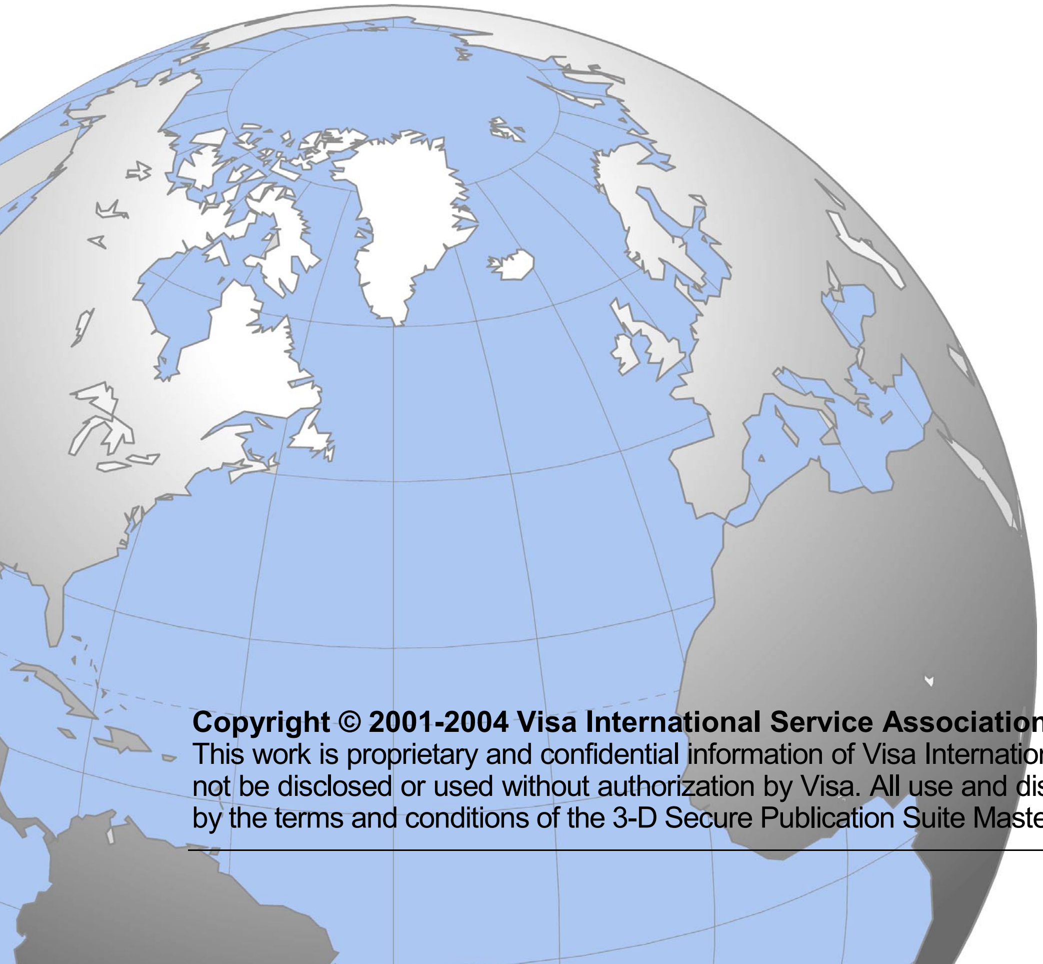
3-D Secure™ Protocol Specification

Core Functions



Version 1.0.2
July 16, 2002
Errata as of January 20, 2004

70000-01

The enclosed documentation and technology has been developed and is owned by Visa. These materials have been distributed and provided to Visa Regions and Visa Members for their internal use. Any use or disclosure of these materials to third party vendors or any other entity outside of the Visa membership association requires that such third party or entity first obtain a license from Visa.

The 3-D Secure Publication Suite Master License Agreement which governs such third party use is available through the "Vendors & Merchants" link on http://corporate.visa.com.

Preface

3-D Secure Introduction and System Overview

A full set of documentation has been developed for 3-D Secure™. The primary sources for introductory and general information are:

3-D Secure: Introduction, Visa Publication 70001-01

3-D Secure: System Overview, Visa Publication 70015-01

If you have not yet read those documents, you are encouraged to do so. The documents are available through the “Vendors & Merchants” link on http://corporate.visa.com.

Table of Contents

Table of Figures

Errata as of January 20, 2004

Table of Tables

Chapter 1: Document Overview

Introduction

Purpose

Payment authentication is the process of verifying cardholder account ownership during a purchase transaction in the remote environment.

Visa has developed the Three-Domain Secure (3-D Secure™) protocol to improve transaction performance online and to accelerate the growth of electronic commerce. The objective is to benefit all participants by providing issuers with the ability to authenticate cardholders during an online purchase, thus reducing the likelihood of fraudulent usage of payment cards and improving transaction performance.

This document describes 3-D Secure, including the messages supporting payment authentication.

As described in this document, 3-D Secure defines a base level of security. Enhancements will be included in later versions.

Intended audience

This document is intended for the use of vendors and Members that are interested in developing 3-D Secure products and supporting 3-D Secure implementations.

Differences between 1.0.1 and 1.0.2

The major changes included in 3-D Secure version 1.0.2 are: *1.0.2*

- Support is provided for generating a proof of authentication attempt when authentication is not available.
- The Account Identifier in the Verify Enrollment Response and Payer Authentication Request must not be the PAN.
- The PAN value that is included in the Payer Authentication Response must be masked.
- Adjusted DTD to indicate that **Serial Number** is optional in the Card Range Request; it is omitted when **Invalid Request Code** is included.

For details, see the Revision Log on page 149.

All changes specific to Version 1.0.2 are boxed as shown here, with the notation “1.0.2” in the right-hand margin. Where necessary, any requirements for downward compatibility with deployed 1.0.1 components are shown.

Latest changes

This document was published with errata on January 16, 2003, and is being republished with additional errata on January 20, 2004.

Revision marks from the January 2003 publication have been removed from the body of the book; however, margin notations indicate where those errata occurred, and the January 2003 revisions are shown in the Errata listing on page 151.

Revision marks for subsequent changes occur in the body of the book as well as in the Errata listing.

Future considerations

The base 3-D Secure protocol has been designed for the current business and market environment. In the future, enhancements to this protocol may be defined in order to properly support the requirements that arise from additional market and business opportunities.

Possible future enhancements include adding signatures to **PAReq** and **VEReq** to provide for more robust merchant authentication.

Errata

From time to time, errata will be published for these specifications. You should ensure that you have all of the relevant errata in addition to this document.

Document word usage

The following words are used often in this document and have specific meanings:

Must	Describes a product or system capability that is required, compelled, or mandatory.
Should	Describes a product or system capability that is highly recommended.
May	Describes a product or system capability that is optional.

References

Introductory information

The following documents provide fundamental information about 3-D Secure and are available through the "Vendors & Merchants" link on http://corporate.visa.com. You are encouraged to read them (in the following order) before reading this document:

3-D Secure: Introduction, Visa Publication 70001-01

3-D Secure: System Overview, Visa Publication 70015-01

3-D Secure specifications

Every 3-D Secure component must comply with the requirements of the core protocol, described in this document:

3-D Secure: Protocol Specification – Core Functions, Visa Publication 70000-01 (licensed)

In addition, each 3-D Secure component must comply with the functional requirements defined for that component:

3-D Secure: Functional Requirements – Access Control Server, Visa Publication 70002-01 (licensed)

3-D Secure: Functional Requirements – Merchant Server Plug-in, Visa Publication 70003-01 (licensed)

3-D Secure: Functional Requirements – Directory Server, Visa Publication 70025-01 (licensed)

Every 3-D Secure issuer component must also comply with the security requirements described in:

3-D Secure: Security Requirements – Enrollment and Access Control Servers, Visa Publication 70016-01 (licensed)

Any 3-D Secure component that supports mobile devices must also comply with the specifications for the supported extension:

3-D Secure: Protocol Specification – Extension for Mobile Internet Devices, Visa Publication 70006-01 (licensed)

3-D Secure: Protocol Specification – Extension for Voice and Messaging Channels, Visa Publication 70004-01 (licensed)

Throughout this publication suite, the requirements described in those documents are collectively referred to as "the 3-D Secure specifications."

Licensed documents

Some 3-D Secure documents are available only to parties that have executed with Visa a 3-D Secure Publication Suite Master License Agreement. Information regarding these publications and the license agreement is available through the "Vendors & Merchants" link on http://corporate.visa.com.

Compliance testing

All software components that are developed as 3-D Secure solutions are required to demonstrate compliance with Visa International's 3-D Secure requirements.

This testing must be completed successfully and the component must receive acknowledgement of compliance from Visa International before any claims of 3-D Secure compliance may be made. For details, see:

> **3-D Secure: Compliance Testing Facility – Policies & Procedures**, Visa Publication 70017-01

Other 3-D Secure documents

The following additional 3-D Secure documents are referenced in this specification and/or provide useful supplementary information:

> **3-D Secure: Implementation Guide – Issuer**, Visa Publication 70013-01
>
> **3-D Secure: Implementation Guide – Acquirer**, Visa Publication 70014-01
>
> **3-D Secure: Implementation Guide – Merchant**, Visa Publication 70020-01

Other documents

The following documents are referenced in this specification and/or provide useful background information:

1) **Certificate Infrastructure Group Brand Certificate Authority – Operating Procedures**, version 1.0, dated August 1999.

2) **Certificate Infrastructure Group Brand Certificate Authority – Business Policies and Procedures**, version 1.0, dated August 1999.

3) **Extensible Markup Language (XML), W3C Recommendation**, version 1.0 (Second Edition), dated 6 October 2000, available at http://www.w3.org/TR/2000/REC-xml-20001006.

4) **Canonical XML, W3C Recommendation**, version 1.0, dated 15 March 2001, available at http://www.w3.org/TR/2001/REC-xml-c14n-20010315.

5) **XML-Signature Syntax and Processing, W3C Recommendation**, dated 12 February 2002, available at http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/ or http://www.ietf.org/rfc/rfc3275.txt.

