** **CapTokenSeq** *is external "baggage".* *If* **PANToken** *is included, it must correspond to a single entry in* **CredRevReqData.CredRevReqSeq** *and a single* **CapToken** in **CapTokenSeq**.
CredRevReqData	**CapRevOrCredReqData***; see page 122.*
CapTokenSeq	**{[CapToken] +}** *One or more* **CapTokens***, in ordered one-to-one correspondence with* **CredRevReqItem** *in* **CapRevOrCredReqData.CapRevOrCredReqItemSeq**. *Note: Any* **CapToken** *may be omitted; that is, may be NULL.*
PANToken	*See page 46*
CapToken	*Copied from corresponding* **AuthRes** *or* **AuthRevRes**.

Table 65: CredRevReq

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

CredRevReqData ::= [2] EXPLICIT CapRevOrCredReqData

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

Continued on next page

Credit Reversal Pair, continued

CredRevReq (continued)

```
PANToken ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   exNonce     Nonce
}

CapToken ::= CHOICE {
   encX  [0] EXPLICIT EncX { P1, P2, CapTokenData, PANToken },
   enc   [1] EXPLICIT Enc { P1, P2, CapTokenData },
   null  [2] EXPLICIT NULL
}
```

Continued on next page

Credit Reversal Pair, continued

CredRevRes

CredRevRes	**Enc(P, M, CredRevResData)**
CredRevResData	**CapRevOrCredResData**; *see page 125.*

Table 66: CredRevRes

```
1565 CredRevRes ::= Enc { P, M, CredRevResData }

1570 CredRevResData ::= [2] EXPLICIT CapRevOrCredResData
```

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.

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

Table 67: PCertReq

```
PCertReq ::= S { M, PCertReqData }

PCertReqData ::= SEQUENCE {
   pCertRRTags     RRTags,
   mThumbs         [0] EXPLICIT Thumbs  OPTIONAL,
   brandAndBINSeq  BrandAndBINSeq,
   pcRqExtensions  [1] MsgExtensions {{PCRqExtensionsIOS}}  OPTIONAL
}

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

BrandAndBIN ::= SEQUENCE {
   brandID  BrandID,
   bin      BIN  OPTIONAL
}

BrandID ::= SETString { ub-BrandID }

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

Continued on next page

Payment Gateway Certificate Request Pair, continued

PCertRes

PCertRes	**S(P, PCertResTBS)**
PCertResTBS	**{PCertRRTags, [BrandCRLIdentifierSeq], PCertResItemSeq, [PCRsExtensions]}**
PCertRRTags	**RRTags** *(see page 34); copied from* **PCertReq**
BrandCRLIdentifierSeq	**{BrandCRLIdentifier +}**
PCertResItemSeq	**{PCertResItem +}** *One or more status codes and certificate thumbprints of the certificates that are returned in a one-to-one correspondance with* **PCertReq.BrandAndBINSeq**
PCRsExtensions	*Note: The Payment Gateway certificate response is not encrypted so this extension must not contain confidential information.*
BrandCRLIdentifier	*List of current CRLs for all CAs under a Brand CA. See page 151.*
PCertResItem	**{PCertCode, [CertThumb]}**
PCertCode	*Enumerated code indicating result of* **PCertReq**
CertThumb	*Thumbprint of returned certificate*

Table 68: PCertRes

```
PCertRes ::= S { P, PCertResTBS }

PCertResTBS ::= SEQUENCE {
   pCertRRTags               RRTags,
   pCertResItemSeq           PCertResItemSeq,
   brandCRLIdentifierSeq     [0] BrandCRLIdentifierSeq  OPTIONAL,
   pcRsExtensions            [1] MsgExtensions {{PCRsExtensionsIOS}}  OPTIONAL
}

PCertCode ::= ENUMERATED {
   success              (0),
   unspecifiedFailure   (1),
   brandNotSupported    (2),
   unknownBIN           (3)
}

BrandCRLIdentifierSeq ::= SEQUENCE SIZE(1..MAX) OF [0] EXPLICIT
BrandCRLIdentifier

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

Batch Administration Pair

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

BatchAdminReq	**Enc(M, P, BatchAdminReqData)**
BatchAdminReqData	**{BatchAdminRRTags, [BatchID], [BrandAndBINSeq], [BatchOperation], ReturnBatchSummaryInd, [ReturnTransactionDetail], [BatchStatus], [TransDetails], [BARqExtensions]}**
BatchAdminRRTags	**RRTags,** *see page 34.* *Fresh* **RRPID** *and* **Date**
BatchID	*Identification of the settlement batch for merchant-acquirer accounting*
BrandAndBINSeq	**{BrandAndBIN +}**
BatchOperation	*Enumerated value indicating the action to be performed on the batch.*
ReturnBatchSummaryInd	*Indicates batch summary data is to be returned in* **BatchAdminRes**
ReturnTransactionDetail	**{StartingPoint, MaximumItems, ErrorsOnlyInd, [BrandID]}** *If* **BrandID** *is specified, only items for that payment card brand are returned.*
BatchStatus	*See page 47.*
TransDetails	**{NextStartingPoint, TransactionDetailSeq}**
BARqExtensions	*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.*

Table 69: BatchAdminReq

Continued on next page

Batch Administration Pair, continued

BatchAdminReq (continued)

BrandAndBIN	**{BrandID, [BIN]}**
StartingPoint	*Zero indicates to send detail for the first group of items; otherwise,* **NextStartingPoint** *from a prior* **BatchAdminRes**.
MaximumItems	*The maximum number of items to be returned in this group of items.*
ErrorsOnlyInd	*Boolean indicating if only items with an error status should be returned.*
BrandID	*Payment card brand (without product type)*
NextStartingPoint	*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.*
TransactionDetailSeq	**{TransactionDetail +}**
BIN	*Bank Identification Number for the processing of Merchant's transactions at the Acquirer*
TransactionDetail	*See page 50.*

Table 69: BatchAdminReq, continued

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

1626 BatchAdminReqData ::= SEQUENCE {
1627    batchAdminRRTags         RRTags,
1628    batchID                  [0] BatchID  OPTIONAL,
1629    brandAndBINSeq           [1] BrandAndBINSeq  OPTIONAL,
1630    batchOperation           [2] BatchOperation  OPTIONAL,
1631    returnBatchSummaryInd    BOOLEAN DEFAULT FALSE,
1632    returnTransactionDetail  [3] ReturnTransactionDetail  OPTIONAL,
1633    batchStatus              [4] BatchStatus  OPTIONAL,
1634    transDetails             [5] TransDetails  OPTIONAL,
1635    baRqExtensions           [6] MsgExtensions {{BARqExtensionsIOS}} OPTIONAL
1636 }

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

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

1640 BatchOperation ::= ENUMERATED {
1641    open   (0),
1642    purge  (1),
1643    close  (2)
1644 }
```

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

BatchAdminRes	**Enc(P, M, BatchAdminResData)**
BatchAdminResData	**{BatchAdminTags, BatchID, [BAStatus], [BatchStatus], [TransmissionStatus], [SettlementInfo], [TransDetails], [BARsExtensions]}**
BatchAdminTags	**RRTags** *(see page 34); copied from prior* **BatchAdminReq**
BatchID	*Identification of the settlement batch for merchant-acquirer accounting*
BAStatus	*Enumerated code indicating status of batch open*
BatchStatus	*See page 47.*
TransmissionStatus	*Enumerated value indicating the status of the transmission from the gateway to the next upstream system*
SettlementInfo	**{SettlementAmount, SettlementType, SettlementAccount, SettlementDepositDate}**
TransDetails	**{NextStartingPoint, TransactionDetailSeq}**
BARsExtensions	*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.*
SettlementAmount	*The net settlement amount to the Merchant's account*
SettlementType	*Enumerated code indicating the type of amount*
SettlementAccount	*The Merchant's account*
SettlementDepositDate	*The date that the* **SettlementAmount** *will be credited to/debited from the Merchant's account*

Table 70: BatchAdminRes

Continued on next page

Batch Administration Pair, continued

BatchAdminRes (continued)

NextStartingPoint	*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.*
TransactionDetailSeq	**{TransactionDetail +}**
TransactionDetail	*See page 50.*

Table 70: BatchAdminRes, continued

```
BatchAdminRes ::= Enc { P, M, 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
}

BatchID ::= INTEGER (0..MAX)

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)
}
```

Continued on next page

Batch Administration Pair, continued

```
BatchStatus ::= SEQUENCE {
   openDateTime     Date,
   closedWhen       [0] ClosedWhen  OPTIONAL,
   batchDetails     BatchDetails,
   batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL
}

TransmissionStatus ::= ENUMERATED {
   pending                  (0),
   inProgress               (1),
   batchRejectedByAcquirer  (2),
   completedSuccessfully    (3),
   completedWithItemErrors  (4)
}

SettlementInfo ::= SEQUENCE {
   settlementAmount       CurrencyAmount,
   settlementType         AmountType,
   settlementAccount      SETString { ub-SettlementAccount },
   settlementDepositDate  Date
}

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

TransactionDetailSeq ::= SEQUENCE OF TransactionDetail

TransactionDetail ::= SEQUENCE {
   transIDs            TransIDs,
   authRRPID           RRPID,
   brandID             BrandID,
   batchSequenceNum    BatchSequenceNum,
   reimbursementID     ReimbursementID  OPTIONAL,
   transactionAmt      CurrencyAmount,
   transactionAmtType  AmountType,
   transactionStatus   [0] TransactionStatus  OPTIONAL,
   transExtensions     [1] MsgExtensions {{TransExtensionsIOS}} OPTIONAL
}
```

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:

Topic	Page
IDData	148
RequestType	149
End Entity and CA Types	150
BrandCRLIdentifier	151
PANData0	153
AcctData	154
RegFormOrReferral	155

IDData

IDData

IDData	**< MerchantAcquirerID, AcquirerID >** *Only for Merchants and Acquirers*
MerchantAcquirerID	**{MerchantBIN, MerchantID}**
AcquirerID	**{AcquirerBIN, [AcquirerBusinessID]}**
MerchantBIN	*Bank Identification Number for the processing of Merchant's transactions at the Acquirer*
MerchantID	*Merchant ID assigned by Acquirer*
AcquirerBIN	*The Bank Identification Number of this Acquirer*
AcquirerBusinessID	*The Business Identification Number of this Acquirer*

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

RequestType	*Enumerated code that indicates:* • *whether a Cardholder, Merchant, or Payment Gateway is issuing the request, and* • *whether it is for a new or renewed signature and/or encryption certificate.*

Table 72: RequestType

```
RequestType ::= ENUMERATED {  -- Indicates requestor and type of request
   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

EE	**< C, M, P >** *For Cardholder, Merchant, or Payment Gateway, respectively. EE is short for “End Entity,” also known as requester.*

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

CA	**< CCA, MCA, PCA >** *For EE = C, M, P, respectively*

Table 74: CA (CA types)

```
2948 CA ::= ENTITY-IDENTIFIER  -- Certifying Authority
```

BrandCRLIdentifier

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.

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

Table 75: BrandCRLIdentifier

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

Continued on next page

BrandCRLIdentifier, continued

BrandCRLIdentifier (continued)

```
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
}

BrandID ::= SETString { ub-BrandID }

CRLIdentifierSeq ::= SEQUENCE OF CRLIdentifier

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

CRLNumber ::= INTEGER (0..MAX)
```

PANData0

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.

PANData0	**{PAN, CardExpiry, CardSecret, EXNonce}**
PAN	*Primary Account Number; typically, the account number on the card*
CardExpiry	*Expiration date on the card*
CardSecret	*Cardholder's proposed half of the shared secret,* **PANSecret**. *Note: this value is saved for use in generating* **TransStain** *(see* "PIHead" *on page 37).*
EXNonce	*A fresh nonce to foil dictionary attacks on* **PANData0**

Table 76: PANData0

```
PANData0 ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   cardSecret  Secret,
   exNonce     Nonce
}

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

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

AcctData

AcctData

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

AcctData	**{AcctIdentification, EXNonce}**
AcctIdentification	*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.*
EXNonce	*A fresh nonce to foil dictionary attacks on* **AcctIdentification**

Table 77: AcctData

```
AcctData ::= SEQUENCE {
   acctIdentification  AcctIdentification,
   exNonce             Nonce
}

AcctIdentification ::= VisibleString (SIZE(ub-acctIdentification))
```

RegFormOrReferral

RegFormOrReferral

RegFormOrReferral	**< RegFormData, ReferralData >**
RegFormData	**{[RegTemplate], PolicyText}**
ReferralData	**{[Reason], [ReferralURLSeq]}**
RegTemplate	**{RegFormID, [BrandLogoURL], [CardLogoURL], [RegFieldSeq]}**
PolicyText	*Statement to be displayed along with* **RegTemplate** *on requester's system*
Reason	*Statement concerning request to be displayed on requester's system*
ReferralURLSeq	**{ReferralURL +}** *Optional URLs pointing to referral information, listed in the order of relevance*
RegFormID	*CA-assigned identifier*
BrandLogoURL	*The URL for the payment card brand logo*
CardLogoURL	*The URL for the financial institution logo*
RegFieldSeq	**{RegField +}**
ReferralURL	*Uniform Resource Locator of alternate CA for processing of certificate requests for this entity.*
RegField	**{[FieldId], FieldName, [FieldDesc], [FieldLen], FieldRequired, FieldInvisible}**
FieldID	*See Object Identifiers appendix in SET Book 2: Programmer's Guide*
FieldName	*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**
FieldDesc	*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.*
FieldLen	*Maximum length of field*
FieldRequired	*Boolean indicating whether data is required (either entered by the Cardholder or, if the field is invisible, populated by the application)*
FieldInvisible	*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.*

Table 78: RegFormOrReferral

Continued on next page

RegFormOrReferral, continued

RegFormOrReferral (continued)

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

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

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

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

PolicyText ::= SETString { ub-PolicyText }

Reason ::= SETString { ub-Reason }

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

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

ReferralURL ::= URL

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
}

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:

Topic	Page
Certificate Initialization Pair - Cardholder	158
Certificate Initialization Pair - Merchant or Payment Gateway	160
Registration Form Pair - Cardholder Only	165
Certificate Request Pair	169
Certificate Inquiry Pair	177

Certificate Initialization Pair - Cardholder

CardCInitReq

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

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

CardCInitRes

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

Table 80: CardCInitRes