6) **Namespaces in XML, W3C Recommendation**, dated 14 January 1999, available at http://www.w3.org/TR/1999/REC-xml-names-19990114.

7) **The TLS [Transport Layer Security] Protocol**, Version 1.0, dated January 1999, available at http://www.ietf.org/rfc/rfc2246.txt.

8) **Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies**, available at http://www.ietf.org/rfc/rfc2045.txt, includes the algorithm for calculating a Base64 result.

9) **Hypertext Transfer Protocol – HTTP/1.1**, available at http://www.ietf.org/rfc/rfc2068.txt, discusses chunked transfer coding.

Document Organization

The document is organized as follows:

- Chapter 1, this chapter, contains an overview of the document and a list of references.
- Chapter 2 provides an entity overview.
- Chapter 3 lists and describes the technical requirements for 3-D Secure.
- Chapter 4 provides an overview of issuer setup and cardholder enrollment functions.
- Chapter 5 describes the design of the purchase transaction.
- Chapter 6 describes the 3-D Secure messages.
- Appendix A includes the XML message format of the 3-D Secure messages, as well as diagrams of the messages.
- Appendix B provides a single sorted list of the fields in all the core 3-D Secure messages.
- Appendix D describes the method for compressing data that is sent from one entity to another through the cardholder browser.
- A glossary defines selected terms and acronyms related to 3-D Secure.
- The revision log summarizes the changes in each production version of the document.

Chapter 2: Entity Overview

Organization

This chapter describes the systems and functions necessary to implement 3-D Secure. Descriptions are divided according to domain:

Issuer Domain	Systems and functions of the issuer and its customers (cardholders)
Acquirer Domain	Systems and functions of the acquirer and its customers (merchants)
Interoperability Domain	Systems, functions, and messages that allow Issuer Domain systems and Acquirer Domain systems to interoperate worldwide

Note that third parties may operate many of the systems in the Issuer and Acquirer Domains on behalf of Visa Members.

Issuer Domain

Cardholder	The cardholder shops online, providing the account holder name, card number, and expiration date, either directly or via software such as a digital wallet, then indicates readiness to finalize the transaction. In response to the Authentication Request Page, the cardholder provides information needed for authentication, such as a password.
Cardholder browser	The cardholder browser acts as a conduit to transport messages between the Merchant Server Plug-in (in the Acquirer Domain) and the Access Control Server (in the Issuer Domain).
Additional cardholder components	Optional cardholder hardware and software may supplement the abilities of the browser. For example, chip card implementations will require additional cardholder software and a card reader. Implementations that use passwords to authenticate cardholders using PCs should not require any additional cardholder hardware or software.
Issuer	A Member financial institution that: • enters into a contractual relationship with the cardholder for issuance of one or more payment cards • determines the cardholder’s eligibility to participate in the 3-D Secure service • defines card number ranges eligible to participate in the 3-D Secure service • provides data about those card number ranges to the Directory Server • performs enrollment of the cardholder for each payment card account (via the Access Control Server, a separate Enrollment Server, or manually)

Table 1: Issuer Domain Entities

Errata as of January 20, 2004

Access Control Server

The Access Control Server (ACS) has two functions:

- to verify whether 3-D Secure authentication (or proof of attempted authentication) is available for a particular card number and device type
- to authenticate the cardholder for a specific transaction or to provide proof of attempted authentication when authentication is not available

Although these functions are described as belonging to a single logical ACS, implementations may divide the processing by function or by other characteristics such as card number range among multiple physical servers.

Table 1: Issuer Domain Entities, continued

Acquirer Domain

Merchant	Existing merchant software handles the shopping experience, obtains the card number, then invokes the Merchant Server Plug-in to conduct payment authentication. After payment authentication, the merchant software may submit an authorization request to the acquirer, if appropriate.
Merchant Server Plug-in	The Merchant Server Plug-in (MPI) creates and processes payment authentication messages, then returns control to the merchant software. As part of processing the authentication response message from the issuer, the MPI may validate the digital signature in the message; alternatively, this function may be performed by a separate server, the acquirer, or a third party.
Validation Process	This function validates the signature received in the message from the ACS to the merchant. (This process may be implemented as an integral part of the Merchant Server Plug-in, or as a separate server. In the latter case, the acquirer may operate it on behalf of multiple merchants.)
Acquirer	A Member financial institution that: • enters into a contractual relationship with a merchant for purposes of accepting payment cards • determines the merchant's eligibility to participate in the 3-D Secure service Following payment authentication, the acquirer performs its traditional role: • receives authorization requests from the merchant • forwards them to the authorization system (such as VisaNet) • provides authorization responses to the merchant • submits the completed transaction to the settlement system (such as VisaNet)

Table 2: Acquirer Domain Entities

Interoperability Domain	**Directory Server**	The Directory Server, operated by each participating Payment Scheme: • receives messages from merchants querying a specific card number • determines whether the card number is in a participating card range • directs the request for cardholder authentication to the appropriate ACS (which may or may not provide Attempts functionality) or responds directly to the merchant • receives the response from the ACS indicating whether payment authentication (or proof of attempted authentication) is available for the cardholder account • forwards the response to the merchant
	Commercial Certificate Authority	Generates selected certificates for the use of 3-D Secure entities, including: • TLS/SSL client and server certificates
	Scheme Certificate Authority	Generates selected certificates for the use of 3-D Secure entities, including: • signing certificates • Root certificate required by the Payment Scheme
	Authentication History Server	The Authentication History Server, operated by each participating Payment Scheme: • receives a message from the ACS for each attempted payment authentication (whether or not authentication was successful) • stores the records received A copy of the data stored by the Authentication History Server is available to acquirers and issuers in case of disputes. Additional information describing how to populate this database is provided in **3-D Secure: Functional Requirements – Access Control Server**.

Table 3: Interoperability Domain Entities

Authorization system	Following payment authentication, the authorization system (such as VisaNet) performs its traditional role: • receives authorization requests from the acquirer • forwards them to the issuer • provides responses from the issuer to the acquirer • provides clearing and settlement services to the acquirer and issuer

Table 3: Interoperability Domain Entities, continued

Chapter 3: Technical Requirements

Organization

The technical requirements for 3-D Secure have been grouped into the following categories:

Message names

The following messages, which are discussed in detail in subsequent chapters, are occasionally mentioned in this one:

CRReq	**CRRes**	Card Range Request and Response
VEReq	**VERes**	Verify Enrollment Request and Response
PAReq	**PARes**	Payer Authentication Request and Response[1]
Error		Error message

An additional message pair, used to populate the Authentication History Server, is mentioned only briefly in this document, and documented in detail in **3-D Secure: Functional Requirements – Access Control Server**:

PATransReq **PATransRes**	Payer Authentication Transaction Request and Response

Table 4: 3-D Secure Messages

[1] Implementations that support mobile Internet devices use the Condensed Payer Authentication Request and Response messages, **CPRQ** and **CPRS**. There are significant differences in transporting and processing these messages, as well as differences in the associated **VEReq** and **VERes** messages. For details, see **3-D Secure: Protocol Specification – Extension for Mobile Internet Devices**.

Transport Security Requirements

Channel encryption

~~The channels listed in Table 5 must be encrypted using 128-bit SSL cipher suite(s).~~ The SSL servers described in Table 5 must be capable of initiating sessions using 128-bit (or stronger) cipher suites, with the exception of the merchant which should be capable of initiating such sessions if possible. Channels 1 and 2 must support the 40-bit SSL cipher suites, due to the proliferation of US-exportable browsers on cardholder systems.

For additional certificate requirements, see page 17 and the Functional Requirements documents listed in "3-D Secure specifications" on page 3.

1) Cardholder to Merchant	This channel is used: • for the cardholder to enter payment information • to transport the **PAReq** from the Merchant Server Plug-in to the cardholder • to transport the **PARes** from the cardholder to the Merchant Server Plug-in The merchant must secure this channel with an SSL session initiated using a server certificate.
2) Cardholder to Access Control Server (ACS)	This channel is used: • to forward the **PAReq** to the ACS • to receive the signed **PARes** from the ACS The ACS must secure this channel with an SSL session initiated using a server certificate. Processors operating an ACS on behalf of multiple issuers must be able to support using SSL server certificates specific to each issuer. The decision about whether to use multiple certificates in a given implementation will be made based on individual processor and issuer business requirements.

Table 5: Transport Security Requirements

3) Merchant to Directory Server	This channel is used to transport **VEReq**, **VERes**, **CRReq**, **CRRes**, and **Error**. The Directory Server must secure this channel with an SSL session initiated using a server certificate. The Directory Server must be able to authenticate the merchant using SSL client certificates (during session initiation). The actual use of SSL client certificates for authentication of the **VEReq** will depend on specific regional requirements, but both systems (Directory Server and merchant) must be capable of supporting client authentication.
4) Directory Server to Access Control Server	This channel is used to transport the **VEReq, VERes**, and **Error** messages. The ACS must secure this channel with an SSL session initiated using a server certificate and a client certificate for the Directory Server.
5) Merchant to Validation Process	When the validation process is implemented as a separate server, this channel is used to transport the **PARes** (for validation) and the server's response. The validation process server must secure this channel with an SSL session initiated using a server certificate. The validation process server must authenticate the merchant initiating the session; it may do so using SSL client certificates or using another mechanism selected by the acquirer.
6) Merchant to Access Control Server	This channel is used to transport the **Error** message. The ACS must secure this channel with an SSL session initiated using a server certificate.

Table 5: Transport Security Requirements, continued

SSL cipher suites

The security protocol used at the transport layer for 3-D Secure is the Transport Layer Security Protocol (TLS) defined in the IETF RFC 2246. This protocol standard is based on the Netscape Secure Sockets Layer V3 (SSL).

TLS sessions for 3-D Secure must use one of the cryptographic suites listed in Table 6. The cryptographic suite listed as *required* must be supported; the suite listed as *optional* may also be supported.

Note: If a session cannot be established using TLS, 3-D Secure components may attempt to establish a session using SSL version 3.

Servers accepting connections from the browser (ACS and MPI) may use US-exportable cipher suites to establish connections with the browser, but should still attempt to establish as strong a connection as possible.

TLS_RSA_WITH_3DES_EDE_CBC_SHA	required
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	optional

Table 6: Cryptographic Suites

Certificate Requirements

Table 7 lists the certificates that are necessary to implement 3-D Secure.

For more information regarding the Visa Certificate Authority Key Management Policies, please see the Certificate Infrastructure Group documents described on page 4.

Note: The Visa Root certificate is a self-signed certificate.

Entity	Purpose	Certificates Required
Merchant Server Plug-in	To authenticate the merchant to the Directory Server and an optional Validation Process Server	SSL client certificate Root certificate required by the Payment Scheme and all other certificates needed to validate the client certificate
	To protect the cardholder's PAN data and the **PAReq** and **PARes** data	SSL server certificate Root certificate required by the Payment Scheme and all other certificates needed to validate the server certificate, if it was issued under that Root
Directory Server	To protect the **VEReq** and **VERes** data	SSL server certificate Root certificate required by the Payment Scheme and all other certificates needed to validate the server certificate
	To authenticate the Directory Server to the ACS	SSL client certificate Root certificate required by the Payment Scheme and all other certificates needed to validate the client certificate
Access Control Server	To protect the **VEReq**, **VERes**, **PAReq**, and **PARes** data	SSL server certificate Root certificate required by the Payment Scheme and all other certificates needed to validate the server certificate, if it was issued under that Root Note: Processors operating an ACS on behalf of multiple issuers must be able to use a different SSL server (encryption) certificate for each issuer.
	To sign the **PARes** message	Signing certificate (issued to issuer) Root certificate required by the Payment Scheme and all other certificates needed to validate the signing certificate Note: Processors operating an ACS on behalf of multiple issuers must be able to use a different signing certificate for each issuer.
Validation Process Server	To validate the **PARes** message signature	Root certificate required by the Payment Scheme and all other certificates needed to validate the server certificate

Table 7: Certificate Requirements

Redundant Routing Requirements

Applications must support multiple path routing to the Directory Server and Access Control Server.

Directory Server	When the Directory Server converts a domain name into an IP address, it must retrieve all registered addresses from the domain name server and attempt a connection with the alternate address(es) if a session cannot be established with the primary address.
Merchant Server Plug-in	When the Merchant Server Plug-in establishes a session with the Directory Server and converts a domain name into an IP address, it must retrieve all registered addresses from the domain name server and attempt a connection with the alternate address(es) if a session cannot be established with the primary address.

Table 8: Redundant Routing Requirements

HTTP Connections

Persistent sessions

3-D Secure components may use *HTTP Keep Alive* to establish persistent sessions with other 3-D Secure components.

Errata as of January 20, 2004

Chapter 4: Setup and Cardholder Enrollment

Organization

This chapter outlines a model implementation of the setup and cardholder enrollment processes. Since the cardholder enrollment process is entirely within the Issuer Domain, alternate implementations are possible. The topics discussed in this chapter are provided as background material for the reader.

The following topics are included:

Errata as of January 20, 2004

Process Flow Diagram

Figure 1 illustrates a possible architecture for cardholder enrollment.

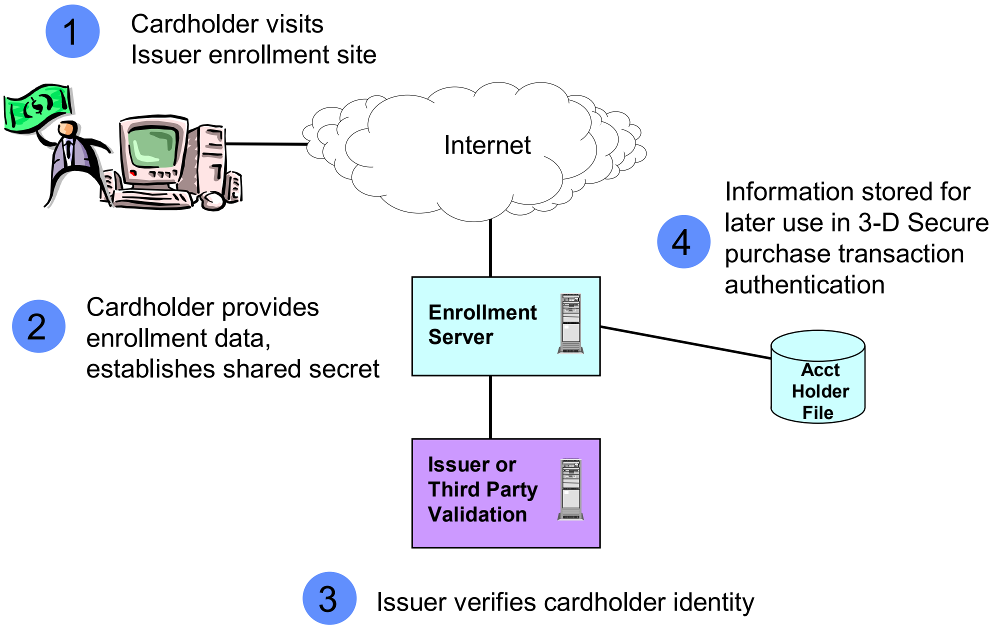


Figure 1: Sample Cardholder Enrollment Process

Process Flow – Issuer Setup

Issuer setup

1) Issuer loads its Enrollment Server with data necessary for 3-D Secure enrollment. Table 9 lists data that may be needed.

   a) Beginning of range (13 – 19 digits)

   b) End of range (13 – 19 digits)

   c) If applicable, the definitions of issuer-specified authentication data. (For example, “Place of Birth”.)

      If this option is selected, one (1) to several fields may be required.

   d) The issuer’s *Terms of Use* and *Data Privacy Policy*

   e) Authentication methods supported by issuer (depending on the implementation, these methods may be an integral part of the Enrollment Server or may be customizable by the issuer)

   f) Data needed by the authentication methods, such as conditions for approval of an enrollment request

   Depending on the implementation, additional data may be required. For example, a service operated on behalf of multiple issuers will also require information to identify the specific issuer that owns a particular card range.

   Table 9: Sample Enrollment Server Data

2) Issuer provides to Payment Scheme the data needed to load the Directory Server.

3) Payment Scheme updates the Directory Server.

Process Flow – Cardholder Enrollment

Cardholder enrollment

A possible cardholder enrollment process follows:

1) Cardholder visits the issuer's 3-D Secure enrollment webpage and is required to enter information such as that listed in Table 10.
2) The issuer displays its 3-D Secure *Terms of Use* and *Data Privacy Policy*, an ACCEPT button and a NOT ACCEPT button. If the cardholder selects ACCEPT, enrollment proceeds. If the cardholder selects NOT ACCEPT, continue with Step 7.
3) The issuer's 3-D Secure application validates that the PAN falls within a card range that is registered in the issuer's Enrollment Server. If the PAN is not within a defined range, the enrollment is rejected; continue with Step 7.
4) The issuer's 3-D Secure application displays an enrollment form to the cardholder.
5) The issuer matches the information entered by the cardholder against its own records.
6) If not successful, an appropriate message is displayed to the cardholder and the process continues with Step 4 (up to an issuer-specified number of failed attempts). If successful, the issuer updates the 3-D Secure Account Holder database. See sample data listed in Table 10.
7) The issuer displays an appropriate completion-of-enrollment message to cardholder.

a) PAN
b) Card Expiry Date
c) Cardholder Name
d) E-mail Address
e) Personal Assurance Message (PAM); created by the user, displayed in the secret code prompt window to help reduce spoofing
f) Cardholder Password
g) Special question/hint
h) Special question/hint response
i) Issuer-specified authentication information

Table 10: Sample Cardholder Enrollment Data

Chapter 5: Purchase Transaction

Organization

This chapter describes the components that are invoked and the messages that are exchanged in a 3-D Secure purchase transaction. The following topics are included:

Purchase Transaction Architecture and Message Flows

Figure 2 illustrates and page 25 describes the steps in the purchase transaction flow.