```
CardCInitRes ::= S { CA, CardCInitResTBS }

CardCInitResTBS ::= SEQUENCE {
   rrpid              RRPID,
   lid-EE             LocalID,
   chall-EE           Challenge,
   lid-CA             LocalID  OPTIONAL,
   caeThumb           [0] EXPLICIT CertThumb,
   brandCRLIdentifier [1] EXPLICIT BrandCRLIdentifier  OPTIONAL,
   thumbs             [2] EXPLICIT Thumbs  OPTIONAL
}

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

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

Thumbs ::= SEQUENCE {
   digestAlgorithm   AlgorithmIdentifier {{DigestAlgorithms}},
   certThumbs        [0] EXPLICIT Digests  OPTIONAL,
   crlThumbs         [1] EXPLICIT Digests  OPTIONAL,
   brandCRLIdThumbs  [2] EXPLICIT Digests  OPTIONAL
}
```

Certificate Initialization Pair - Merchant or Payment Gateway

Me-AqCInitReq

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

Table 81: Me-AqCInitReq

```
Me-AqCInitReq ::= SEQUENCE {
   rrpid        RRPID,
   lid-EE       LocalID,
   chall-EE     Challenge,
   requestType  RequestType,
   idData       IDData,
   brandID      BrandID,
   language     Language,
   thumbs       [0] EXPLICIT Thumbs  OPTIONAL
}
```

Continued on next page

Certificate Initialization Pair - Merchant or Payment Gateway, continued

Me-AqCInitReq (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 }

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

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

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    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 }
```

Continued on next page

Certificate Initialization Pair - Merchant or Payment Gateway, continued

Me-AqCInitRes (continued)

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

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

Thumbs ::= SEQUENCE {
   digestAlgorithm   AlgorithmIdentifier {{DigestAlgorithms}},
   certThumbs        [0] EXPLICIT Digests  OPTIONAL,
   crlThumbs         [1] EXPLICIT Digests  OPTIONAL,
   brandCRLIdThumbs  [2] EXPLICIT Digests  OPTIONAL
}
```

Registration Form Pair - Cardholder Only

RegFormReq

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

Table 83: RegFormReq

```
RegFormReq ::= EXH { CA, RegFormReqData, PANOnly }

RegFormReqData ::= SEQUENCE {
   rrpid        RRPID,
   lid-EE       LocalID,
   chall-EE2    Challenge,
   lid-CA       [0] LocalID  OPTIONAL,
   requestType  RequestType,
   language     Language,
   thumbs       [1] EXPLICIT Thumbs  OPTIONAL
}

PANOnly ::= SEQUENCE {
   pan      PAN,
   exNonce  Nonce
}
```

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

RegFormRes

RegFormRes	**S(CA, RegFormResTBS)**
RegFormResTBS	**{RRPID, LID-EE, Chall-EE2, [LID-CA], Chall-CA, [CAEThumb], RequestType, RegFormOrReferral, [BrandCRLIdentifier], [Thumbs]}**
RRPID	*Request/response pair ID*
LID-EE	*Copied from* **RegFormReq**
Chall-EE2	*Copied from* **RegFormReq**
LID-CA	*Local ID; generated by and for CA system (new value may be specified)*
Chall-CA	*CA's challenge to requester's signature freshness*
CAEThumb	*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.*
RequestType	*See page 149*
RegFormOrReferral	*See page 155.*
BrandCRLIdentifier	*See page 151.*
Thumbs	*Copied from* **RegFormReq**

Table 84: RegFormRes

```
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,
   formOrReferal      RegFormOrReferral,
   brandCRLIdentifier  [2] EXPLICIT BrandCRLIdentifier  OPTIONAL,
   thumbs             [3] EXPLICIT Thumbs  OPTIONAL
}
```

Continued on next page

Registration Form Pair - Cardholder Only, continued

RegFormRes (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 }

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

191 BrandCRLIdentifier ::= SIGNED {
192    EncodedBrandCRLID
193 } ( CONSTRAINED 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 }
```

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)

RegFormID	*CA-assigned identifier*
RegForm	**{RegFormItems +}** *The field names copied from* **RegFormRes** *or* **Me-AqCInitRes***, now accompanied by values filled in by EE's implementation*
CABackKeyData	**{CAAlgId, CAKey}**
PublicKeySorE	**{[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.*
EEThumb	*Thumbprint of entity key-encryption certificate that is being renewed.*
Thumbs	*Lists of Certificate (including Root), CRL, and BrandCRLIdentifier currently held by EE*
PANData0	*See page 153.*
AcctData	*See page 154.*
RegFormItems	**{FieldName, FieldValue}**
CAAlgId	*Symmetric key algorithm identifier*
CAKey	*Secret key corresponding to the algorithm identifier*
PublicKeyS	*Proposed public signature key to certify*
PublicKeyE	*Proposed public encryption key to certify*
FieldName	*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**
FieldValue	*Values entered by EE*

Table 85: CertReq, continued

Continued on next page

Certificate Request Pair, continued

CertReq (continued)

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

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
}

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

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

RequestType ::= ENUMERATED {  -- Indicates requestor and type of request
   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)
}
```

Continued on next page

Certificate Request Pair, continued

CertReq (continued)

```
404 IDData ::= 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

CertRes

CertRes	**< S(CA, CertResData), EncK(CABackKeyData, CA, CertResData) >** *The* **EncK** *version of this message is only needed if the optional* **CAMsg** *component is included in the* **CertRes** *and it is only used if* **CaBackKeyData** is included in the **CertReq**.
CertResData	**{RRPID, LID-EE, Chall-EE3, LID-CA, CertStatus, [CertThumbs], [BrandCRLIdentifier], [Thumbs]}**
CABackKeyData	*Copied from* **CertReq**
RRPID	*Request/response pair ID*
LID-EE	*Copied from prior* **CertReq**
Chall-EE3	*Copied from* **CertReq**. *Requester checks for match with remembered value.*
LID-CA	*Copied from* **CertReq**. *If not present in the* **CertReq**, *new values are assigned.*
CertStatus	**{CertStatusCode, [Nonce-CCA], [EEMessage], [CaMsg], [FailedItemSeq]}**
CertThumbs	*If request is complete, the thumbprints of the enclosed signature and or encryption certificates*
BrandCRLIdentifier	*See page 151.*
Thumbs	*Copied from* **CertReq**

Continued on next page

Certificate Request Pair, continued

CertRes (continued)

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

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

Certificate Request Pair, continued

CertRes (continued)

```
CAMsg ::= SEQUENCE {
   cardLogoURL    [0] URL  OPTIONAL,
   brandLogoURL   [1] URL  OPTIONAL,
   cardCurrency   [2] Currency  OPTIONAL,
   cardholderMsg  [3] EXPLICIT
                           SETString { ub-cardholderMsg }  OPTIONAL
}

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

FailedItem ::= SEQUENCE {
   itemNumber  INTEGER (1..50),
   itemReason  SETString { ub-Reason }
}
```

Certificate Inquiry Pair

CertInqReq

CertInqReq	**S(EE, CertInqReqTBS)**
CertInqReqTBS	**{RRPID, LID-EE, Chall-EE3, LID-CA}**
RRPID	*Request/response pair identifier.*
LID-EE	*Copied from* **CertRes**
Chall-EE3	*EE's challenge to CA's signature freshness*
LID-CA	*Copied from* **CertRes**

Table 86: CertInqReq

```
CertInqReq ::= S { EE, CertInqReqTBS }

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

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

CertInqRes

CertInqRes	*Identical to a* **CertRes***; see page 173.*

Table 87: CertInqRes

```
CertInqRes ::= CertRes
```

Part II
ASN.1 Code

SET modules

The following modules are defined in SET.

Module	Starting Line Number	Page Number
SetAttribute	2989	231
SetCertificate	1980	213
SetCertificateExtensions	2046	214
SetCertMsgs	362	185
SetCRL	2612	224
SetMessage	4	179
SetPayMsgs	721	191
SetPKCS7Plus	2660	225

```
 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 {}
21       FROM SetCertificate
22
23    EXTENSION, Extensions, ub-cityName, ub-postalCode, ub-stateProvince
24       FROM SetCertificateExtensions
25
26    CardCInitReq, CardCInitRes, CertInqReq, CertInqRes, CertReq, CertRes,
27    Me-AqCInitReq, Me-AqCInitRes, RegFormReq, RegFormRes
28       FROM SetCertMsgs
29
30    AuthReq, AuthRes, AuthRevReq, AuthRevRes, BatchAdminReq,
31    BatchAdminRes, CapReq, CapRes, CapRevReq, CapRevRes, CredReq,
32    CredRes, CredRevReq, CredRevRes, InqReq, InqRes, PCertReq,
33    PCertRes, PInitReq, PInitRes, PReq, PRes
34       FROM SetPayMsgs
35
```

```
   CA, ContentEncryptionAlgorithms, Digest, DigestAlgorithms, Digests, EE, S {}
      FROM SetPKCS7Plus

   ub-phone
      FROM SetMarketData;


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

-- An information object set is defined for each extensible PDU
--
-- Note: each of these information object sets uses the extension
-- marker (...) to allow vendors to add supported extensions to
-- their local copy of the ASN.1. Extensions added by vendors
-- should appear after the extension marker.

MWExtensionsIOS EXTENSION ::= { ... }

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
}

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

MESSAGE ::= TYPE-IDENTIFIER           -- ISO/IEC 8824-2:1995(E), Annex A

Message ::= CHOICE {

   purchaseInitRequest          [ 0] EXPLICIT PInitReq,
   purchaseInitResponse         [ 1] EXPLICIT PInitRes,

   purchaseRequest              [ 2] EXPLICIT PReq,
   purchaseResponse             [ 3] EXPLICIT PRes,

   inquiryRequest               [ 4] EXPLICIT InqReq,
   inquiryResponse              [ 5] EXPLICIT InqRes,

   authorizationRequest         [ 6] EXPLICIT AuthReq,
   authorizationResponse        [ 7] EXPLICIT AuthRes,

   authReversalRequest          [ 8] EXPLICIT AuthRevReq,
   authReversalResponse         [ 9] EXPLICIT AuthRevRes,

   captureRequest               [10] EXPLICIT CapReq,
   captureResponse              [11] EXPLICIT CapRes,

   captureReversalRequest       [12] EXPLICIT CapRevReq,
   captureReversalResponse      [13] EXPLICIT CapRevRes,

   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
}

-- Note: the parameter InfoObjectSet in the following definitions
-- allows a distinct information object set to be specified for
-- each PDU that can be extended thus permitting the organization
-- defining the extension to indicate where it intends for the
-- extension to appear.

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
 }

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

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)
 }

-- Brand CRL Identifiers

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
}

-- Distribution of BrandCRLIdentifier to CAs and payment gateways

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 {
   date     Date,
   timeInd  BOOLEAN DEFAULT FALSE
}

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 }

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

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

PANData ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   panSecret   Secret,
   exNonce     Nonce
}

PANData0 ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   cardSecret  Secret,
   exNonce     Nonce
}

PANToken ::= SEQUENCE {
   pan         PAN,
   cardExpiry  CardExpiry,
   exNonce     Nonce
}

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

Phone ::= SETString { ub-phone }

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

```

```
Secret ::= OCTET STRING (SIZE(20))

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

Thumbs ::= SEQUENCE {
   digestAlgorithm   AlgorithmIdentifier {{DigestAlgorithms}},
   certThumbs        [0] EXPLICIT Digests  OPTIONAL,
   crlThumbs         [1] EXPLICIT Digests  OPTIONAL,
   brandCRLIdThumbs  [2] EXPLICIT Digests  OPTIONAL
}

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

URL ::= VisibleString (SIZE(1..ub-URL))      -- Universal Resource Locator

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

-- Upper bounds of SETString{} types

ub-BrandID           INTEGER ::=  40
ub-MerchantID        INTEGER ::=  30
ub-SWIdent           INTEGER ::= 256
ub-acqBusinessID     INTEGER ::=  32
ub-locationID        INTEGER ::=  10
ub-paySysID          INTEGER ::=  64
ub-RFC1766-language  INTEGER ::=  35
ub-URL               INTEGER ::= 512

END
```

```
SetCertMsgs
  { joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 1}
      DEFINITIONS IMPLICIT TAGS ::= BEGIN

--
-- Types used in the SET Certificate Management Protocol messages.
--

-- EXPORTS All;

IMPORTS

   SETString {}, SignatureAlgorithms
      FROM SetAttribute

   SubjectPublicKeyInfo{}
      FROM SetCertificate

   BackKeyData, BIN, BrandCRLIdentifier, BrandID,
   CertThumb,Challenge, Currency, Date, Language, LocalID, MerchantID,
   Nonce, PAN, PANData0, RRPID, Thumbs, ub-acqBusinessID, URL
      FROM SetMessage

   CA, EE, Enc {}, EncK {}, EncX {}, EXH {}, KeyEncryptionAlgorithms, L {},
   S {}, SO {}
      FROM SetPKCS7Plus;


-- Certificate Management Payload Components

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

AcctData ::= SEQUENCE {
   acctIdentification  AcctIdentification,
   exNonce             Nonce
}

AcctIdentification ::= VisibleString (SIZE(ub-acctIdentification))

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

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

AcquirerID ::= SEQUENCE {
   acquirerBIN         BIN,
   acquirerBusinessID  AcquirerBusinessID  OPTIONAL
}

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

RequestType ::= ENUMERATED {  -- Indicates requestor and type of request
   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)
}