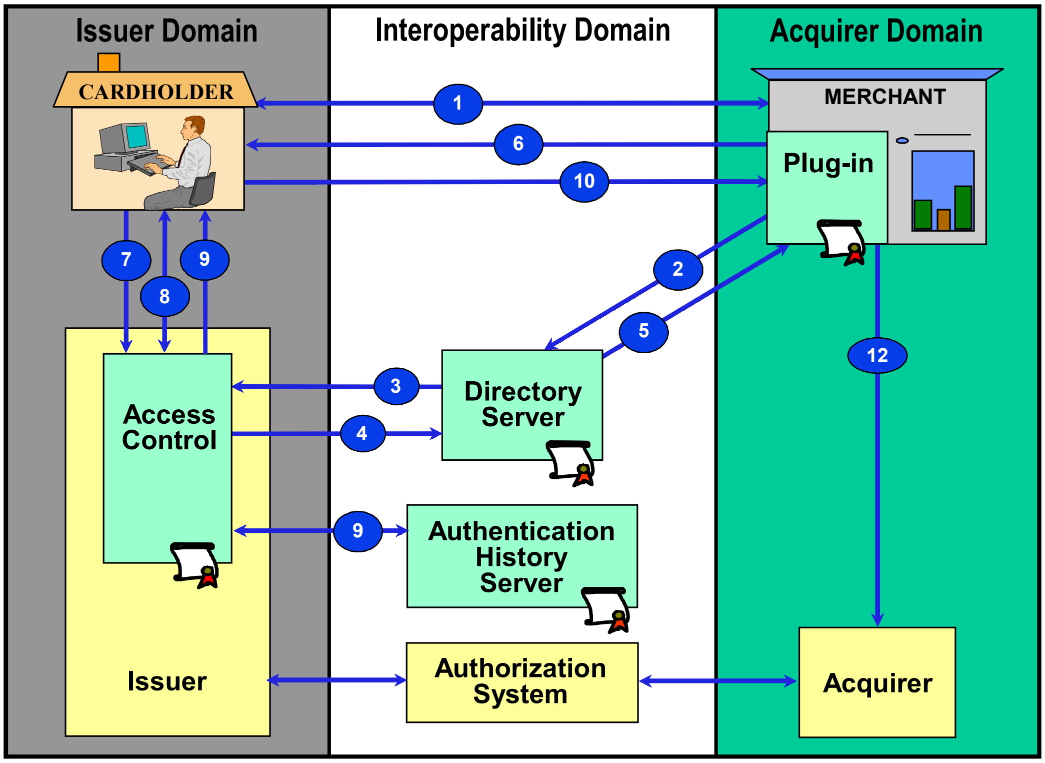


Figure 2: Sample Purchase Transaction

Step 1	Shopper browses at merchant site, adds items to shopping cart, then finalizes purchase. (Note: Merchant now has all necessary data, including PAN and user device information.)
Step 2	Merchant Server Plug-in (MPI) sends PAN (and user device information, if applicable) to Directory Server.
Step 3	Directory Server queries appropriate Access Control Server (ACS) to determine whether authentication (or proof of authentication attempt) is available for the PAN and device type. (If no appropriate ACS is available, the Directory Server creates a response for the MPI and processing continues with Step 5.)
Step 4	ACS responds to Directory Server.
Step 5	Directory Server forwards ACS response (or its own) to MPI. If neither authentication nor proof of authentication attempt is available, 3-D Secure processing ends, and the merchant, acquirer, or payment processor may submit a traditional authorization request, if appropriate.
Step 6	MPI sends Payer Authentication Request[2] to ACS via shopper's device.
Step 7	ACS receives Payer Authentication Request.
Step 8	ACS authenticates shopper using processes applicable to PAN (password, chip, PIN, etc.). Alternatively, ACS may produce a proof of authentication attempt. ACS then formats Payer Authentication Response[3] message with appropriate values and signs it.
Step 9	ACS returns Payer Authentication Response to MPI via shopper's device. ACS sends selected data to Authentication History Server.
Step 10	MPI receives Payer Authentication Response.
Step 11	MPI validates Payer Authentication Response signature (either by performing the validation itself or by passing the message to a separate Validation Server).
Step 12	Merchant proceeds with authorization exchange with its acquirer.
	Following Step 12, acquirer processes authorization with issuer via an authorization system such as VisaNet, then returns the results to merchant.

Table 11: Sample Purchase Transaction

[2] The Payer Authentication Request message may be **PAReq** (for cardholders using PCs) or **CPRQ** (for cardholders using mobile Internet devices – see **3-D Secure: Protocol Specification – Extension for Mobile Internet Devices**).

[3] The Payer Authentication Response message is **PARes** if **PAReq** was received, or **CPRS** if **CPRQ** was received. (**CPRS** is created using values from the **PARes**.)

Purchase Transaction Process Flow Description

Merchant Server Plug-in – load cache

Step 0.

To eliminate the need to query the Directory Server (DS) for each purchase transaction (in Step 2), the merchant should have the ability to copy the contents of the DS into a local cache. If this capability is used, the merchant can determine immediately from the cache if the account is part of an enrolled range. If the merchant implements this capability, the contents of the cache must expire and be refreshed at least every 24 hours; the cache should be requested when the Merchant Server Plug-in is loaded and at the same time each day that follows. The request time must not be hard-coded. (This will help ensure that every Merchant Server Plug-in (MPI) is not requesting the card range updates simultaneously.)

a) The MPI formats a Card Range Request (**CRReq**) message (as described in Table 18 on page 54) and sends it to the Directory Server.

   If this is the first time the cache is being loaded (or if the cache has been flushed and needs to be reloaded), the **Serial Number** element is not included in the request, which will result in the DS returning the entire list of participating card ranges.

   Otherwise, the MPI should include the **Serial Number** from the most recently processed Card Range Response (**CRRes**), which will result in the DS returning only the changes since the previous **CRRes**.

b) The Directory Server validates the syntax of the **CRReq** (as described in Table 18 on page 54) and returns an **Error** if validation fails.

   The DS authenticates the merchant as described for **VEReq** in Step 3a. If authentication fails, the DS formats a **CRRes** with **Invalid Request Data** set to the corresponding value from Table 25 on page 95. ***errata 9***

   The DS formats a **CRRes** (as described in Table 19 on page 58) containing the participating ranges and sends it to the MPI. If the **CRReq** includes a value for **Serial Number**, the DS returns only those updates made since that value of **Serial Number** was current.

   The DS includes a serial number in the response that defines the current state of the card range database. (The specific value is meaningful only to the DS that returns it.)

Errata as of January 20, 2004

Merchant Server Plug-in – load cache, continued

Step 0, continued.

c) The MPI validates the syntax of the **CRRes** (as described in Table 19 on page 58) and should send an **Error** to the Directory Server if validation fails.

The MPI updates its cache. The list must be processed in the order returned with ranges being added or deleted as indicated by the **Action** element. The MPI should retain the **Serial Number** value to be included with the next day's **CRReq** message.

Note: If **CRRes** indicates any error condition, the MPI should clear its cache and submit the **CRReq** without a **Serial Number** element. ***1.0.2***

Cardholder – initiate purchase transaction

Step 1.

a) Cardholder visits a merchant Web site via a browser and selects items to purchase.

b) The cardholder checks out and finalizes the purchase transaction. At this point, the merchant has all the information required to process the purchase transaction, including the PAN, expiration date, and address information.

The PAN must be provided to the merchant during the checkout process either via cardholder keyboard entry or a wallet function. The expiration date must be no earlier than the current month.

Merchant Server Plug-in – query PAN enrollment

Step 2.

This process occurs after the final "Buy" confirmation from the cardholder during the merchant checkout process. The merchant software invokes the Merchant Server Plug-in (MPI) to determine whether payment authentication is available for the PAN and user device information.

a) If the Merchant Server Plug-in (MPI) has implemented caching (as discussed in "Merchant Server Plug-in – load cache" on page 26), it checks its cache of participating card number ranges.

If the PAN is **not** in one of the ranges, continue with Step 12 on page 42.

b) The MPI formats a Verify Enrollment Request (**VEReq**) message (as described in Table 20 on page 63).

c) The MPI determines whether it currently has a secure connection with the Directory Server.

If not, the MPI establishes an SSL connection with the Directory Server. If the Directory Server configuration indicates that the merchant has been issued an SSL client certificate, it will require the merchant to present it during the establishment of the SSL session.

If connection cannot be established or if authentication fails, continue with Step 12 on page 42.

d) The MPI posts the **VEReq** to the Directory Server.

Directory Server – query ACS

Step 3.

This process occurs immediately after the Directory Server receives the **VEReq** from the Merchant Server Plug-in.

a) The Directory Server validates the syntax of the **VEReq** (as described in Table 20 on page 63) and returns an **Error** if validation fails.

The Directory Server validates the **VEReq** data, ensuring that each of the following is true:

- The **Acquirer BIN** represents a participating acquirer.
- The endpoint submitting the transaction is a valid merchant endpoint. The **Merchant ID** may be used to perform this validation, by ensuring that it represents a participating merchant of the acquirer identified by **Acquirer BIN**. *errata 10*
- If the Visa region of the acquirer requires a merchant password for 3-D Secure:
  - a value for **Password** was received, and
  - it is valid for the combination of **Acquirer BIN** and **Merchant ID**.

If any of these tests fails:

- The Directory Server formats a Verify Enrollment Response (**VERes**) including:
  - **PAN Authentication Available** set to "N"
  - no **Account Identifier**, **ACS URL** or **Payment Protocols**
  - **Invalid Request Data** set to the corresponding value from Table 25 on page 95.
- The Directory Server returns the **VERes** to the Merchant Server Plug-in and stops.

Directory Server – query ACS, continued

Step 3, continued.

b) The Directory Server searches for a record specifying a card range that includes the **Cardholder PAN** that was received in the **VEReq**.

If not found:

- The Directory Server formats a Verify Enrollment Response (**VERes**) message (as described in Table 21 on page 69) including:
  - **PAN Authentication Available** set to "N"
  - no **Account Identifier**, **ACS URL**, **Payment Protocols**, or **Invalid Request Data**
- The Directory Server returns **VERes** message to the Merchant Server Plug-in and stops.

c) The Directory Server determines whether it currently has a secure connection with the ACS.

If not, then the Directory Server establishes an SSL connection with the ACS. The SSL client certificate of the Directory Server and the server certificate of the ACS must be presented and validated during the establishment of the SSL session.

If the Directory Server maintains alternate URLs for the ACS, and if the first URL attempted is not available, the Directory Server will attempt to connect to one of the alternate URLs.

If unsuccessful on all attempts:

- The Directory Server formats a Verify Enrollment Response (**VERes**) (as described in Table 21 on page 69) message including:
  - **PAN Authentication Available** set to "N"
  - no **Account Identifier**, **ACS URL**, **Payment Protocols**, or **Invalid Request Data**
- The Directory Server returns **VERes** message to the Merchant Server Plug-in and stops.

d) The Directory Server removes the **Password** from the **VEReq** message and forwards the message to the ACS URL.

Access Control Server – verify enrollment

Step 4.

This process occurs immediately after the ACS receives the **VEReq** message via the Directory Server. The ACS validates the syntax of the **VEReq** (as described in Table 20 on page 63) and returns an **Error** if validation fails.

> Note: When it is not possible to authenticate a payment transaction, it is sometimes possible to provide a proof of authentication attempt instead. Such processing significantly changes the ACS processing logic, and is described in **3-D Secure: Functional Requirements – Access Control Server**. ***1.0.2***

a) The ACS uses the **Cardholder PAN** from the **VEReq** message and queries the Account Holder database to determine whether the cardholder is enrolled.

b) If the PAN was not found, the ACS formats a Verify Enrollment Response (**VERes**) message (as described in Table 21 on page 69) including:

   - **PAN Authentication Available** set to "N"
   - no **Account Identifier**, **ACS URL**, **Payment Protocols**, or **Invalid Request Data**

   Continue with Step 4f below.

c) If **Device Category** is present:

   - If the ACS cannot process transactions sent by the device category indicated or if the ACS does not recognize the device category value, the ACS formats a Verify Enrollment Response (**VERes**) message including:
     - **PAN Authentication Available** set to "U"
     - no **Account Identifier**, **ACS URL**, **Payment Protocols**, or **Invalid Request Data**

     Continue with Step 4f below.

d) If either **Accept Headers** or **User Agent** is present:

   - If the ACS cannot process transactions sent by the cardholder device or the user agent indicated by the values of those elements, the ACS formats a Verify Enrollment Response (**VERes**) message including:
     - **PAN Authentication Available** set to "U"
     - no **Account Identifier**, **ACS URL**, **Payment Protocols**, or **Invalid Request Data**

     Continue with Step 4f below.
   - If special processing is required, continue processing as described in the appropriate document (for example, the mobile Internet extension described in "3-D Secure specifications" on page 3).

Access Control Server – verify enrollment, continued

Step 4, continued.

e) The ACS formats a Verify Enrollment Response (**VERes**) message (as described in Table 21 on page 69) including:

- **PAN Authentication Available** set to "Y"
- an **Account Identifier** that the ACS internally associates with the PAN (note that this identifier must not be the PAN). (MPI developers should be aware that the **Account Identifier** in a Version 1.0.1 **VERes** may contain the original PAN, rather than a unique identifier). *1.0.2*
- the **ACS URL** to be used to transmit the **PAReq**
- **Payment Protocols** set as defined on page 71
- no **Invalid Request Data**

f) The ACS sends the **VERes** message to the Directory Server.

Directory Server – forward response

Step 5.

From the point of view of the Directory Server, this process occurs immediately after the Directory Server forwards the **VEReq** message to the ACS URL. (From the point of view of the ACS, it occurs immediately after the ACS sends the **VERes** message to the Directory Server.)

a) The Directory Server reads the response, which contains the corresponding **VERes** or **Error**. If a **VERes** was received, the Directory Server validates its syntax (as described in Table 21 on page 69) and returns an **Error** to the ACS if validation fails.

b) If the message received from the ACS is syntactically correct, the Directory Server forwards the **VERes** or **Error** to the MPI.

c) If the message received from the ACS is not syntactically correct:

- The Directory Server formats a Verify Enrollment Response (**VERes**) message (as described in Table 21 on page 69) including:
    - **PAN Authentication Available** set to “N”
    - no **Account Identifier**, **ACS URL**, **Payment Protocols**, or **Invalid Request Data**
- The Directory Server returns the **VERes** message to the Merchant Server Plug-in and stops.

Merchant Server Plug-in – receive response

Step 5, continued.

From the point of view of the Merchant Server Plug-in, this process occurs immediately after the MPI posts the **VEReq** to the Directory Server. (From the point of view of the Directory Server, it occurs immediately after the Directory Server forwards the **VERes** to the MPI.)

a) The MPI reads the response, which contains the corresponding **VERes** or **Error**.

b) If an **Error** message is received, continue with Step 12 on page 42.

Merchant Server Plug-in – send payer authentication request

Step 6.

This process occurs immediately after the Merchant Server Plug-in (MPI) receives the **VERes** message from the Directory Server. The MPI validates the syntax of the **VERes** (as described in Table 21 on page 69) and should send an **Error** to the Directory Server if validation fails.

a) If the value of **PAN Authentication Available** is not equal to "Y", continue with Step 12 on page 42.

~~b)Examine the **Payment Protocols** and select the desired protocol to be used. If that protocol is "ThreeDSecure", continue processing; if it is a protocol other than "ThreeDSecure", execute the appropriate processing for the selected protocol.~~

~~Note: The protocol selection is made based on the capabilities of the merchant systems as well as region, acquirer, and merchant policies.~~

errata 42

b) The MPI formats a Payer Authentication Request (**PAReq**) message (as described in Table 22 on page 75) including the **Account Identifier** received in **VERes**.

c) The MPI deflates and Base64-encodes the **PAReq** as described in Appendix D on page 139. The result is referred to as **PaReq** (note the case difference).

Merchant Server Plug-in – send payer authentication request, continued

Step 6, continued.

d) The MPI constructs a form containing the following fields:

PaReq (note case)	The result of Step 6d.
TermUrl	The merchant URL to which the final reply must be posted.
MD	The **MD** ("Merchant Data") field: merchant state data that must be returned to the merchant. This field is used to accommodate the different ways merchant systems handle session state. If the merchant system can associate the final post with the original shopping session without any further assistance, the **MD** field may be empty. If the merchant system does not maintain state for a given shopping session, the **MD** can carry whatever data the merchant needs to continue the session. Since the content of this field varies by merchant implementation, the ACS must preserve it unchanged and without assumptions about its content. This field must contain only ASCII characters in the range 0x20 to 0x7E; if other data is needed, the field must be Base64 encoded. The size of the field (after Base64 encoding, if applicable) is limited to 1024 bytes. If **MD** includes confidential data (such as the PAN), it must be encrypted.

Table 12: Fields in Form MPI Sends to ACS

e) The MPI passes the **PAReq** through the cardholder browser to the **ACS URL** received in the **VERes**, by causing the cardholder browser to POST the form to the ACS. This is typically accomplished by using JavaScript. (For further detail, see **3-D Secure: Functional Requirements – Merchant Server Plug-in**.) All connections are HTTPS to accommodate the cardholder browser.

Access Control Server – receive payer authentication request

Step 7.

This process occurs immediately after the ACS receives the post including the **PAReq** from the Merchant Server Plug-in. The following description applies to the case where cardholder authentication is performed using a password. Other methods, such as those that rely on applications on a chip card, may be used.

a) The ACS Base64-decodes and inflates the **PaReq** field reversing the process described in Appendix D on page 139 to obtain the **PAReq** (note the case difference). The ACS validates the syntax of the **PAReq** (as described in Table 22 on page 75) and returns an **Error** if validation fails.

The ACS validates the **PAReq** data, ensuring that each of the following is true:

- The ACS is able to link the **PAReq** with a **VERes** in which the value of **PAN Authentication Available** was “Y”.

  The validation may take place through whatever mechanism the ACS chooses, such as by comparing the **Account Identifier** supplied in the two messages or by correlating the URL to which the message was posted to a customized **ACS URL** supplied in **VERes**.

- The **Merchant Country Code** is a valid ISO 3166 Country Code.
- The **Purchase Currency** is a valid ISO 4217 numeric currency code.

errata 7

If any of these tests fails:

- The ACS formats a Payer Authentication Response (**PARes**) message (as described in Table 24 on page 86) with **Transaction Status** set to ~~“N”~~ “U” and **Invalid Request Data** set to appropriate values as outlined in Table 25 on page 95.[4]
- Continue with Step 8f on page 39.

errata 30

[4] The protocol specification previously permitted use of the **IReq** element in a **PARes** message with a **Transaction Status** of either “N” or “U”. In order to ensure interoperability, MPI developers must continue to permit either value to accompany the **IReq** element.

Access Control Server – authenticate shopper

Step 8.

a) If proof of authentication attempt will be generated, display a message such as "Processing…" and continue with Step 8d. *1.0.2*

The ACS responds to the post with an HTML page that displays a form to the cardholder, including items such as those listed in Table 13. (The specific contents of the form are the choice of the issuer, subject to any applicable regional requirements. See Figure 3 on page 38 for an example.)

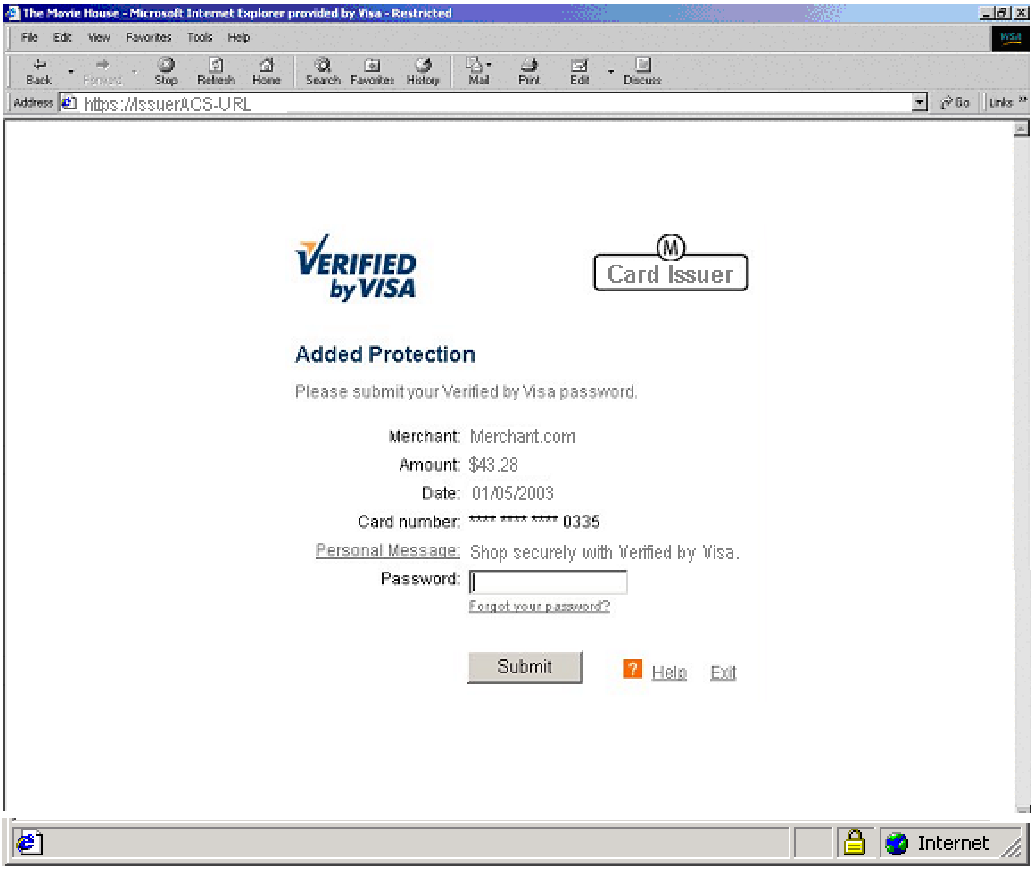
Data Item	From ACS	From **PAReq**
Member logo	X	
Visa Mark or Payment Scheme logo	X	
Merchant name		X
Total amount and currency Note: See "Displaying purchase amount" on page 83 for an explanation of how to display amount and currency.	X	
Merchant date (month, day and year)		X
PAN (xxx out except for last four digits) Note: The ACS needs to correlate the **Account Identifier** received in the **PAReq** with the actual PAN stored in the <u>Account Holder</u> database.	X	
Personal Assurance Message (PAM)	X	
Prompt for Cardholder Password	X	

errata 7

1.0.2

Table 13: Example Fields Displayed for Cardholder Authentication

Access Control Server – authenticate shopper, continued



Added Protection

Please submit your Verified by Visa password.