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 }

-- Certificate Initialization Pair - Cardholder

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

CardCInitRes ::= S { CA, CardCInitResTBS }

CardCInitResTBS ::= SEQUENCE {
   rrpid               RRPID,
   lid-EE              LocalID,
   chall-EE            Challenge,
   lid-CA              LocalID  OPTIONAL,
   caeThumb            [0] EXPLICIT CertThumb,
   brandCRLIdentifier  [1] EXPLICIT BrandCRLIdentifier  OPTIONAL,
   thumbs              [2] EXPLICIT Thumbs  OPTIONAL
}

-- Certificate Initialization Pair - Merchant or Payment Gateway

Me-AqCInitReq ::= SEQUENCE {
   rrpid        RRPID,
   lid-EE       LocalID,
   chall-EE     Challenge,
   requestType  RequestType,
   idData       IDData,
   brandID      BrandID,
   language     Language,
   thumbs       [0] EXPLICIT Thumbs  OPTIONAL
}

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

Me-AqCInitResTBS ::= SEQUENCE {
   rrpid               RRPID,
   lid-EE              LocalID,
   chall-EE            Challenge,
   lid-CA              [0] LocalID  OPTIONAL,
   chall-CA            Challenge,
   requestType         RequestType,
   regFormOrReferral   RegFormOrReferral,
   acctDataField       [1] RegField  OPTIONAL,
   caeThumb            [2] EXPLICIT CertThumb,
   brandCRLIdentifier  [3] EXPLICIT BrandCRLIdentifier  OPTIONAL,
   thumbs              [4] EXPLICIT Thumbs  OPTIONAL
}

-- Registration Form Pair - Cardholder Only

RegFormReq ::= EXH { CA, RegFormReqData, PANOnly }

-- Intermediate results of EXH
RegFormReqTBE ::= L { RegFormReqData, PANOnly }

RegFormReqData ::= SEQUENCE {
   rrpid        RRPID,
   lid-EE       LocalID,
   chall-EE2    Challenge,
   lid-CA       [0] LocalID  OPTIONAL,
   requestType  RequestType,
   language     Language,
   thumbs       [1] EXPLICIT Thumbs  OPTIONAL
}

PANOnly ::= SEQUENCE {
   pan      PAN,
   exNonce  Nonce
```

```
}

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,
   formOrReferal      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,
   fieldValue  FieldValue
}

FieldName ::= SETString { ub-FieldName }

FieldValue ::= CHOICE {
   setString    SETString { ub-FieldValue },
   octetString  OCTET STRING (SIZE(1..ub-FieldValue))
}

PublicKeySorE ::= SEQUENCE {
   publicKeyS  [0] EXPLICIT SubjectPublicKeyInfo{{SignatureAlgorithms}}
                                                              OPTIONAL,
   publicKeyE  [1] EXPLICIT SubjectPublicKeyInfo{{KeyEncryptionAlgorithms}}
                                                              OPTIONAL
}  --
   -- At least one component shall be present. A user may request a
   -- signature certificate, an encryption certificate, or both.
   --
   ( WITH COMPONENTS { ..., publicKeyS PRESENT } |
     WITH COMPONENTS { ..., publicKeyE PRESENT } )

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

-- Intermediate results of EncK
CertResTBE ::= S { CA, CertResData }

CertResData ::= SEQUENCE {
   rrpid              RRPID,
   lid-EE             LocalID,
   chall-EE3          Challenge,
   lid-CA             LocalID,
   certStatus         CertStatus,
   certThumbs         [0] EXPLICIT Thumbs  OPTIONAL,
   brandCRLIdentifier  [1] EXPLICIT BrandCRLIdentifier  OPTIONAL,
   thumbs             [2] EXPLICIT Thumbs  OPTIONAL
}

CertStatus ::= SEQUENCE {
   certStatusCode  CertStatusCode,
   nonceCCA        [0] Nonce  OPTIONAL,
   eeMessage       SETString { ub-eeMessage }  OPTIONAL,
   caMsg           [1] CAMsg  OPTIONAL,
   failedItemSeq   [2] FailedItemSeq  OPTIONAL
}

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

FailedItem ::= SEQUENCE {
   itemNumber  INTEGER (1..50),
   itemReason  SETString { ub-Reason }
}

CertStatusCode ::= ENUMERATED {          -- In-process status of CertReq
   requestComplete           (1),
   invalidLanguage           (2),
   invalidBIN                (3),
   sigValidationFail         (4),
   decryptionError           (5),
   requestInProgress         (6),
   rejectedByIssuer          (7),
   requestPended             (8),
   rejectedByAquirer         (9),
   regFormAnswerMalformed   (10),
   rejectedByCA             (11),
   unableToEncryptResponse  (12)
}

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


END
```

```
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,
   piRqExtensions  [2] MsgExtensions {{PIRqExtensionsIOS}}  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 ::= { ... }
```

```

-- Purchase Pair

PReq ::= CHOICE {
   pReqDualSigned  [0] EXPLICIT PReqDualSigned,
   pReqUnsigned    [1] EXPLICIT PReqUnsigned
}

-- Signed components used by a cardholder with a certificate

PReqDualSigned ::= SEQUENCE {
   piDualSigned  PIDualSigned,
   oiDualSigned  OIDualSigned
}

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

-- Intermediate results of EX
PIDualSignedTBE ::= L { PI-OILink, PANData }

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

OIDualSigned ::= L { OIData, PIData }

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

PI-TBS ::= SEQUENCE {
   hPIData  HPIData,
   hOIData  HOIData
}

HPIData ::= DD { PIData }                         -- PKCS#7 DigestedData

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

PI ::= CHOICE {
   piUnsigned    [0] EXPLICIT PIUnsigned,
   piDualSigned  [1] EXPLICIT PIDualSigned,
   authToken     [2] EXPLICIT AuthToken
}

PIData ::= SEQUENCE {
   piHead   PIHead,
   panData  PANData
}

PIHead ::= SEQUENCE {
   transIDs          TransIDs,
   inputs            Inputs,
   merchantID        MerchantID,
   installRecurData  [0] InstallRecurData  OPTIONAL,
   transStain        TransStain,
   swIdent           SWIdent,
   acqBackKeyData    [1] EXPLICIT BackKeyData  OPTIONAL,
   piExtensions      [2] MsgExtensions {{PIExtensionsIOS}} OPTIONAL
}

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
}

OIExtensionsIOS 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 }

```

```
-- Intermediate results of EXH
PIUnsignedTBE ::= L { PI-OILink, PANToken }

PRes ::= S { M, PResData }

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

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

PResPayload ::= SEQUENCE {
   completionCode  CompletionCode,
   results         Results  OPTIONAL,
   pRsExtensions   [0] MsgExtensions {{PRsExtensionsIOS}} OPTIONAL
}

PRsExtensionsIOS EXTENSION ::= { ... }

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

Results ::= SEQUENCE {
   acqCardMsg     [0] EXPLICIT AcqCardMsg  OPTIONAL,
   authStatus     [1] AuthStatus  OPTIONAL,
   capStatus      [2] CapStatus  OPTIONAL,
   credStatusSeq  [3] CreditStatusSeq  OPTIONAL
}

AuthStatus ::= SEQUENCE {
   authDate   Date,
   authCode   AuthCode,
   authRatio  FloatingPoint,
   currConv   [0] CurrConv  OPTIONAL
}

CapStatus ::= SEQUENCE {
   capDate   Date,
   capCode   CapCode,
   capRatio  FloatingPoint
}

CreditStatusSeq ::= SEQUENCE SIZE(1..MAX) OF CreditStatus

CreditStatus ::= SEQUENCE {
   creditDate   Date,
   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 {
   subsequentAuthInd   BOOLEAN DEFAULT FALSE,
   authReqAmt          CurrencyAmount,        -- May differ from PurchAmt
   avsData             [0] AVSData  OPTIONAL,
   specialProcessing   [1] SpecialProcessing  OPTIONAL,
   cardSuspect         [2] CardSuspect  OPTIONAL,
   requestCardTypeInd  BOOLEAN DEFAULT FALSE,
   installRecurData    [3] InstallRecurData  OPTIONAL,
   marketSpecAuthData  [4] EXPLICIT MarketSpecAuthData  OPTIONAL,
   merchData           MerchData,
   aRqExtensions       [5] MsgExtensions {{ARqExtensionsIOS}} OPTIONAL
}

ARqExtensionsIOS EXTENSION ::= { ... }

AVSData ::= SEQUENCE {
   streetAddress  SETString { ub-AVSData } OPTIONAL,
   location       Location
}

SpecialProcessing ::= ENUMERATED {
   directMarketing    (0),
   preferredCustomer  (1)
}

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

MerchData ::= SEQUENCE {
   merchCatCode  MerchCatCode  OPTIONAL,
   merchGroup    MerchGroup  OPTIONAL
}

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

MerchGroup ::= ENUMERATED {
   commercialTravel  (1),
   lodging           (2),
   automobileRental  (3),
   restaurant        (4),
   medical           (5),
   mailOrPhoneOrder  (6),
   riskyPurchase     (7),
   other             (8)
}

AuthRes ::= CHOICE {
   encB   [0] EXPLICIT EncB { P, M, AuthResData, AuthResBaggage },
   encBX  [1] EXPLICIT EncBX { P, M, AuthResData, AuthResBaggage, PANToken }
}

```

```
-- Intermediate results of EncB and EncBX
AuthResTBE ::= S { P, AuthResTBS }

AuthResTBEX ::= SEQUENCE {
   authResTBS  AuthResTBS,
   s           SO { P, AuthResTBSX }
}

AuthResTBS ::= L { AuthResData, AuthResBaggage}

AuthResTBSX ::= SEQUENCE {
   authResTBS  AuthResTBS,
   panToken    PANToken
}

AuthResData ::= SEQUENCE {
   authTags            AuthTags,
   brandCRLIdentifier  [0] EXPLICIT BrandCRLIdentifier  OPTIONAL,
   peThumb             [1] EXPLICIT CertThumb  OPTIONAL,
   authResPayload      AuthResPayload
}

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

AcqBackKey ::= BackKeyData

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

-- Intermediate result of EncK
AcqCardCodeMsgTBE ::= S { P, AcqCardCodeMsg }

AcqCardCodeMsg ::= SEQUENCE {
   acqCardCode     AcqCardCode,
   acqCardMsgData  AcqCardMsgData
}

AcqCardCode ::= ENUMERATED {
   messageOfDay         (0),
   accountInfo          (1),
   callCustomerService  (2)
}

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

AuthResPayload ::= SEQUENCE {
   authHeader     AuthHeader,
   capResPayload  CapResPayload  OPTIONAL,
   aRsExtensions  [0] MsgExtensions {{ARsExtensionsIOS}} OPTIONAL
}

```

```
ARsExtensionsIOS EXTENSION ::= { ... }

AuthHeader ::= SEQUENCE {
   authAmt       CurrencyAmount,
   authCode      AuthCode,
   responseData  ResponseData,
   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),
   platinum               ( 3),
   premier                ( 4),
   debit                  ( 5),
   pinBasedDebit          ( 6),
   atm                    ( 7),
   electronicOnly         ( 8),
   unspecifiedConsumer    ( 9),
   corporateTravel        (10),
   purchasing             (11),
   business               (12),
   unspecifiedCommercial  (13),
   privateLabel           (14),
   proprietary            (15)
}

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,
   authResPayload   [2] AuthResPayload  OPTIONAL,
   authNewAmt       CurrencyAmount,
   aRvRqExtensions  [3] MsgExtensions {{ARvRqExtensionsIOS}} OPTIONAL
}

ARvRqExtensionsIOS EXTENSION ::= { ... }

AuthRevReqBaggage ::= SEQUENCE {
   pi        PI,
   capToken  CapToken  OPTIONAL
}

AuthRevTags ::= SEQUENCE {
   authRevRRTags  AuthRevRRTags,
   authRetNum     AuthRetNum  OPTIONAL
}

AuthRevRRTags ::= RRTags

AuthRetNum ::= INTEGER (0..MAX)

AuthRevRes ::= CHOICE {
   encB  [0] EXPLICIT EncB { P, M, AuthRevResData, AuthRevResBaggage },
   enc   [1] EXPLICIT Enc { P, M, AuthRevResData }
}

-- Intermediate results of Enc and EncB
AuthRevResTBE ::= S { P, AuthRevResData }

AuthRevResTBEB ::= S { P, AuthRevResTBS }

AuthRevResTBS ::= L { AuthRevResData, AuthRevResBaggage }

AuthRevResBaggage ::= SEQUENCE {
   capTokenNew   CapToken  OPTIONAL,
   authTokenNew  AuthToken  OPTIONAL
}

AuthRevResData ::= SEQUENCE {
   authRevCode         AuthRevCode,
   authRevTags         AuthRevTags,
   brandCRLIdentifier  [0] EXPLICIT BrandCRLIdentifier  OPTIONAL,
   peThumb             [1] EXPLICIT CertThumb  OPTIONAL,
   authNewAmt          CurrencyAmount,                    -- May be zero
   authResDataNew      AuthResDataNew,
   aRvRsExtensions     [2] MsgExtensions {{ARvRsExtensionsIOS}} OPTIONAL
}

ARvRsExtensionsIOS EXTENSION ::= { ... }

AuthRevCode ::= ENUMERATED {
   approved            ( 0),
   unspecifiedFailure  ( 1),
   noReply             ( 2),
   amountError         ( 3),
   expiredCard         ( 4),
   invalidTransaction  ( 5),
   systemError         ( 6),
   missingCapToken     ( 7),
   invalidCapToken     ( 8),
   invalidAmount       ( 9)
}

AuthResDataNew ::= SEQUENCE {
   transIDs           TransIDs,
   authResPayloadNew  AuthResPayload  OPTIONAL      -- Contains new data
```