Merchant: Merchant.com
Amount: $43.28
Date: 01/05/2003
Card number: **** **** **** 0335
Personal Message: Shop securely with Verified by Visa.
Password:
Forgot your password?

Submit   Help   Exit

Figure 3: Sample Authentication Request Page

Access Control Server – authenticate shopper, continued

Step 8, continued.

b) The ACS prompts the cardholder to enter the password.

c) The ACS accepts the cardholder input and validates it against the Account Holder database.

d) The ACS sets **Transaction Status** as described in Table 14.

condition	value of **Transaction Status**
Customer was successfully authenticated.	Y
Customer failed or cancelled authentication. Transaction denied.	N
Authentication could not be completed, due to technical or other problem ~~(for example, because of the failure of an ACS system component such as the cardholder database)~~, as indicated in **PARes.IReq**.	U
Proof of authentication attempt was generated.	A

1.0.2

Table 14: ACS Sets Transaction Status

Access Control Server – prepare payer authentication response

Step 8, continued.

e) The ACS formats a Payer Authentication Response (**PARes**) message (as described in Table 24 on page 86), including **Transaction Status**.

If the transaction was authenticated successfully (that is, the value of **Transaction Status** is “Y”) or if a proof of authentication attempt was generated (that is, the value of **Transaction Status** is “A”), a **Cardholder Authentication Verification Value** (**cavv**) is generated and included in **PARes**. (See **3-D Secure: Functional Requirements – Access Control Server** for information about generating the **Cardholder Authentication Verification Value**.)

1.0.2

If the transaction was not authenticated successfully, the ACS must return all zeros in the **PAN** field.

f) The ACS digitally signs the message.

Access Control Server – return payer authentication response

Step 9.

a) The ACS deflates and Base64-encodes the **PARes** formatted and signed in Step 8 (or an **Error** message if one has been generated) as described in Appendix D on page 139. The result is referred to as **PaRes** (note case difference).

b) The ACS constructs a form containing the following fields:

PaRes (note case)	The result of the previous step.
MD	The exact content of the **MD** field as posted to the ACS, unchanged.

Table 15: Fields in Form ACS Sends to MPI

c) The ACS passes the signed **PARes** through the cardholder's browser to the merchant's URL (**TermUrl** in the post from the MPI) by causing the cardholder browser to POST the form to the MPI. In the process, control is returned to the merchant's browser window. This is typically accomplished by using JavaScript. (For further detail, see **3-D Secure: Functional Requirements – Access Control Server**.)

d) The ACS formats a Payer Authentication Transaction (**PATransReq**) message and sends it to the Authentication History Server. See **3-D Secure: Functional Requirements – Access Control Server** for the requirements for the data and for its transmission.

Errata as of January 20, 2004

Merchant Server Plug-in – validate payer authentication response

Step 10.

This process occurs after the Merchant Server Plug-in posts the **PAReq** to the ACS.

a) The MPI reads the response, which contains the **PaRes** field. The MPI Base64-decodes and inflates the **PaRes** field reversing the process described in Appendix D on page 139 to obtain the **PARes** (note the case difference) or **Error**.

b) If an **Error** message is received, continue with Step 12 on page 42.

c) The MPI validates the syntax of the **PARes** (as described in Table 24 on page 86) and should send an **Error** to the ACS (via the **ACS URL** received in the **VERes**) if validation fails.

Step 11.

The Validation Process validates the **PARes** signature using the Root Certificate required by the Payment Scheme. Note that this may either be an integral part of the Merchant Server Plug-in or may be implemented as an independent Validation Server.

Note: Digital signature validation must fully comply with the underlying digital signature specification. In particular, the **PARes** signature must be validated over the entire contents of the **SignedInfo** element, including any inter-element white space.

errata 28

If implemented using a separate Validation Server:

- The Merchant Server Plug-in sends the **PARes** to the Validation Process.
- Validation Process validates the signature on the **PARes** using the Root certificate required by the Payment Scheme.
- Validation Process returns the result of signature validation to the Merchant Server Plug-in.

Errata as of January 20, 2004

Payment authorization

Step 12.

This process occurs after the previous steps have been completed.

> The Merchant Server Plug-in notifies the merchant payment system of the results of the authentication attempt, and provides data needed for further processing.

errata 11

Electronic commerce processing

If authentication could not be performed, the merchant may proceed with a normal payment authorization using the available information from the checkout process. In this case, the merchant payment system must process the transaction as an unauthenticated electronic commerce transaction, which is out-of-scope to this document.

Note: The Electronic Commerce Indicator must be set to a value corresponding to the authentication results and the characteristics of the checkout process.

Alternate account

If the merchant is unable to process an authenticated transaction using the account selected by the cardholder during the checkout process, the merchant may either abandon the transaction or give the customer the option of selecting an alternate account.

If an alternate account is selected, the authentication process must be repeated starting with Step 1 for that account.

Chapter 6: Message Descriptions

Organization

This section describes the following messages and content:

Format

All listed messages are in XML format. For details, see Appendix A on page 103.

Inclusion values

In the tables that follow, the Inclusion column includes the values in the following table:

	Meaning	Sender requirements	Recipient requirements
R	Required	must include the field	must check for presence of the field and validate its contents
C	Conditional	must include the field if the conditions are satisfied	must: • check for presence of the field when the conditions are satisfied, and • validate its contents when present
O	Optional	may include the field	must validate the contents when present

Table 16: Field Inclusion Values

Errata as of January 20, 2004

Message Handling

Version numbers

A valid 3-D Secure version number will always be in the format *major.minor*[*.update*] (such as 1.0 or 1.0.1).

When the root element of a message is **ThreeDSecure**, any version number less than 1.0.1 is an error. The 3-D Secure component must return an **Error** message with an **Error Code** of 6.

Versioning and parsing

In order to support future versions of the protocol, implementations must use (or configure) XML parsers that do not validate strictly. ~~Specifically, unrecognized elements in a message must be silently ignored (except for an unrecognized **Extension** element that has a **critical** attribute with a value of "true").~~ In particular: *errata 1*

- The ACS must silently ignore unrecognized elements in the **VEReq** message. *errata 51*
- All entities must silently ignore unrecognized non-critical **Extension** elements (that is, any **Extension** element that does not have a **critical** attribute with a value of "true").
- All entities must silently ignore unrecognized child elements of any non-critical **Extension** element.

Implementations built to this specification must support messages having the following version numbers: ***1.0.2***

- 1.0.1
- 1.0.2

If a requirement applies only to Version 1.0.2, it is marked with the notation "1.0.2" in the right-hand margin. Any requirements for implementing downward compatibility with Version 1.0.1 are defined within this specification.

In addition, components must deal with other version numbers (for example, a component built to these specifications must deal with any version 1.1 message it receives) as described in Table 17 beginning on page 45.

Entity	Message Received	If version number is:	then:
Directory Server[5]	**CRReq**	higher than DS supports[6]	If the message contains any unrecognized **Extension** element that has an attribute of **critical** with a value of "true", return an **Error** message (using the highest supported message version) with the value of **errorCode** set to 4 (critical element not recognized). Otherwise, process the request and generate the response using the highest supported protocol version.
		supported version	If the message contains any unrecognized **Extension** element that has an attribute of **critical** with a value of "true", return an **Error** message with the value of **errorCode** set to 4 (critical element not recognized). Otherwise, process the request and generate the response using the version of the protocol indicated in the request.
		lower than DS supports	Return an **Error** message with **Error Code** = 6.
	VEReq	higher than DS supports	Ignore unrecognized elements, including unrecognized **Extension** elements. As appropriate, either forward the request to the ACS, or process the request and generate the response using the highest supported protocol version.
		supported version	Ignore unrecognized elements, including unrecognized **Extension** elements. As appropriate, either forward the request to the ACS, or process the request and generate response using the version of the protocol indicated in the request.
		lower than DS supports	Return an **Error** message with **Error Code** = 6.
	VERes	any	Forward without regard to version number.
	Error	any	Forward without regard to version number.

errata 1

errata 1

errata 1

errata 1

Table 17: Versioning and Parsing

[5] These are the default actions for a DS to support the protocol. The Payment Scheme may specify other validations or actions in order to meet scheme-specific requirements.

[6] Note: Typically, the Directory Server will be updated to the latest version of this protocol.

Entity	Message Received	If version number is:	then:
Access Control Server	~~any~~ **VEReq**	higher than the ACS supports	Process the request and generate a response using the highest supported protocol version. Any element not defined for ~~that~~ the highest supported version must be ignored (provided it is not a "critical" extension as discussed on page 98).
		supported version	Generate response using the version of the protocol indicated in the request. Any non-critical **Extension** element not supported by the ACS ~~defined for that version~~ must be ignored ~~(provided it is not a "critical" extension)~~.
	PAReq	equal to version number returned in **VERes**	Process the request and generate a response under the specified protocol version. Any non-critical **Extension** element not supported by the ACS must be ignored.
		different version number	Send **PARes** with **iReqCode** = 55. If the **PAReq** version number is higher than supported, format the **PARes** using the highest supported version; otherwise, use the version number of the **PAReq**.
Merchant Server Plug-in	**VERes**	supported version	Generate subsequent **PAReq** message (if any) using the version of the protocol indicated in the **VERes**. Any non-critical **Extension** element not recognized by the MPI ~~defined for that version~~ must be ignored ~~(provided it is not a "critical" extension)~~.
	Error	any	If the Directory Server returns an **Error** message with the value of **ErrorCode** set to 6, the MPI must use the version found in the **Error** message for any subsequent **CRReq** or **VEReq** messages.

errata 47

errata 48

Table 17: Versioning and Parsing, continued

The id attribute: Message element

Request and response pairs are matched using the **id** attribute of the **Message** element. The **id** attribute of a request must be generated by the sender using an algorithm that is likely to generate unique **id**s; the **id** attribute of the response must be copied from the corresponding request. For example:

- the MPI assigns the **id** of the **Message** element that contains **VEReq,** and
- the **id** returned by the ACS in the **Message** element that contains **VERes** matches the **id** assigned by the MPI.

The value of the **id** attribute in the request must be no longer than 128 characters. The ACS must be able to accept and process any **Message.id** up to 128 characters in length. If the value exceeds 128 characters, the ACS must respond with an **Error** message with **Error Code** = 5. The **Message.id** of the **Error** message must contain the first 128 bytes of the received **Message.id**.

errata 29

The MPI must generate **id** values that meet the requirements of the ID data type as defined in **Extensible Markup Language (XML), W3C Recommendation**.

The id attribute: PARes element

The **id** attribute of the **PARes** element is generated by the ACS and used within the signature element to refer to the **PARes**. (See the signature example on page 105.)

The **PARes.id** value must be no longer than 128 characters. The MPI must be able to accept and process any **PARes.id** up to 128 characters in length. If the value exceeds 128 characters, the MPI must treat this as an error (as defined in "treat as an error" on page 85).

errata 29

The ACS must generate **PARes.id** values that meet the requirements of the ID data type as defined in **Extensible Markup Language (XML), W3C Recommendation**.[7] Failure to do so may result in the MPI being unable to validate the Signature of the **PARes**.

[7] The **PARes.id** value is used as a reference in the **PARes** Signature, and therefore the value of the **PARes.id** must meet the requirements for the ID data type as defined in the W3C XML Recommendation. At http://www.w3.org/TR/2000/REC-xml-20001006#sec-attribute-types, the Recommendation specifies that the value of an attribute of type ID must match the *Name* production, which is defined at http://www.w3.org/TR/2000/REC-xml-20001006#sec-common-syn.

HTTP POST

Requests are sent via a POST using HTTPS. Messages exchanged between 3-D Secure components are either XML documents or HTML pages with forms including fields of 3-D Secure elements. In particular:

For messages passed as XML documents – **VEReq**, **VERes**, **CRReq**, **CRRes**, and **Error** messages sent in response to one of those:

- If chunked transfer coding[8] is not used, the '**Content-Length:**' header must be present (and set to the length of the message body).
- The '**Content-Type:**' header must be present and contain the value[9]:

```
application/xml; charset=utf-8
```

errata 46

Note: For POSTs through the cardholder browser (merchant to ACS and ACS to merchant), the browser automatically assigns the Content-Type (typically, the value will be **application/x-www-form-urlencoded**).

Responses are formatted in a similar manner (including the Content-Length and Content-Type) and sent in the reply to the HTTP POST.

Base64 decoding

As specified in section 6.8 of the IETF RFC 2045, **Multipurpose Internet Mail Extensions (MIME) Part One**, Base64 decoding software must ignore any white space (such as carriage returns or line ends) within Base64 encoded data, and must not treat the presence of such characters as an error.

errata 33

Handling XML data

All XML messages transferred within 3-D Secure must use the default and recommended encoding of "**utf-8**" as described in Extensible Markup Language (XML) W3C Recommendation (see "References" on page 3 of this document).

Digital signatures

XML digital signatures for 3-D Secure messages must be generated using the algorithms defined on page 104.

Note: The minimum key length for the RSA key used to generate the signature is 768 bits; however, a key length of 1024 bits is recommended.

[8] **Hypertext Transfer Protocol – HTTP/1.1** discusses chunked transfer coding.

[9] As defined in IETF RFC 2045, `utf-8` and `"utf-8"` are completely equivalent.

Partial system outages

A 3-D Secure component may experience system failures that effectively render the component inoperable. For example:

- The Directory Server may not be able to access the database containing ACS URLs.
- An ACS may not be able to access the database containing the list of enrolled cardholders.
- An ACS may not be able to access its hardware security module providing digital signature processing.

When such failures are detected, the component should close its TCP/IP ports and stop accepting requests until the failure has been corrected. An **Error** response with an **Error Code** of 98 or 99 should be sent for any outstanding requests.

Message bundling

The DTD allows multiple requests or responses to be included in a single 3-D Secure message. However, this functionality has not been extensively tested with existing implementations of 3-D Secure. 3-D Secure components must embed exactly one 3-D Secure message (**CRReq**, **CRRes**, **VEReq**, etc.) in a **ThreeDSecure** element.

Error Codes and Invalid Request Codes

Future versions of this specification (including future errata) may define additional error and invalid request codes. All 3-D Secure components must accept any ~~positive integer~~ value in the **Error Code** field and the **Invalid Request Code** field.

If the list of **Error Code** values in Table 29 or the list of **Invalid Request Code** values in Table 25 does not contain an entry that matches a condition detected by a 3-D Secure component, the developer should select a reasonably close match. Requests for new values should be submitted using the 3-D Secure change control process.

Note that new values will not be defined to provide detailed information about the cause of a failure that is unrelated to the message received.

Detailed **Error Code** values are intended to give the other party information needed to find and fix a problem in its system. For example:

- If an MPI has sent an invalid value in one of the fields of a **PAReq**, it needs to know which field is at fault.
- However, if the ACS is unable to authenticate a card number for reasons that have nothing to do with the message received (for instance, because it is unable to access the database of enrolled cardholders), the specific reason does not matter to the merchant.

Message validation

errata 26

The recipient of a 3-D Secure message must validate that:

- The XML message is well-formed.
- The Root Element is "ThreeDSecure".
- There is a "Message" Element inside of the Root Element.
- There is an appropriate message in the "Message" Element.

  For example, a Directory Server expects to receive the following messages: **CRReq**, **VEReq**, **VERes**, or **Error**. Any other message is treated as an error.

- Each required field is present.
- For responses: Message ID matches that of request.

The recipient of a 3-D Secure message should perform only those validations necessary to ensure that the message can be correctly processed. This includes validations necessary to ensure that the business context of the transaction is valid and those necessary to ensure that the message meets applicable technical requirements. The tables that follow define the required and optional validations for each data element.

Field edit criteria

Note: Only the specified validations are to be performed. Do not reject a message based on any validation that is not listed in the tables that follow.

Required field

A field is required if either:

- the element is always required (as denoted by an R in the Inclusion column), or
- the element is conditional (as denoted by a C in the Inclusion column), but required in this instance because the contents of the message meet conditions specified in the table.

Unless explicitly noted otherwise in the following tables, if a field is required, then the tags must be present and the element must not be empty.

Optional field

When no data is to be sent for an optional element (including a conditional element that is not required based on the contents of the message), the element may be either absent, or present and empty. *errata 34*

For example, a normal one-time purchase does not require (and should not contain) any installment payment information in the **PAReq** message. In this case, the MPI may omit the **install** element entirely from the message. Alternatively, the MPI may include an empty element:

<**install**></**install**>

Missing field

A data field is missing either if the element tags are absent, or if the element is present and empty. *errata 35*

Unless explicitly noted otherwise in the following tables, an empty element must be treated in the same manner as if the element tags were absent.[10] For example, field **c** is missing in both of the following XML instances.[11]

```
<a><b>some data</b></a>
<a><b>some data</b><c></c></a>
```

[10] Note that an empty element is not equivalent to an element with white space as content (such as `<tag>   </tag>`).

[11] Field **c** is also missing in the following XML instance:

```
<a><b>some data</b><c/></a>
```

However, section 3.1 of the W3C XML Recommendation discourages the use of the empty tag (e.g., `<c/>`), for interoperability reasons.

CRReq

Purpose

The **CRReq** (Card Range Request) is sent from the Merchant Server Plug-in (MPI) to the Directory Server to request the list of participating card ranges in order to update the MPI’s internal cache information.

CRReq fields

Table 18 lists the defined fields for a **CRReq** message.

Implementation choices

Table 18 outlines the default validation requirements for the **CRReq** message. The Payment Scheme may specify other Directory Server validations or actions in order to meet scheme-specific requirements.

Required field missing

Unless explicitly noted otherwise in the table, if a required **CRReq** field is missing, the Directory Server will return an **Error** message with **Error Code** = 3. This applies whether the field is always required or conditionally required.

See “Missing field” on page 52 for details.

Edit criteria

If a **CRReq** field is present but its value does not conform to the edit criteria specified in the table, the Directory Server will return an **Error** message with **Error Code** = 5.

Field Name	DTD Element	Inclusion	Description	Directory Server Validation
Message Version Number	**version**	R	Version identifier; "1.0.2". **Edit Criteria** **Length:** 3 or more characters **Format:** n+.n+[.n+]* where: • "n" represents a numeric digit • "+"represents "one or more" • "*"represents "zero or more" The square bracket is not part of the format, but encloses the optional portion of the string.	~~As stated~~ Additional requirements are defined in "Versioning and parsing" on page 44.
Acquirer BIN	**Merchant. acqBIN**	R	Acquiring institution identification code. **Edit Criteria** **Length:** 1-11 characters **Format:** numeric digits Note: For Visa transactions, this is typically a 6-digit BIN assigned to the acquirer by Visa.	If value does not represent a participating acquirer, DS must send a **CRRes** with **iReqCode** = 50.