```
}

-- Capture Pair

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

-- Intermediate results of EncB and EncBX
CapReqTBE ::= S { M, CapReqTBS }

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

CapReqTBS ::= L { CapReqData, CapTokenSeq }

CapReqTBSX ::= SEQUENCE {
   capReqTBS  CapReqTBS,
   panToken   PANToken
}

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,
   authResPayload  [1] AuthResPayload  OPTIONAL,
   saleDetail      [2] SaleDetail  OPTIONAL,
   cPayExtensions  [3] MsgExtensions {{CPayExtensionsIOS}} OPTIONAL
}

CPayExtensionsIOS EXTENSION ::= { ... }

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

-- Intermediate results of Enc
CapResTBE ::= S { P, CapResData }
```

```

CapResData ::= SEQUENCE {
   capRRTags           CapRRTags,
   brandCRLIdentifier  [0] EXPLICIT BrandCRLIdentifier  OPTIONAL,
   peThumb             [1] EXPLICIT CertThumb  OPTIONAL,
   batchStatusSeq      [2] BatchStatusSeq  OPTIONAL,
   capResItemSeq       CapResItemSeq,
   cRsExtensions       [3] MsgExtensions {{CRsExtensionsIOS}} OPTIONAL
}

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,
   cRsPayExtensions  [2] MsgExtensions {{CRsPayExtensionsIOS}} 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,
   cRvRsExtensions         [3] MsgExtensions {{CRvRsExtensionsIOS}} OPTIONAL
}

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,
   cRvRsPayExtensions     [2] MsgExtensions {{CRvRsPayExtensionsIOS}} 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 ::= [1] EXPLICIT CapRevOrCredResData

-- Credit Reversal Pair

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 }

-- Intermediate results of Enc
CredRevResTBE ::= S { P, CredRevResData }

CredRevResData ::= [2] EXPLICIT CapRevOrCredResData

-- Payment Gateway Certificate Request Pair

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,
   pCertResItemSeq        PCertResItemSeq,
   brandCRLIdentifierSeq  [0] BrandCRLIdentifierSeq  OPTIONAL,
   pcRsExtensions         [1] MsgExtensions {{PCRsExtensionsIOS}}  OPTIONAL
}

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 }

-- Intermediate results of Enc
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
}

```

```
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
}
```

```
CloseStatus ::= ENUMERATED {
   closedbyMerchant  (0),
   closedbyAcquirer  (1)
}

BatchStatusSeq ::= SEQUENCE OF BatchStatus

BatchStatus ::= SEQUENCE {
   openDateTime     Date,
   closedWhen       [0] ClosedWhen  OPTIONAL,
   batchDetails     BatchDetails,
   batchExtensions  [1] MsgExtensions {{BSExtensionsIOS}} OPTIONAL
}

BSExtensionsIOS EXTENSION ::= { ... }

BatchDetails ::= SEQUENCE {
   batchTotals           BatchTotals,
   brandBatchDetailsSeq  BrandBatchDetailsSeq  OPTIONAL
}

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

BrandBatchDetails ::= SEQUENCE {
   brandID      BrandID,
   batchTotals  BatchTotals
}

BatchTotals ::= SEQUENCE {
   transactionCountCredit     INTEGER (0..MAX),
   transactionTotalAmtCredit  CurrencyAmount,
   transactionCountDebit      INTEGER (0..MAX),
   transactionTotalAmtDebit   CurrencyAmount,
   batchTotalExtensions       [0] MsgExtensions {{BTExtensionsIOS}} OPTIONAL
}

BTExtensionsIOS EXTENSION ::= { ... }

TransactionDetailSeq ::= SEQUENCE OF TransactionDetail

TransactionDetail ::= SEQUENCE {
   transIDs            TransIDs,
   authRRPID           RRPID,
   brandID             BrandID,
   batchSequenceNum    BatchSequenceNum,
   reimbursementID     ReimbursementID  OPTIONAL,
   transactionAmt      CurrencyAmount,
   transactionAmtType  AmountType,
   transactionStatus   [0] TransactionStatus  OPTIONAL,
   transExtensions     [1] MsgExtensions {{TransExtensionsIOS}} OPTIONAL
}

TransExtensionsIOS EXTENSION ::= { ... }

AmountType ::= ENUMERATED {
   credit  (0),
   debit   (1)
}
```

```

TransactionStatus ::= ENUMERATED {
   success             (0),
   unspecifiedFailure  (1)
}

ReimbursementID ::= ENUMERATED {
   unspecified     (0),
   standard        (1),
   keyEntered      (2),
   electronic      (3),
   additionalData  (4),
   enhancedData    (5),
   marketSpecific  (6)
}

-- Payment Message Components

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

-- Intermediate results of EncX
AuthTokenTBE ::= SEQUENCE {
   authTokenData  AuthTokenData,
   s              SO { P1, AuthTokenTBS }
}

AuthTokenTBS ::= SEQUENCE {
   authTokenData  AuthTokenData,
   panToken       PANToken
}

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
}

BatchID ::= INTEGER (0..MAX)

BatchSequenceNum ::= INTEGER (1..MAX)

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

-- Intermediate results of Enc and EncX
CapTokenTBE ::= S { P1, CapTokenData }

CapTokenTBEX ::= SEQUENCE {
   capTokenData  CapTokenData,
   s             SO { P1, CapTokenTBS }
}

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,
   transport    [2] MarketTransportAuth
}

MarketSpecCapData ::= CHOICE {
   auto-rental  [0] MarketAutoCap,
   hotel        [1] MarketHotelCap,
   transport    [2] MarketTransportCap
}

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,
   marketSpecSaleData       [ 5] MarketSpecSaleData  OPTIONAL,
   commercialCardData       [ 6] CommercialCardData  OPTIONAL,
   orderSummary             [ 7] EXPLICIT SETString { ub-summary }  OPTIONAL,
   customerReferenceNumber  [ 8] EXPLICIT SETString { ub-reference }  OPTIONAL,
   customerServicePhone     [ 9] EXPLICIT Phone  OPTIONAL,
   okToPrintPhoneInd        [10] BOOLEAN DEFAULT TRUE,
   saleExtensions           [11] MsgExtensions {{SaleExtensionsIOS}}  OPTIONAL
}

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

-- Upper bound of SETString{} type

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

END
```

```
SetCertificate
  { joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 3 }
      DEFINITIONS EXPLICIT TAGS ::= BEGIN

--
-- This module defines types for CRL and X.509v3 certificate support.
--

-- EXPORTS All;

IMPORTS

   ALGORITHM-IDENTIFIER, AlgorithmIdentifier {}, Name,
   SignatureAlgorithms, SupportedAlgorithms
      FROM SetAttribute

   Extensions
      FROM SetCertificateExtensions;
```

```


UnsignedCertificate ::= SEQUENCE {
   version              [0] CertificateVersion,
   serialNumber         CertificateSerialNumber,
   signature            AlgorithmIdentifier {{SignatureAlgorithms}},
   issuer               Name,
   validity             Validity,
   subject              Name,
   subjectPublicKeyInfo  SubjectPublicKeyInfo{{SupportedAlgorithms}},
   issuerUniqueID       [1] IMPLICIT UniqueIdentifier OPTIONAL,
   subjectUniqueID      [2] IMPLICIT UniqueIdentifier OPTIONAL,
   extensions           [3] Extensions        -- Required for SET usage
}

CertificateVersion ::= INTEGER { ver3(2) } ( ver3 )

CertificateSerialNumber ::= INTEGER

-- Compute the encrypted hash of this value if issuing a certificate,
-- or recompute the issuer's signature on this value if validating a
-- certificate.
--
EncodedCertificate ::= TYPE-IDENTIFIER.&Type (UnsignedCertificate)

Certificate::=   SIGNED {
   EncodedCertificate
} ( CONSTRAINED BY { -- Verify Or Sign Certificate -- } )

SIGNED { ToBeSigned } ::= SEQUENCE {
   toBeSigned  ToBeSigned,
   algorithm   AlgorithmIdentifier {{SignatureAlgorithms}},
   signature   BIT STRING
}

Validity ::= SEQUENCE {
   notBefore  UTCTime,      -- Not valid before this date
   notAfter   UTCTime       -- Not valid after this date
}

UniqueIdentifier ::= BIT STRING          -- Not used in the SET protocol

SubjectPublicKeyInfo {ALGORITHM-IDENTIFIER:Algorithms} ::= SEQUENCE {
   algorithm         AlgorithmIdentifier {{Algorithms}},
   subjectPublicKey  BIT STRING
}

END
```

```
SetCertificateExtensions
  { joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 4 }
      DEFINITIONS IMPLICIT TAGS ::= BEGIN

--
-- Defines X.509 Version 3 certificate extensions.
--
```

```

-- 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 ::= {                   -- Information Object Set
   --
   -- 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 {
   extnID     EXTENSION.&id({ExtensionSet}),
   critical   EXTENSION.&critical({ExtensionSet}{@extnID}) DEFAULT FALSE,
   extnValue  OCTET STRING -- DER representation of &ExtenType extension
                           -- object for the object identified by extnID
}

-- Key and policy information extensions --

authorityKeyIdentifier EXTENSION ::= {
   SYNTAX         AuthorityKeyIdentifier
   IDENTIFIED BY id-ce-authorityKeyIdentifier
}

AuthorityKeyIdentifier ::= SEQUENCE {
   keyIdentifier              [0] KeyIdentifier  OPTIONAL,
   authorityCertIssuer        [1] GeneralNames  OPTIONAL,
   authorityCertSerialNumber  [2] CertificateSerialNumber  OPTIONAL
} ( WITH COMPONENTS {  keyIdentifier ABSENT,
      authorityCertIssuer PRESENT, authorityCertSerialNumber PRESENT } )

KeyIdentifier ::= OCTET STRING

keyUsage EXTENSION ::= {
   SYNTAX         KeyUsage
   CRITICAL       TRUE
   IDENTIFIED BY id-ce-keyUsage
}

KeyUsage ::= BIT STRING {
   digitalSignature  (0),
   nonRepudiation    (1),
   keyEncipherment   (2),
   dataEncipherment  (3),
   keyAgreement      (4),
   keyCertSign       (5),              -- For use in CA-certificates only
   cRLSign           (6)               -- For use in CA-certificates only
}

privateKeyUsagePeriod EXTENSION ::= {
   SYNTAX         PrivateKeyUsagePeriod
   IDENTIFIED BY id-ce-privateKeyUsagePeriod
}

PrivateKeyUsagePeriod ::= SEQUENCE {
   notBefore  [0] GeneralizedTime  OPTIONAL,
   notAfter   [1] GeneralizedTime  OPTIONAL
} ( WITH COMPONENTS { ..., notBefore PRESENT } |
    WITH COMPONENTS { ..., notAfter  PRESENT } )

certificatePolicies EXTENSION ::= {
   SYNTAX         CertificatePoliciesSyntax
   CRITICAL       TRUE
   IDENTIFIED BY id-ce-certificatePolicies
}

CertificatePoliciesSyntax ::= SEQUENCE SIZE(1..MAX) OF PolicyInformation

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
   IDENTIFIED BY id-ce-subjectAltName
}

GeneralNames ::= SEQUENCE SIZE(1..MAX) OF GeneralName

GeneralName ::= CHOICE {
   directoryName              [4] EXPLICIT Name,
   uniformResourceIdentifier  [6] IA5String,
   registeredID               [8] OBJECT IDENTIFIER
   -- Other choices defined in X.509 not used by SET
}

issuerAltName EXTENSION ::= {
   SYNTAX         GeneralNames
   IDENTIFIED BY id-ce-issuerAltName
}

-- Certification path constraints extensions --

basicConstraints EXTENSION ::= {
   SYNTAX         BasicConstraintsSyntax
   CRITICAL       TRUE
   IDENTIFIED BY id-ce-basicConstraints
}

BasicConstraintsSyntax ::= SEQUENCE {
   cA                 BOOLEAN  DEFAULT FALSE,
   pathLenConstraint  INTEGER (0..MAX)  OPTIONAL
}

-- Basic CRL extensions --

cRLNumber EXTENSION ::= {                        -- For use in CRLs only
   SYNTAX         CRLNumber
   IDENTIFIED BY id-ce-cRLNumber
}

CRLNumber ::= INTEGER (0..MAX)

-- Set protocol private extensions --

hashedRootKey EXTENSION ::= {               -- Only in root certificates
   SYNTAX         HashedRootKeySyntax
   CRITICAL       TRUE
   IDENTIFIED BY id-set-hashedRootKey
}

HashedRootKeySyntax ::= RootKeyThumb

RootKeyThumb ::= SEQUENCE {
   rootKeyThumbprint  DD { SubjectPublicKeyInfo{{SupportedAlgorithms}} }
}

certificateType EXTENSION ::= {
   SYNTAX         CertificateTypeSyntax
   CRITICAL       TRUE
   IDENTIFIED BY id-set-certificateType
}
```

```

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) }

-- Object identifiers assigned under id-set arc

OID ::= OBJECT IDENTIFIER

id-set-contentType                         OID ::= { id-set 0 }
id-set-msgExt                              OID ::= { id-set 1 }
id-set-field                               OID ::= { id-set 2 }
id-set-attribute                           OID ::= { id-set 3 }
id-set-algorithm                           OID ::= { id-set 4 }
id-set-policy                              OID ::= { id-set 5 }
id-set-module                              OID ::= { id-set 6 }
id-set-certExt                             OID ::= { id-set 7 }
id-set-brand                               OID ::= { id-set 8 }
id-set-vendor                              OID ::= { id-set 9 }
id-set-national                            OID ::= { id-set 10 }

   -- Content type
id-set-content-PANData                     OID ::= { id-set-contentType 0 }
id-set-content-PANToken                    OID ::= { id-set-contentType 1 }
id-set-content-PANOnly                     OID ::= { id-set-contentType 2 }
id-set-content-OIData                      OID ::= { id-set-contentType 3 }
id-set-content-PI                          OID ::= { id-set-contentType 4 }
id-set-content-PIData                      OID ::= { id-set-contentType 5 }
id-set-content-PIDataUnsigned              OID ::= { id-set-contentType 6 }
id-set-content-HODInput                    OID ::= { id-set-contentType 7 }
id-set-content-AuthResBaggage              OID ::= { id-set-contentType 8 }
id-set-content-AuthRevReqBaggage           OID ::= { id-set-contentType 9 }
id-set-content-AuthRevResBaggage           OID ::= { id-set-contentType 10 }
id-set-content-CapTokenSeq                 OID ::= { id-set-contentType 11 }
id-set-content-PInitResData                OID ::= { id-set-contentType 12 }
```

```
id-set-content-PI-TBS                    OID ::= { id-set-contentType 13 }
id-set-content-PResData                  OID ::= { id-set-contentType 14 }
id-set-content-InqReqData                OID ::= { id-set-contentType 15 }
id-set-content-AuthReqTBS                OID ::= { id-set-contentType 16 }
id-set-content-AuthResTBS                OID ::= { id-set-contentType 17 }
id-set-content-AuthResTBSX               OID ::= { id-set-contentType 18 }
id-set-content-AuthTokenTBS              OID ::= { id-set-contentType 19 }
id-set-content-CapTokenData              OID ::= { id-set-contentType 20 }
id-set-content-CapTokenTBS               OID ::= { id-set-contentType 21 }
id-set-content-AcqCardCodeMsg            OID ::= { id-set-contentType 22 }
id-set-content-AuthRevReqTBS             OID ::= { id-set-contentType 23 }
id-set-content-AuthRevResData            OID ::= { id-set-contentType 24 }
id-set-content-AuthRevResTBS             OID ::= { id-set-contentType 25 }
id-set-content-CapReqTBS                 OID ::= { id-set-contentType 26 }
id-set-content-CapReqTBSX                OID ::= { id-set-contentType 27 }
id-set-content-CapResData                OID ::= { id-set-contentType 28 }
id-set-content-CapRevReqTBS              OID ::= { id-set-contentType 29 }
id-set-content-CapRevReqTBSX             OID ::= { id-set-contentType 30 }
id-set-content-CapRevResData             OID ::= { id-set-contentType 31 }
id-set-content-CredReqTBS                OID ::= { id-set-contentType 32 }
id-set-content-CredReqTBSX               OID ::= { id-set-contentType 33 }
id-set-content-CredResData               OID ::= { id-set-contentType 34 }
id-set-content-CredRevReqTBS             OID ::= { id-set-contentType 35 }
id-set-content-CredRevReqTBSX            OID ::= { id-set-contentType 36 }
id-set-content-CredRevResData            OID ::= { id-set-contentType 37 }
id-set-content-PCertReqData              OID ::= { id-set-contentType 38 }
id-set-content-PCertResTBS               OID ::= { id-set-contentType 39 }
id-set-content-BatchAdminReqData         OID ::= { id-set-contentType 40 }
id-set-content-BatchAdminResData         OID ::= { id-set-contentType 41 }
id-set-content-CardCInitResTBS           OID ::= { id-set-contentType 42 }
id-set-content-Me-AqCInitResTBS          OID ::= { id-set-contentType 43 }
id-set-content-RegFormResTBS             OID ::= { id-set-contentType 44 }
id-set-content-CertReqData               OID ::= { id-set-contentType 45 }
id-set-content-CertReqTBS                OID ::= { id-set-contentType 46 }
id-set-content-CertResData               OID ::= { id-set-contentType 47 }
id-set-content-CertInqReqTBS             OID ::= { id-set-contentType 48 }
id-set-content-ErrorTBS                  OID ::= { id-set-contentType 49 }
id-set-content-PIDualSignedTBE           OID ::= { id-set-contentType 50 }
id-set-content-PIUnsignedTBE             OID ::= { id-set-contentType 51 }
id-set-content-AuthReqTBE                OID ::= { id-set-contentType 52 }
id-set-content-AuthResTBE                OID ::= { id-set-contentType 53 }
id-set-content-AuthResTBEX               OID ::= { id-set-contentType 54 }
id-set-content-AuthTokenTBE              OID ::= { id-set-contentType 55 }
id-set-content-CapTokenTBE               OID ::= { id-set-contentType 56 }
id-set-content-CapTokenTBEX              OID ::= { id-set-contentType 57 }
id-set-content-AcqCardCodeMsgTBE         OID ::= { id-set-contentType 58 }
id-set-content-AuthRevReqTBE             OID ::= { id-set-contentType 59 }
id-set-content-AuthRevResTBE             OID ::= { id-set-contentType 60 }
id-set-content-AuthRevResTBEB            OID ::= { id-set-contentType 61 }
id-set-content-CapReqTBE                 OID ::= { id-set-contentType 62 }
id-set-content-CapReqTBEX                OID ::= { id-set-contentType 63 }
id-set-content-CapResTBE                 OID ::= { id-set-contentType 64 }
id-set-content-CapRevReqTBE              OID ::= { id-set-contentType 65 }
id-set-content-CapRevReqTBEX             OID ::= { id-set-contentType 66 }
id-set-content-CapRevResTBE              OID ::= { id-set-contentType 67 }
id-set-content-CredReqTBE                OID ::= { id-set-contentType 68 }
id-set-content-CredReqTBEX               OID ::= { id-set-contentType 69 }
id-set-content-CredResTBE                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-CredRevResTBE            OID ::= { id-set-contentType 73 }
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 }
```

```
                         -- contact: david.armes@aexp.com
id-set-JCB               OID ::= { id-set-brand 35 }
                         -- contact: ohashi@cp.jcb.co.jp
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@novusnet.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: jeanne.gorman@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: asn1@mindspring.com
id-set-Certicom          OID ::= { id-set-vendor 11 }
                         -- contact: sshannon@certicom.ca
id-set-OSS               OID ::= { id-set-vendor 12 }
                         -- contact: baos@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: mortimer@entrust.com
id-set-VIAnet            OID ::= { id-set-vendor 24 }
                         -- contact: via.net@mail.eunet.pt
id-set-III               OID ::= { id-set-vendor 25 }
                         -- contact: wu@iii.org.tw
id-set-OpenMarket        OID ::= { id-set-vendor 26 }
                         -- contact: treese@OpenMarket.com
id-set-Lexem             OID ::= { id-set-vendor 27 }
                         -- contact: lje@lexem.fr
id-set-Intertrader       OID ::= { id-set-vendor 28 }
                         -- contact: rachel@intertrader.com
id-set-Persimmon         OID ::= { id-set-vendor 29 }
                         -- contact: carol.smith@persimmon.com
id-set-NABLE             OID ::= { id-set-vendor 30 }
                         -- contact: tony@nabletech.com
id-set-espace-net        OID ::= { id-set-vendor 31 }
                         -- contact: fm@well.com
id-set-Hitachi           OID ::= { id-set-vendor 32 }
                         -- contact: horimai@iabs.hitachi.co.jp
id-set-Microsoft         OID ::= { id-set-vendor 33 }
                         -- contact: rickj@microsoft.com
id-set-NEC               OID ::= { id-set-vendor 34 }
                         -- contact: nakata@mms.mt.nec.co.jp
id-set-Mitsubishi        OID ::= { id-set-vendor 35 }
                         -- contact: yoshitake@iss.isl.melco.co.jp
id-set-NCR               OID ::= { id-set-vendor 36 }
                         -- contact: Julian.Inza@spain.ncr.com
id-set-e-COMM            OID ::= { id-set-vendor 37 }
                         -- contact: 101643.426@compuserve.com
id-set-Gemplus           OID ::= { id-set-vendor 38 }
                         -- contact: florent.neu@ccmail.edt.fr

-- National markets: The value following id-set-national corresponds
-- to ISO-3166 numeric codes
id-set-Japan             OID ::= { id-set-national 392 }

END
```

```
SetCRL
  { joint-iso-itu-t(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.
--

-- EXPORTS All;

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
   (WITH COMPONENTS { ..., authenticatedAttributes   PRESENT,
                      unauthenticatedAttributes ABSENT })

SignerInfo ::= SEQUENCE {
   siVersion                  INTEGER { siVer2(2) } (siVer2),
   issuerAndSerialNumber      IssuerAndSerialNumber,
   digestAlgorithm            AlgorithmIdentifier {{DigestAlgorithms}},
   authenticatedAttributes    [2] EXPLICIT
                                  AttributeSeq {{Authenticated}}  OPTIONAL,
   digestEncryptionAlgorithm  AlgorithmIdentifier {{DigestEncryptionAlgorithms}},
   encryptedDigest            EncryptedDigest,
   unauthenticatedAttributes  [3] EXPLICIT AttributeSeq {{...}}  OPTIONAL
}

Authenticated ATTRIBUTE ::={
   { WITH SYNTAX ContentType   ID contentType   } |
   { 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-desCDMF } |
   { 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 })
   (WITH COMPONENTS { ..., contentInfo
        (WITH COMPONENTS {
                  ..., content PRESENT }) } ^
    WITH COMPONENTS { ..., signerInfos (SIZE(1..2)) })

SO { SIGNER, ToBeSigned } ::= SignedData             -- Detached content
   (CONSTRAINED BY { SIGNER, -- signs -- ToBeSigned })
   (WITH COMPONENTS { ..., contentInfo
       (WITH COMPONENTS{
                  ..., content ABSENT }) } ^
    WITH COMPONENTS { ..., signerInfos (SIZE(1..2)) })


-- Set Encapsulation Types


-- Simple Encapsulation with Signature --

Enc { SIGNER, RECIPIENT, T } ::= E {
   RECIPIENT,
   S { SIGNER, T }
}


-- Simple Encapsulation with Signature and a Provided Key --

EncK { KeyData, SIGNER, T } ::= EK {
   KeyData,
   S { SIGNER, T }
}


-- Extra Encapsulation with Signature --

EncX { SIGNER, RECIPIENT, T, Parameter } ::= E {
   RECIPIENT,
   SEQUENCE {
       t  T,
       s  SO { SIGNER, SEQUENCE { t  T, p  Parameter } }
    }
} (CONSTRAINED BY { Parameter -- data, which shall contain a fresh --
                    -- nonce 'n', is included in the OAEP block.  -- } )


-- Simple Encapsulation with Signature and Baggage --

EncB { SIGNER, RECIPIENT, T, Baggage } ::= SEQUENCE {
   enc      Enc { SIGNER, RECIPIENT, L { T, Baggage } },
   baggage  Baggage
}


-- Extra Encapsulation with Signature and Baggage --

EncBX { SIGNER, RECIPIENT, T, Baggage, Parameter } ::= SEQUENCE {
   encX     EncX { SIGNER, RECIPIENT, L { T, Baggage }, Parameter },
   baggage  Baggage
}


-- Other Cryptographic Messages --

E { RECIPIENT, ToBeEnveloped } ::= EnvelopedData
   (CONSTRAINED BY { ToBeEnveloped, -- is encrypted, and the --
                      -- session key is encrypted using the --
                      -- public key of -- RECIPIENT } )
   (WITH COMPONENTS {..., encryptedContentInfo
                  (WITH COMPONENTS { ..., encryptedContent PRESENT }) } ^
    WITH COMPONENTS { ..., recipientInfos (SIZE(1)) })

EH { RECIPIENT, ToBeEnveloped } ::= E {
   RECIPIENT,
   ToBeEnveloped
} (CONSTRAINED BY { -- H(ToBeEnveloped) included in the OAEP block -- })

EX { RECIPIENT, ToBeEnveloped, Parameter } ::= E {
   RECIPIENT,
   L { ToBeEnveloped, Parameter }
}(CONSTRAINED BY { Parameter -- data is included in the OAEP block -- })

EXH { RECIPIENT, ToBeEnveloped, Parameter } ::= EX {
   RECIPIENT,
   ToBeEnveloped,
   Parameter
} (CONSTRAINED BY { -- H(ToBeEnveloped) included in the OAEP block -- })

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

ENTITY-IDENTIFIER ::= TYPE-IDENTIFIER             -- Generic placeholder

C  ::= ENTITY-IDENTIFIER  -- Cardholder
M  ::= ENTITY-IDENTIFIER  -- Merchant
P  ::= ENTITY-IDENTIFIER  -- Payment Gateway
EE ::= ENTITY-IDENTIFIER  -- End Entity
CA ::= ENTITY-IDENTIFIER  -- Certifying Authority
P1 ::= ENTITY-IDENTIFIER  -- Gateway One
P2 ::= ENTITY-IDENTIFIER  -- Gateway Two

-- Object Identifiers --

secsig OBJECT IDENTIFIER ::= {
   iso(1) identified-organization(3) oiw(14) secsig(3) }

pkcs-1 OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }

rsaOAEPEncryptionSET OBJECT IDENTIFIER ::= { pkcs-1 6 }

id-rsaEncryption OBJECT IDENTIFIER ::= { pkcs-1 1 }

id-sha1-with-rsa-signature  OBJECT IDENTIFIER ::= { pkcs-1 5 }

id-sha1  OBJECT IDENTIFIER ::= { secsig 2 26 }

id-desCBC  OBJECT IDENTIFIER ::= { secsig 2 7 }

id-desCDMF  OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) 10}

pkcs-7 OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 7 }
```

```

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.
--

-- EXPORTS All;

IMPORTS

   id-sha1-with-rsa-signature, KeyEncryptionAlgorithms
      FROM SetPKCS7Plus;

-- 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 data types

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 information object class specification

MATCHING-RULE ::= CLASS {
   &AssertionType  OPTIONAL,
   &id             OBJECT IDENTIFIER UNIQUE
}
WITH SYNTAX {
   [ 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 }
```

```
id-at-organizationName        OBJECT IDENTIFIER ::= { id-at 10 }
id-at-organizationalUnitName  OBJECT IDENTIFIER ::= { id-at 11 }

END
```

```
SetMarketData
  { joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 8 }
      DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS All;

IMPORTS

   Date, DateTime, Distance, Location, Phone
      FROM SetMessage

   CurrencyAmount, FloatingPoint, ub-merType
      FROM SetPayMsgs

   SETString
      FROM SetAttribute;

CommercialCardData ::= SEQUENCE {
   chargeInfo        [0] ChargeInfo  OPTIONAL,
   merchantLocation  [1] Location  OPTIONAL,
   shipFrom          [2] Location  OPTIONAL,
   shipTo            [3] Location  OPTIONAL,
   itemSeq           [4] ItemSeq  OPTIONAL
}

ChargeInfo ::= SEQUENCE {
   totalFreightShippingAmount  [ 0] CurrencyAmount  OPTIONAL,
   totalDutyTariffAmount       [ 1] CurrencyAmount  OPTIONAL,
   dutyTariffReference         [ 2] EXPLICIT SETString { ub-reference }
OPTIONAL,
   totalNationalTaxAmount      [ 3] CurrencyAmount  OPTIONAL,
   totalLocalTaxAmount         [ 4] CurrencyAmount  OPTIONAL,
   totalOtherTaxAmount         [ 5] CurrencyAmount  OPTIONAL,
   totalTaxAmount              [ 6] CurrencyAmount  OPTIONAL,
   merchantTaxID               [ 7] EXPLICIT SETString { ub-taxID }  OPTIONAL,
   merchantDutyTariffRef       [ 8] EXPLICIT SETString { ub-reference }
OPTIONAL,
   customerDutyTariffRef       [ 9] EXPLICIT SETString { ub-reference }
OPTIONAL,
   summaryCommodityCode        [10] EXPLICIT SETString { ub-commCode }  OPTIONAL,
   merchantType                [11] EXPLICIT SETString { ub-merType }  OPTIONAL
}

ItemSeq ::= SEQUENCE SIZE(1..ub-items) OF Item

Item ::= SEQUENCE {
   quantity           INTEGER (1..MAX) DEFAULT 1,
   unitOfMeasureCode  [ 0] EXPLICIT SETString { ub-unitMeasure }  OPTIONAL,
   descriptor         SETString { ub-description  },
   commodityCode      [ 1] EXPLICIT SETString { ub-commCode }  OPTIONAL,
   productCode        [ 2] EXPLICIT SETString { ub-productCode }  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
}

```

```
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

END
```

```
SetPKCS10
  { joint-iso-itu-t(2) internationalRA(23) set(42) module(6) 9 }
      DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS All;

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
   --
   { WITH SYNTAX KeyUsage               ID id-ce-keyUsage              } |
   { WITH SYNTAX PrivateKeyUsagePeriod  ID id-ce-privateKeyUsagePeriod } |
   { WITH SYNTAX GeneralNames           ID id-ce-subjectAltName        } |
   --
   -- Attributes corresponding to SET private extensions
   --
   { WITH SYNTAX CertificateTypeSyntax  ID id-set-certificateType      } |
   { WITH SYNTAX TunnelingSyntax        ID id-set-tunneling            } |
   --
   -- Attributes corresponding to certificate policy
   --
   { WITH SYNTAX AdditionalPolicy       ID id-set-additionalPolicy     },
   ...
}

END
```

Cross reference

AcctData	397	394													
AcctIdentification	402	398													
AcctInfo	392	575	589												
AcqBackKey	1102	1104													
AcqCardCode	1114	1110													
AcqCardCodeMsg	1109	1104	1107												
AcqCardCodeMsgTBE	1107														
AcqCardMsg	1104	934	1098												
AcqCardMsgData	1120	1111													
AcquirerBusinessID	419	416													
AcquirerID	414	406													
AdditionalPolicies	2208	2205													
AdditionalPolicy	2210	2208	3414												
ALGORITHM-IDENTIFIER	3051	244	2751	2756	2783	2798	3054	3055	3059	3065	3070				
AlgorithmIdentifier	3053	255	331	2003	2029	2041	2636	2726	2729	2749	2777	2794	2809		
AmountType	1765	1686	1758												
ApprovalCode	1215	1171													
ARqExtensionsIOS	1028	1025													
ARsExtensionsIOS	1132	1129													
ARvRqExtensionsIOS	1245	1242													
ARvRsExtensionsIOS	1288	1285													
ATTRIBUTE	3082	2734	3012	3017	3022	3027	3035	3036	3040	3041	3044	3083	3399		
Attribute	3034	2817	3383												
AttributeSeq	2816	2728	2731												
AttributeSet	3382	3396													
AttributeTypeAndValue	3039	3080													
AttributeUsage	3107	3092													
AuthCharInd	1217	1172	1925												
AuthCode	1142	942	1136												
Authenticated	2734	2728													
AuthHeader	1134	1127													
AuthorityKeyIdentifier	2124	2120													
authorityKeyIdentifier	2119	2090													
AuthReq	983	86													
AuthReqData	990	983	988	1239											
AuthReqItem	998	991	1352												
AuthReqPayload	1015	1001													
AuthReqTBE	986														
AuthReqTBS	988	986													
AuthRes	1069	87													
AuthResBaggage	1096	1070	1071	1082											
AuthResData	1089	1070	1071	1082											
AuthResDataNew	1303	1284													
AuthResPayload	1126	1093	1240	1305	1353										
AuthResTBE	1075														
AuthResTBEX	1077														
AuthResTBS	1082	1075	1078	1085											
AuthResTBSX	1084	1079													
AuthRetNum	1259	1007	1254												
AuthRevCode	1290	1279													
AuthRevReq	1229	89													
AuthRevReqBaggage	1247	1229	1234												
AuthRevReqData	1236	1229	1234												
AuthRevReqTBE	1232														
AuthRevReqTBS	1234	1232													
AuthRevRes	1261	90													
AuthRevResBaggage	1273	1262	1271												
AuthRevResData	1278	1262	1263	1267	1271										
AuthRevResTBE	1267														

AuthRevResTBEB	1269														
AuthRevResTBS	1271	1269													
AuthRevRRTags	1257	1253													
AuthRevTags	1252	1237	1280												
AuthStatus	940	935													
AuthTags	1004	999	1090												
AuthToken	1787	825	1099	1275											
AuthTokenData	1800	1787	1791	1796											
AuthTokenTBE	1790														
AuthTokenTBS	1795	1792													
AuthValCodes	1170	1163													
AutoApplicableRate	3238	3230													
AutoCharges	3243	3221													
AutoNoShow	3224	3214													
AutoRateInfo	3229	3218													
AVSData	1030	1018													
AVSResult	1205	1166													
BackKey	248	245													
BackKeyData	243	603	692	840	1102	1804									
BARqExtensionsIOS	1638	1635													
BARsExtensionsIOS	1674	1671													
basicConstraints	2246	2096													
BasicConstraintsSyntax	2252	2247													
BAStatus	1691	1666													
BatchAdminReq	1621	107													
BatchAdminReqData	1626	1621	1624												
BatchAdminReqTBE	1624														
BatchAdminRes	1658	108													
BatchAdminResData	1663	1658	1661												
BatchAdminResTBE	1661														
BatchDetails	1727	1721													
BatchID	1812	1387	1426	1457	1628	1665	1921								
BatchOperation	1640	1630													
BatchSequenceNum	1814	1388	1458	1755	1922										
BatchStatus	1718	1138	1633	1667	1716										
BatchStatusSeq	1716	1369	1439												
BatchTotals	1739	1728	1736												
BCIDistribution	225														
BCIDistributionTBS	227	225													
BIN	250	410	415	763	861	1589	2306								
BIT STRING	ASN.1	2030	2038	2042	2139	2286									
BMPString	ASN.1	3127	3132												
BOOLEAN	ASN.1	269	466	467	993	1016	1021	1429	1631	1649	1931	2075	2253	2309	2324
		2334	3088	3089	3091	3200									
BrandAndBIN	1587	1585													
BrandAndBINSeq	1585	1579	1629												
BrandBatchDetails	1734	1732													
BrandBatchDetailsSeq	1732	1729													
BrandCRLIdentifier	191	229	502	531	568	650	777	909	1091	1281	1367	1437	1617		
BrandCRLIdentifierSeq	1617	1597													
BrandID	232	200	490	514	762	860	1588	1650	1735	1754					
BSExtensionsIOS	1725	1722													
BTExtensionsIOS	1747	1744													
C	2944	811	968												
CA	2948	209	216	225	494	519	537	557	575	576	636	637	641		
CAKey	692	637													
CAMsg	684	658													
CapCode	1394	949	1385												
CapItem	1343	1341													
CapItemSeq	1341	1333													
CapPayload	1349	1346	1425												
CapReq	1310	92													
CapReqData	1330	1311	1312	1323											
CapReqTBE	1316														

CapReqTBEX	1318														
CapReqTBS	1323	1316	1319	1326											
CapReqTBSX	1325	1320													
CapRes	1360	93													
CapResData	1365	1360	1363												
CapResItem	1378	1376													
CapResItemSeq	1376	1370													
CapResPayload	1384	1128	1381												
CapResTBE	1363														
CapRevData	1501	1482	1483	1494											
CapRevOrCredCode	1464	957	1455												
CapRevOrCredReqData	1411	1501	1532	1563											
CapRevOrCredReqItem	1422	1420													
CapRevOrCredReqItemSeq	1420	1414													
CapRevOrCredResData	1435	1508	1539	1570											
CapRevOrCredResItem	1448	1446													
CapRevOrCredResItemSeq	1446	1440													
CapRevOrCredResPayload	1454	1451													
CapRevReq	1481	95													
CapRevReqTBE	1487														
CapRevReqTBEX	1489														
CapRevReqTBS	1494	1487	1490	1497											
CapRevReqTBSX	1496	1491													
CapRevRes	1503	96													
CapRevResData	1508	1503	1506												
CapRevResTBE	1506														
CapRRTags	1339	1331	1366												
CapStatus	947	936													
CapToken	1816	1097	1249	1274	1841										
CapTokenData	1835	1817	1818	1823	1826	1831									
CapTokenSeq	1841	1311	1312	1323	1482	1483	1494	1513	1514	1525	1544	1545	1556		
CapTokenTBE	1823														
CapTokenTBEX	1825														
CapTokenTBS	1830	1827													
cardCertRequired	2323	2104													
CardCInitReq	486	110													
CardCInitRes	494	111													
CardCInitResTBS	496	494													
CardExpiry	252	302	309	316	2846										
CardSuspect	1040	1020													
CardType	1186	1165													
CBC8Parameter	2789	2784	2785												
CERT-POLICY-QUALIFIER	2189	2177	2179	2184	2198										
Certificate	2023	2746													
certificatePolicies	2160	2093													
CertificatePoliciesSyntax	2166	2161													
Certificates	2746	2714													
CertificateSerialNumber	2015	2002	2127	2647	2692										
certificateType	2280	2102													
CertificateTypeSyntax	2286	2213	2281	3409											
CertificateVersion	2013	2001													
CertificationRequest	3388														
CertificationRequestInfo	3392	3386													
CertInqReq	696	122													
CertInqReqTBS	698	696													
CertInqRes	705	123													
CertPolicyId	2174	2169	2211												
CertReq	574	119													
CertReqData	592	575	576	580	583	588									
CertReqTBE	580														
CertReqTBEX	582														
CertReqTBS	587	584													
CertRes	635	120	705												
CertResData	643	636	637	641											

CertResTBE	641														
CertStatus	654	648													
CertStatusCode	669	655													
CertThumb	254	155	501	530	565	605	778	1092	1282	1368	1438	1607			
Challenge	259	489	499	511	524	526	545	562	564	595	597	646	701	761	775
		776	856	859	908	973									
ChargeInfo	3175	3168													
CheckDigests	1010	1000													
CHOICE	ASN.1	75	144	159	392	404	442	574	618	635	787	822	963	1069	1261
		1310	1481	1512	1543	1816	1878	1884	1952	2232	3075	3125	3130	3238	
CLASS	ASN.1	2073	2189	3082	3116										
ClosedWhen	1706	1720													
CloseStatus	1711	1707													
CommercialCardData	3167	1927													
commonName	3012	3048													
CompletionCode	923	916													
ContentEncryptionAlgorithms	2783	244	2777												
ContentInfo	2702	2713	2810												
CONTENTS	2695	2697	2704	2708											
Contents	2697	2704	2708												
ContentType	2708	2703	2735	2775											
contentType	2984	2735													
CountryCode	261	287	2308												
countryName	3017	3045													
CPayExtensionsIOS	1358	1355													
CreditStatus	955	953													
CreditStatusSeq	953	937													
CredReq	1512	98													
CredReqData	1532	1513	1514	1525											
CredReqTBE	1518														
CredReqTBEX	1520														
CredReqTBS	1525	1518	1521	1528											
CredReqTBSX	1527	1522													
CredRes	1534	99													
CredResData	1539	1534	1537												
CredResTBE	1537														
CredRevReq	1543	101													
CredRevReqData	1563	1544	1545	1556											
CredRevReqTBE	1549														
CredRevReqTBEX	1551														
CredRevReqTBS	1556	1549	1552	1559											
CredRevReqTBSX	1558	1553													
CredRevRes	1565	102													
CredRevResData	1570	1565	1568												
CredRevResTBE	1568														
CRL	2654	2688													
CRLEntry	2646	2644													
CRLEntryList	2644	2640													
CRLIdentifier	236	234													
CRLIdentifierSeq	234	203													
CRLNotification	209														
CRLNotificationRes	216														
CRLNotificationResTBS	218	216													
CRLNotificationTBS	211	209													
CRLNumber	2264	2260													
cRLNumber	2259	2097													
CRLSequence	2688	2715													
CRqExtensionsIOS	1337	1334													
CRsExtensionsIOS	1374	1371													
CRsPayExtensionsIOS	1392	1389													
CRvRqExtensionsIOS	1418	1415													
CRvRqItemExtensionsIOS	1433	1430													
CRvRsExtensionsIOS	1444	1441													
CRvRsPayExtensionsIOS	1462	1459													

CurrConv	1853	944	1139												
Currency	263	687	1844	1855											
CurrencyAmount	1843	848	874	1017	1135	1241	1283	1351	1386	1428	1456	1685	1741	1743	1757
		1802	1808	1837	3176	3177	3179	3180	3181	3182	3198	3199	3201	3202	3205
		3206	3207	3231	3232	3239	3240	3244	3245	3247	3248	3249	3250	3251	3252
		3253	3254	3255	3256	3257	3258	3279	3280	3284	3285	3286	3287	3288	3289
		3290	3291	3292	3293	3294	3295	3296	3297	3298	3299	3300	3323		
data	2976														
Date	265	61	212	219	228	268	341	599	941	948	956	1350	1427	1688	1708
		1719	1807	1917	1959	3262	3264	3305	3317						
DateTime	267	3213	3220												
DD	2826	818	820	870	2277	2823									
DetachedDigest	2830	2217	2826												
Digest	2744	213	220	256	2740	2742	2811	2840							
DigestAlgorithmIdentifiers	2748	2712													
DigestAlgorithms	2751	255	331	2726	2749	2809									
DigestedData	2807	2830													
digestedData	2979														
DigestEncryptionAlgorithms	2756	2729													
Digests	2742	332	333	334											
DirectoryString	3125	3013	3023	3028											
Distance	272	3233	3246												
DistanceScale	277	273													
ds	3141	3141	3143												
Duration	1869	1861	1865												
E	2913	2869	2885	2921	2926										
EE	2947	149	575	576	580	584	696								
EH	2921														
EK	2937	2877													
Enc	2869	576	1263	1360	1503	1534	1565	1621	1658	1818	2898				
EncB	2897	983	1070	1229	1262	1311	1482	1513	1544						
EncBX	2905	1071	1312	1483	1514	1545									
EncK	2877	637	1104												
EncodedBrandCRLID	195	192													
EncodedCertificate	2021	2024													
EncodedCertificationRequestInfo	3385	3389													
EncodedCRL	2652	2655													
EncryptedContent	2781	2778													
EncryptedContentInfo	2774	2763	2769												
EncryptedData	2761	2937													
EncryptedDigest	2814	2730													
EncryptedKey	2805	2795													
encryptionAlgorithm	ASN.1	2971													
EncX	2885	575	1787	1817	2906										
ENTITY-IDENTIFIER	2942	2944	2945	2946	2947	2948	2949	2950							
ENUMERATED	ASN.1	164	277	421	669	923	1035	1040	1058	1114	1142	1177	1186	1205	1217
		1290	1394	1464	1610	1640	1676	1691	1711	1765	1770	1775	1871	1897	1937
		3107	3224	3274	3326	3331									
EnvelopedData	2766	2913													
envelopedData	2978														
Error	144	125													
ErrorCode	164	152													
ErrorMsg	159	156													
ErrorTBS	151	146	149												
EX	2926	801	2931												
EXH	2931	537	898												
EXTENSION	2073	56	138	139	141	768	783	844	866	880	921	977	1028	1132	1245
		1288	1337	1358	1374	1392	1418	1433	1444	1462	1583	1601	1638	1674	1725
		1747	1763	1935	1950	2086	2111	2112	2119	2133	2149	2160	2225	2239	2246
		2259	2268	2280	2299	2323	2328	2340							
Extension	2110	2084													
Extensions	2084	204	2010	2641	2649										
ExtensionSet	2086	2111	2112												
FailedItem	664	662													

FailedItemSeq	662	659													
FieldName	616	463	612												
FieldValue	618	613													
FloatingPoint	1858	943	950	958	1854	3203									
GeneralizedTime	ASN.1	201	202	265	2155	2156									
GeneralName	2232	2230													
GeneralNames	2230	2126	2226	2240	3405										
H	2835														
hashedRootKey	2268	2101													
HashedRootKeySyntax	2274	2269													
HMAC	2840	851													
HMACPanData	2844														
HOD	870	847	857	1012											
HODInput	872	870													
HOIData	820	815	1011												
HotelCharges	3283	3271													
HotelNoShow	3274	3263													
HotelRateInfo	3278	3270													
HPIData	818	814													
IA5String	ASN.1	2234													
id-at	3143	3144	3145	3146	3147										
id-at-commonName	3144	3014													
id-at-countryName	3145	3019													
id-at-organizationalUnitName	3147	3029													
id-at-organizationName	3146	3024													
id-ce	2358	2359	2360	2361	2362	2363	2364	2365	2366						
id-ce-authorityKeyIdentifier	2366	2121													
id-ce-basicConstraints	2363	2249													
id-ce-certificatePolicies	2365	2163													
id-ce-cRLNumber	2364	2261													
id-ce-issuerAltName	2362	2241													
id-ce-keyUsage	2359	2136	3403												
id-ce-privateKeyUsagePeriod	2360	2151	3404												
id-ce-subjectAltName	2361	2227	3405												
id-desCBC	2968	2785													
id-desCDMF	2970	2784													
id-rsaEncryption	2962	2757													
id-set	2368	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385			
id-set-accountNumber	2487														
id-set-additionalPolicy	2494	3414													
id-set-address	2484														
id-set-algorithm	2379														
id-set-AmericanExpress	2516														
id-set-amount	2486														
id-set-Antares	2556														
id-set-attribute	2378	2491													
id-set-attribute-cert	2491	2493	2494												
id-set-BankGate	2542														
id-set-birthFamilyName	2479														
id-set-BlueMoney	2566														
id-set-brand	2383	2512	2514	2516	2518	2520	2522	2524							
id-set-cardCertRequired	2506	2325													
id-set-certExt	2382	2503	2504	2505	2506	2507	2508	2509							
id-set-Certicom	2550														
id-set-certificateType	2504	2283	3409												
id-set-CompuSource	2546														
id-set-content-AcqCardCodeMsg	2410														
id-set-content-AcqCardCodeMsgTBE	2446														
id-set-content-AuthReqTBE	2440														
id-set-content-AuthReqTBS	2404														
id-set-content-AuthResBaggage	2396														
id-set-content-AuthResTBE	2441														
id-set-content-AuthResTBEX	2442														
id-set-content-AuthResTBS	2405														

id-set-content-AuthResTBSX	2406														
id-set-content-AuthRevReqBaggage	2397														
id-set-content-AuthRevReqTBE	2447														
id-set-content-AuthRevReqTBS	2411														
id-set-content-AuthRevResBaggage	2398														
id-set-content-AuthRevResData	2412														
id-set-content-AuthRevResTBE	2448														
id-set-content-AuthRevResTBEB	2449														
id-set-content-AuthRevResTBS	2413														
id-set-content-AuthTokenTBE	2443														
id-set-content-AuthTokenTBS	2407														
id-set-content-BatchAdminReqData	2428														
id-set-content-BatchAdminReqTBE	2462														
id-set-content-BatchAdminResData	2429														
id-set-content-BatchAdminResTBE	2463														
id-set-content-BCIDistributionTBS	2470														
id-set-content-CapReqTBE	2450														
id-set-content-CapReqTBEX	2451														
id-set-content-CapReqTBS	2414														
id-set-content-CapReqTBSX	2415														
id-set-content-CapResData	2416														
id-set-content-CapResTBE	2452														
id-set-content-CapRevReqTBE	2453														
id-set-content-CapRevReqTBEX	2454														
id-set-content-CapRevReqTBS	2417														
id-set-content-CapRevReqTBSX	2418														
id-set-content-CapRevResData	2419														
id-set-content-CapRevResTBE	2455														
id-set-content-CapTokenData	2408														
id-set-content-CapTokenSeq	2399														
id-set-content-CapTokenTBE	2444														
id-set-content-CapTokenTBEX	2445														
id-set-content-CapTokenTBS	2409														
id-set-content-CardCInitResTBS	2430														
id-set-content-CertInqReqTBS	2436														
id-set-content-CertReqData	2433														
id-set-content-CertReqTBE	2465														
id-set-content-CertReqTBEX	2466														
id-set-content-CertReqTBS	2434														
id-set-content-CertResData	2435														
id-set-content-CertResTBE	2467														
id-set-content-CredReqTBE	2456														
id-set-content-CredReqTBEX	2457														
id-set-content-CredReqTBS	2420														
id-set-content-CredReqTBSX	2421														
id-set-content-CredResData	2422														
id-set-content-CredResTBE	2458														
id-set-content-CredRevReqTBE	2459														
id-set-content-CredRevReqTBEX	2460														
id-set-content-CredRevReqTBS	2423														
id-set-content-CredRevReqTBSX	2424														
id-set-content-CredRevResData	2425														
id-set-content-CredRevResTBE	2461														
id-set-content-CRLNotificationResTBS	2469														
id-set-content-CRLNotificationTBS	2468														
id-set-content-ErrorTBS	2437														
id-set-content-HODInput	2395														
id-set-content-InqReqData	2403														
id-set-content-Me-AqCInitResTBS	2431														
id-set-content-OIData	2391														
id-set-content-PANData	2388														
id-set-content-PANOnly	2390														
id-set-content-PANToken	2389														
id-set-content-PCertReqData	2426														

id-set-content-PCertResTBS	2427														
id-set-content-PI	2392														
id-set-content-PI-TBS	2401														
id-set-content-PIData	2393														
id-set-content-PIDataUnsigned	2394														
id-set-content-PIDualSignedTBE	2438														
id-set-content-PInitResData	2400														
id-set-content-PIUnsignedTBE	2439														
id-set-content-PResData	2402														
id-set-content-RegFormReqTBE	2464														
id-set-content-RegFormResTBS	2432														
id-set-contentType	2375	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399	2400	2401
		2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
		2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429
		2430	2431	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443
		2444	2445	2446	2447	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457
		2458	2459	2460	2461	2462	2463	2464	2465	2466	2467	2468	2469	2470	
id-set-Cybercash	2532														
id-set-date	2483														
id-set-Diners	2514														
id-set-e-COMM	2602														
id-set-ECC	2558														
id-set-eLab	2572														
id-set-Entrust	2574														
id-set-espace-net	2590														
id-set-familyName	2478														
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-Maithean	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														

id-set-policy	2380	2500													
id-set-policy-root	2500														
id-set-rootKeyThumb	2493														
id-set-RSADSI	2536														
id-set-setExtensions	2508	2342													
id-set-setQualifier	2509	2199													
id-set-telephone	2485														
id-set-TenthMountain	2554														
id-set-Terisa	2534														
id-set-Trintech	2540														
id-set-tunneling	2507	2330	3410												
id-set-vendor	2384	2528	2530	2532	2534	2536	2538	2540	2542	2544	2546	2548	2550	2552	2554
		2556	2558	2560	2562	2564	2566	2568	2570	2572	2574	2576	2578	2580	2582
		2584	2586	2588	2590	2592	2594	2596	2598	2600	2602	2604			
id-set-VeriFone	2538														
id-set-VeriSign	2564														
id-set-VIAnet	2576														
id-set-Visa	2520														
id-sha1	2966	2752													
id-sha1-with-rsa-signature	2964	3071													
IDData	404	513	600												
identified-organization	ASN.1	2955													
Inputs	846	835													
InqReq	963	83													
InqReqData	970	965	968												
InqReqSigned	968	964													
InqRes	979	84													
InqRqExtensionsIOS	977	974													
InstallRecurData	1945	837	876	1022	1805										
InstallRecurInd	1952	1946													
INTEGER	ASN.1	59	60	198	199	238	261	263	274	352	353	354	355	356	357
		358	359	453	465	601	665	709	710	711	712	713	714	715	716
		717	1259	1647	1648	1654	1740	1742	1806	1812	1814	1845	1846	1869	1909
		1910	1911	1953	1958	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975
		1976	1977	2013	2015	2254	2264	2349	2350	2351	2352	2353	2354	2635	2711
		2724	2762	2767	2792	2808	3137	3138	3139	3193	3265	3335	3336	3337	3338
		3339	3340	3341	3342	3343	3344	3345	3346	3347	3348	3349	3350	3351	3352
		3353	3354	3355	3356	3357	3358	3359	3393						
internationalRA	ASN.1	2369													
IRExtensionsIOS	1950	1947													
iso	ASN.1	2955	2958	2971	2974	2982									
issuerAltName	2239	2095													
IssuerAndSerialNumber	2690	2725	2793												
Item	3192	3190													
ItemSeq	3190	3172													
joint-iso-itu-t	ASN.1	2369	3141												
KeyEncryptionAlgorithms	2798	626	2794	3061											
KeyIdentifier	2131	2125													
KeyUsage	2139	2134	3403												
keyUsage	2133	2091													
L	2821	540	805	807	809	891	901	988	1082	1234	1271	1323	1494	1525	1556
		2898	2906	2928											
Language	282	343	515	548	758	2315									
LocalID	284	68	69	338	339	488	498	500	510	523	525	544	546	561	563
		594	596	645	647	700	702	759	760						
Location	286	1032	3169	3170	3171	3212	3219								
LogRefID	1213	1167													
M	2945	770	903	983	986	1070	1071	1229	1232	1262	1263	1311	1312	1316	1320
		1360	1482	1483	1487	1491	1503	1513	1514	1518	1522	1534	1544	1545	1549
		1553	1565	1574	1621	1624	1658								
MarketAutoAuth	1860	1879													
MarketAutoCap	3210	1885													
MarketHotelAuth	1864	1880													
MarketHotelCap	3261	1886													

MarketSpecAuthData	1878	1023													
MarketSpecCapData	1884	1892													
MarketSpecDataID	1897	1174	1891												
MarketSpecSaleData	1890	1926													
MarketTransportAuth	1895	1881													
MarketTransportCap	3303	1887													
MATCHING-RULE	3116	3085	3086	3087											
Me-AqCInitReq	508	113													
Me-AqCInitRes	519	114													
Me-AqCInitResTBS	521	519													
member-body	ASN.1	2958	2971	2974	2982										
MerchantAcquirerID	409	405													
merchantData	2299	2103													
MerchantDataSyntax	2304	2300													
MerchantID	294	411	836	1803	1907	2305									
MerchCatCode	1054	1050													
MerchData	1049	1024													
MerchGroup	1058	1051													
MerNames	2314	2312													
MerNameSeq	2312	2307													
MerOrderNum	1904	1924													
MerTermIDs	1906	1916													
MESSAGE	73	45													
Message	75	45													
MessageDigest	2740	2736													
messageDigest	2986	2736													
MessageHeader	58	44	160												
MessageIDs	67	62													
MessageWrapper	43														
MsgExtension	137	135													
MsgExtensions	134	46	765	780	841	863	877	918	974	1025	1129	1242	1285	1334	1355
		1371	1389	1415	1430	1441	1459	1580	1598	1635	1671	1722	1744	1760	1932
		1947													
MWExtensionsIOS	56	46													
Name	3075	237	2004	2006	2233	2637	2691	3394							
Nonce	296	153	304	311	317	399	554	656	858	875					
NULL	ASN.1	1819	1895	2752	2757	2799	3071								
NumericString	ASN.1	250	252	298	419	1054	1213								
OBJECT IDENTIFIER	ASN.1	154	462	868	2074	2174	2190	2235	2338	2345	2358	2359	2360	2361	2362
		2363	2364	2365	2366	2368	2373	2954	2957	2960	2962	2964	2966	2968	2970
		2973	2976	2977	2978	2979	2981	2984	2986	3093	3118	3141	3143	3144	3145
		3146	3147												
OCTET STRING	ASN.1	161	248	259	284	296	324	326	348	620	882	2113	2131	2744	2781
		2789	2805	2814	2835										
OD	882	873													
ODExtensionsIOS	880	877													
OID	2373	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385	2388	2389	2390
		2391	2392	2393	2394	2395	2396	2397	2398	2399	2400	2401	2402	2403	2404
		2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415	2416	2417	2418
		2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431	2432
		2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446
		2447	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460
		2461	2462	2463	2464	2465	2466	2467	2468	2469	2470	2476	2477	2478	2479
		2480	2481	2482	2483	2484	2485	2486	2487	2488	2491	2493	2494	2500	2503
		2504	2505	2506	2507	2508	2509	2512	2514	2516	2518	2520	2522	2524	2528
		2530	2532	2534	2536	2538	2540	2542	2544	2546	2548	2550	2552	2554	2556
		2558	2560	2562	2564	2566	2568	2570	2572	2574	2576	2578	2580	2582	2584
		2586	2588	2590	2592	2594	2596	2598	2600	2602	2604	2609			
OIData	853	807	809	820	891										
OIDList	868	862													
OIDualSigned	809	796													
OIExtensionsIOS	866	863													
OIUnsigned	891	888													
oiw	ASN.1	2955													

organizationalUnitName	3027	3047													
organizationName	3022	3046													
P	2946	801	898	983	1070	1071	1075	1079	1104	1107	1229	1262	1263	1267	1269
		1311	1312	1360	1363	1482	1483	1503	1506	1513	1514	1534	1537	1544	1545
		1565	1568	1592	1621	1658	1661								
P1	2949	1787	1792	1817	1818	1823	1827								
P2	2950	1787	1817	1818											
PAN	298	301	308	315	553	2845									
PANData	300	801	805	830											
PANData0	307	393													
PANOnly	552	537	540												
PANToken	314	895	898	901	1071	1086	1312	1327	1483	1498	1514	1529	1545	1560	1787
		1797	1817	1832											
PayRecurInd	1937	1923													
PaySysID	320	342													
PCertCode	1610	1606													
PCertReq	1574	104													
PCertReqData	1576	1574													
PCertRes	1592	105													
PCertResItem	1605	1603													
PCertResItemSeq	1603	1596													
PCertResTBS	1594	1592													
PCRqExtensionsIOS	1583	1580													
PCRsExtensionsIOS	1601	1598													
Phone	322	1930	3267	3268											
PI	822	983	988	1248											
PI-OILink	807	801	805	898	901										
PI-TBS	813	811													
PIData	828	809	818												
PIDataUnsigned	893	891													
PIDualSigned	799	795	824												
PIDualSignedTBE	805														
PIExtensionsIOS	844	841													
PIHead	833	807	829	894											
PInitReq	756	77													
PInitRes	770	78													
PInitResData	772	770													
PIRqExtensionsIOS	768	765													
PIRsExtensionsIOS	783	780													
PISignature	811	800													
PIUnsigned	898	823	887												
PIUnsignedTBE	901														
pkcs	ASN.1	2958	2974	2982											
pkcs-1	2957	2960	2962	2964											
pkcs-7	2973	2976	2977	2978	2979										
pkcs-9	2981	2984	2986												
PolicyInformation	2168	2166													
PolicyQualifierInfo	2176	2171													
PolicyText	482	449													
PReq	787	80													
PReqDualSigned	794	788													
PReqUnsigned	886	789													
PRes	903	81	979												
PResData	905	903													
PResPayload	915	913													
PResPayloadSeq	913	910													
Prestige	1871	1866													
PrintableString	ASN.1	3018	3126												
PrivateKeyUsagePeriod	2154	2150	3404												
privateKeyUsagePeriod	2149	2092													
PRsExtensionsIOS	921	918													
PublicKeySorE	623	604													
RDNSequence	3078	3076													
REAL	ASN.1	1858													

Reason	476	471													
RecipientInfo	2791	2772													
RecipientInfos	2772	2768													
Recurring	1957	1954													
ReferralData	470	444													
ReferralURL	480	478													
ReferralURLSeq	478	472													
RegField	461	459	529												
RegFieldSeq	459	456													
RegForm	609	602													
RegFormData	447	443													
RegFormItems	611	609													
RegFormOrReferral	442	528	567												
RegFormReq	537	116													
RegFormReqData	542	537	540												
RegFormReqTBE	540														
RegFormRes	557	117													
RegFormResTBS	559	557													
RegTemplate	452	448													
ReimbursementID	1775	1756													
RelativeDistinguishedName	3080	3078													
RequestType	421	512	527	547	566	598									
ResponseData	1162	1137													
RespReason	1177	1164													
Restrictions	3331	3311													
Results	933	917													
ReturnTransactionDetail	1646	1632													
RootKeyThumb	2276	2274													
RRPID	324	63	487	497	509	522	543	560	593	644	699	757	774	855	907
		972	1345	1380	1424	1450	1753	1836	1915						
RRTags	1914	1005	1257	1339	1412	1436	1577	1595	1627	1664					
rsadsi	ASN.1	2958	2971	2974	2982										
rsaOAEPEncryptionSET	2960	2799													
S	2849	149	209	216	225	494	519	557	580	636	641	696	770	903	968
		986	1075	1107	1232	1267	1269	1316	1363	1487	1506	1518	1537	1549	1568
		1574	1592	1624	1661	1823	2871	2879							
SaleDetail	1920	994	1354												
SaleExtensionsIOS	1935	1932													
Secret	326	303	310	851											
secsig	2954	2955	2966	2968											
SEQUENCE	ASN.1	43	58	67	135	137	151	197	211	218	227	234	236	243	254
		267	272	286	300	307	314	330	337	397	409	414	447	452	459
		461	470	478	486	496	508	521	542	552	559	582	587	592	609
		611	623	643	654	662	664	684	698	756	772	794	799	813	828
		833	846	853	868	872	886	893	905	913	915	933	940	947	953
		955	970	990	998	1004	1010	1015	1030	1049	1077	1084	1089	1096	1109
		1120	1126	1134	1162	1170	1236	1247	1252	1273	1278	1303	1318	1325	1330
		1341	1343	1349	1365	1376	1378	1384	1411	1420	1422	1435	1446	1448	1454
		1489	1496	1520	1527	1551	1558	1576	1585	1587	1594	1603	1605	1617	1626
		1646	1653	1663	1684	1706	1716	1718	1727	1732	1734	1739	1749	1751	1790
		1795	1800	1825	1830	1835	1841	1843	1853	1860	1864	1890	1906	1914	1920
		1945	1957	2000	2027	2033	2040	2084	2110	2124	2154	2166	2168	2170	2176
		2203	2208	2210	2216	2230	2252	2276	2304	2312	2314	2333	2338	2345	2634
		2644	2646	2688	2690	2702	2710	2719	2723	2742	2746	2749	2761	2766	2772
		2774	2791	2807	2817	2821	2844	2887	2889	2897	2905	3034	3039	3053	3078
		3167	3175	3190	3192	3210	3229	3243	3261	3278	3283	3303	3314	3316	3392
SET	ASN.1	3036	3080	3383											
set	ASN.1	2369													
setExtensions	2340	2106													
SETExtensionsSyntax	2345	2341													
SetPolicyQualifier	2203	2200													
setPolicyQualifier	2198	2185													
SETQualifier	2216	2204	2212												
SETString	3130	232	288	289	290	291	294	322	464	476	482	616	619	657	666

		689	1031	1121	1123	1687	1928	1929	2218	2316	2317	2318	2319	2320	3178
		3183	3184	3185	3186	3187	3194	3195	3196	3197	3204	3211	3215	3216	3217
		3234	3235	3266	3269	3304	3306	3308	3309	3310	3318	3319	3321	3322	
SettlementInfo	1684	1669													
sha1-with-rsa-signature	3070	3066													
SignatureAlgorithms	3065	624	2003	2029	2636	3062									
SIGNED	2027	191	2023	2654	3388										
SignedData	2710	2698	2849	2856											
signedData	2977	2698													
SignedError	149	145													
SignerInfo	2723	2719													
SignerInfos	2719	2716													
SO	2856	584	811	1079	1320	1491	1522	1553	1792	1827	2889				
SpecialProcessing	1035	1019													
StopOverCode	3326	3320													
subjectAltName	2225	2094													
SubjectPublicKeyInfo	2040	624	626	2007	2277	3395									
SupportedAlgorithms	3059	2007	2277	3395											
SupportedAttributes	3044	3040	3041												
SupportedCRIAttributes	3399	3396													
SupportedPolicyQualifiers	2184	2178	2180												
SWIdent	328	64	839												
Thumbs	330	491	503	516	532	549	569	606	649	651	764	779	992	1238	1332
		1413	1578												
TokenOpaque	1962	1809	1838												
TransactionDetail	1751	1749													
TransactionDetailSeq	1749	1655													
TransactionStatus	1770	1759													
TransDetails	1653	1634	1670												
TransExtensionsIOS	1763	1760													
TransIDs	337	773	834	854	906	971	1006	1304	1344	1379	1423	1449	1752	1801	
TransmissionStatus	1676	1668													
TransStain	851	838													
TripLeg	3316	3314													
TripLegSeq	3314	3307													
TunnelAlg	2338	2335													
tunneling	2328	2105													
TunnelingSyntax	2333	2329	3410												
TYPE-IDENTIFIER	ASN.1	73	195	1962	2021	2652	2695	2942	3051	3386					
ub-acctIdentification	709	402													
ub-acqBusinessID	355	419													
ub-acqCardPhone	1967	1123													
ub-acqCardText	1966	1121													
ub-airportCode	3335	3306	3321												
ub-approvalCode	1968	1215													
ub-AVSData	1969	1031													
ub-BrandID	352	232													
ub-cardholderMsg	710	689													
ub-carrierCode	3336	3318													
ub-cityName	2350	288	2317												
ub-commCode	3337	3186	3196												
ub-common-name	3137	3013													
ub-corpID	3338	3235													
ub-countryName	2349	2320													
ub-description	3339	3195													
ub-eeMessage	711	657													
ub-fareBasis	3340	3322													
ub-FieldDesc	712	464													
ub-FieldList	713	459	609	662											
ub-FieldName	714	616													
ub-FieldValue	715	465	465	619	620										
ub-hotelFolio	3341	3266													
ub-insuranceType	3342	3217													
ub-items	3343	3190													

ub-locationID	356	291													
ub-logRefID	1970	1213													
ub-MerchantID	353	294													
ub-merName	2351	2316													
ub-merOrderNum	1971	1904													
ub-merType	1972	1054	3187												
ub-organization-name	3138	3023													
ub-organizational-unit-name	3139	3028													
ub-passName	3344	3304													
ub-paySysID	357	320													
ub-phone	3345	322													
ub-PolicyText	716	482													
ub-postalCode	2352	290	2319												
ub-productCode	3346	3197													
ub-programCode	3347	3269													
ub-Reason	717	476	666												
ub-recurringFrequency	1973	1958													
ub-reference	3348	1929	3178	3184	3185										
ub-rentalNum	3349	3215													
ub-rentalRefNum	3350	3216													
ub-renterName	3351	3211													
ub-RFC1766-language	358	282													
ub-serviceClass	3352	3319													
ub-SettlementAccount	1974	1687													
ub-stateProvince	2353	289	2318												
ub-summary	1975	1928													
ub-SWIdent	354	328													
ub-taCode	3353	3309													
ub-taName	3354	3310													
ub-taxID	3355	3183													
ub-taxType	3356	3204													
ub-terminalID	1976	1908													
ub-terseStatement	2354	2218													
ub-ticketNum	3357	3308													
ub-unitMeasure	3359	3194													
ub-URL	359	346													
ub-validationCode	1977	1225													
ub-vehicleClass	3358	3234													
UniqueIdentifier	2038	2008	2009												
UnsignedBrandCRLIdentifier	197	195													
UnsignedCertificate	2000	2021													
UnsignedCertificateRevocationList	2634	2652													
URL	346	454	455	480	685	686	1122	2219	2220						
us	ASN.1	2958	2971	2974	2982										
userApplications	ASN.1	3092													
UTCTime	ASN.1	2034	2035	2638	2639	2648									
ValidationCode	1225	1173													
Validity	2033	2005													
VisibleString	ASN.1	282	320	328	346	402	1215	1225	1904	1908	3131				
XID	348	70	340	851											

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, EncBX, 14
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
Integrity Encryption - EH, 9

—K—

Keyed hash mechanism - HMAC, 7

—L—

L, 7
Linkage - L, 7

—M—

MarketAutoCap, 62
MarketHotelCap, 66
MarketTransportCap, 69
Me-AqCInitReq, 160
Me-AqCInitRes, 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
RegFormReq, 165
RegFormRes, 167
RequestType, 149
RRTags, 34

—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