Table 18: CRReq Fields

Field Name	DTD Element	Inclusion	Description	Directory Server Validation
Merchant ID	**Merchant.merID**	R	Acquirer-defined merchant identifier. **Edit Criteria** **Length:** 1-24 characters **Format:** any characters Note: Individual Payment Schemes may impose specific format and character requirements on the contents of this field. For Visa, these requirements are defined in the acquirer Implementation Guide (see page 4) and are enforced at the time that the Merchant ID is populated into the DS.	~~Depending on the method used for merchant authentication in the merchant's region,~~ If the Payment Scheme or regional organization uses Merchant ID and Password for merchant authentication, DS ~~may~~ must validate against values previously populated in the DS by the merchant's acquirer. If so, and if the value does not represent a participating merchant of **Merchant.acqBIN**, then DS must send a **CRRes** with **iReqCode** = 51.

errata 17

Table 18: CRReq Fields, continued

Field Name	DTD Element	Inclusion	Description	Directory Server Validation
Password	**Merchant. password**	C	Merchant password provided by merchant's acquirer. **Edit Criteria** **Length:** 8 characters **Format:** alphanumeric Required if Merchant ID and Password is used as the authentication methodology; omitted otherwise. The requirements for use of this field will be specific to the Payment Scheme. The Visa requirements are indicated in the acquirer Implementation Guide.	If the Payment Scheme or regional organization uses Merchant ID and Password for merchant authentication, DS must validate against values previously populated in the DS by the merchant's acquirer. If ~~not the correct password for the acquirer and merchant,~~ the password is invalid, the DS must send a **CRRes** with **CH.enrolled** = N and: • If the element is missing when required, **iReqCode** = 52. • If the password is not valid for the combination of **Merchant.acqBIN** and **Merchant.merID**, **iReqCode** = 53.
Serial Number	**serialNumber**	O	A value returned in a previous **CRRes**. If this element is present, the Directory Server returns card ranges that have been updated since the time of the **CRRes**; if this element is absent, the Directory Server returns all participating card ranges. **Edit Criteria** **Length:** 1-20 characters **Format:** numeric digits (representing a maximum 64-bit unsigned integer)	If cannot locate (for example, if value is too old), DS must send a **CRRes** with **iReqCode** = 57.

errata 36

Table 18: CRReq Fields, continued

CRRes

Purpose

The **CRRes** (Card Range Response) is sent from the Directory Server to the Merchant Server Plug-in (MPI) in response to a **CRReq** messages. It is used to provide the list of participating card ranges in order to update the MPI's internal cache information.

CRRes fields

Table 19 lists the defined fields for a **CRRes** message.

"treat as an error"

For **CRRes**, the term "treat as an error" indicates that the MPI:

- must not store the contents of the **CRRes**, and
- may optionally send an **Error** message to the Directory Server.

errata 37

Required field missing

Unless explicitly noted otherwise in the table, if a required **CRRes** field is missing, the MPI must treat it as an error. This applies whether the field is always required or conditionally required.

If the MPI sends an **Error** message in this situation, the **Error Code** must be 3.

Edit criteria

If a **CRRes** field is present but its value does not conform to the edit criteria specified in the table, the MPI must treat it as an error.

If the MPI sends an **Error** message in this situation, the **Error Code** must be 5.

Field Name	DTD Element	Inclusion	Description	MPI Validation
Message Version Number	**version**	R	Version identifier; “1.0.2”. **Edit Criteria** **Length:** 3 or more characters **Format:** n+.n+[.n+]* where: • “n” represents a numeric digit • “+”represents “one or more” • “*”represents “zero or more” The square bracket is not part of the format, but encloses the optional portion of the string.	~~As stated~~ Additional requirements are defined in “Versioning and parsing” on page 44.
Card Range	**CR**	C	If **CRReq** does not include **Serial Number**, a **CR** element is included for each card range populated in the Directory Server.[12] If **CRReq** includes **Serial Number**: • If the value of **CRReq.serialNumber** is current, indicating that the Directory Server data has not changed since the previous **CRReq** from this merchant, no **CR** elements are returned. • Otherwise, only **CR** elements that have been added or deleted since the previous **CRReq** are returned. **CR** is absent when **IReq** is present. The card range consists of three required data elements as listed below.	

Table 19: CRRes Fields

[12] The Directory Server will also provide all card ranges if a 1.0.1 MPI submits a **CRReq** with **Serial Number** = 0.

Field Name	DTD Element	Inclusion	Description	MPI Validation
First Number in Card Range	**CR.begin**	R	Starting Account Number from Directory Server. **Edit Criteria:** **Length:** 13-19 characters **Format:** numeric digits	
Last Number in Card Range	**CR.end**	R	Ending Account Number from Directory Server. **Edit Criteria:** **Length:** same length as **First Number in Card Range** **Format:** numeric digits	
Action	**CR.action**	R	Indicates the action to take with the card range. **Edit Criteria** **Length:** 1 character **Value:** must be one of the following: A = Add the card range to the cache. D = Delete the card range from the cache. The card ranges must be processed in the order returned. Note: If the serial number was not included in the **CRReq** the action is *add* for all ranges returned.	

Table 19: CRRes Fields, continued

Field Name	DTD Element	Inclusion	Description	MPI Validation
Serial Number	**serialNumber**	C	Indicates the current state of the card range database. (The specific value is meaningful only to the Directory Server.) The MPI should retain this value for submission in a future **CRReq** to request only changes that have been made to the participating card range list since this message was generated. **Edit Criteria** **Length:** 1-20 characters **Format:** numeric digits (representing a maximum 64-bit unsigned integer) If **Invalid Request Code** is included, **Serial Number** element must be omitted.	

1.0.2

Table 2: CRRes Fields, continued

Errata as of January 20, 2004

Field Name	DTD Element	Inclusion	Description	MPI Validation
Invalid Request Data	**IReq**	C	Required if the **CRReq** is syntactically correct, but business processing cannot be performed for one of the reasons defined in Table 25 on page 95. **Invalid Request Data** consists of one required, one conditional, and one optional element, as listed below.	If present, MPI must not store the contents of the **CRRes**.
Invalid Request Code	**IReq.iReqCode**	~~C~~R	Code indicating the problem identified in the **CRReq**. Must be one of the values listed in "Invalid Request Data Values" on page 94. **Edit Criteria** **Length:** 1-3 characters	Note that additional values may be defined at any time. The MPI must accept any value.
Invalid Request Detail	**IReq.iReqDetail**	C	May ~~identify~~ provide supporting detail, such as the specific data elements that caused the **Invalid Request Code**. Table 25 on page 95 defines standard contents to be used. **Edit Criteria** **Length:** 0-2048 characters **Format:** any characters	
Vendor Code	**IReq. vendorCode**	O	Error code (or explanatory text) to be used for trouble shooting. **Edit Criteria** **Length:** ~~max~~ 0-256 characters **Format:** any characters	

errata 38

Table 19: CRRes Fields, continued

VEReq

Purpose

The **VEReq** (Verify Enrollment Request) is sent by the Merchant Server Plug-in (MPI) to the Directory Server (and by the Directory Server to the ACS) to determine whether authentication is available for a particular PAN.

VEReq fields

Table 20 lists the defined fields for a **VEReq** message.

Implementation choices

Table 20 outlines the default validation requirements for the **VEReq** message. The Payment Scheme may specify other Directory Server validations or actions in order to meet scheme-specific requirements.

Required field missing

Unless explicitly noted otherwise in the table, if a required **VEReq** field is missing, the receiving entity must return an **Error** message with **Error Code** = 3. This applies whether the field is always required or conditionally required.

See “Missing field” on page 52 for details.

Edit criteria

If a **VEReq** field is present but its value does not conform to the edit criteria specified in the table, the receiving entity must return an **Error** message with **Error Code** = 5.

Field Name	DTD Element	Inclusion	Description	Validation
Message Version Number	**version**	R	Version identifier; “1.0.2”. **Edit Criteria** **Length:** 3 or more characters **Format:** n+.n+[.n+]* where: • “n” represents a numeric digit • “+”represents “one or more” • “*”represents “zero or more” The square bracket is not part of the format, but encloses the optional portion of the string.	~~As stated~~ Additional requirements are defined in “Versioning and parsing” on page 44.
Cardholder PAN	**pan**	R	Account Number; it must be the same PAN that will be used in the authorization request. The value may be: • the account number on the card • a permanent account number that is only used online • produced by the wallet as a proxy • pulled from the merchant’s local wallet • or any other number that can be submitted for authorization. **Edit Criteria** **Length:** 13-19 characters **Format:** numeric digits	If the PAN is not part of a participating range, the DS must send a **VERes** with **CH.enrolled** = N. If the PAN is not enrolled in the Payment Scheme’s 3-D Secure program, the ACS must send a **VERes** with **CH.enrolled** = N. If the message has been misrouted (the PAN does not belong to one of the issuer’s card ranges), the ACS must send a **VERes** with **CH.enrolled** = N and **iReqCode** = 56.

errata 39

Table 20: VEReq Fields

Field Name	DTD Element	Inclusion	Description	Validation
Acquirer BIN	**Merchant. acqBIN**	R	Acquiring institution identification code. **Edit Criteria** **Length:** 1-11 characters **Format:** numeric digits Note: For Visa transactions, this is typically a 6-digit BIN assigned to the acquirer by Visa.	If value does not represent a participating acquirer, DS must send a **VERes** with **CH.enrolled** = N and **iReqCode** = 50. ACS must store value for subsequent **PAReq** processing.
Merchant ID	**Merchant.merID**	R	Acquirer-defined merchant identifier. **Edit Criteria** **Length:** 1-24 characters **Format:** any characters Note: Individual Payment Schemes may impose specific format and character requirements on the contents of this field. For Visa, these requirements are defined in the acquirer Implementation Guide (see page 4) and are enforced at the time that the Merchant ID is populated into the DS.	~~Depending on the method used for merchant authentication in the merchant's region,~~ If the Payment Scheme or regional organization uses Merchant ID and Password for merchant authentication, DS ~~may~~ must validate against values previously populated in the DS by the merchant's acquirer. If so, and if the value does not represent a participating merchant of **Merchant.acqBIN**, then DS must send a **VERes** with **CH.enrolled** = N and **iReqCode** = 51. No ACS validation of content.

errata 17

Table 20: VEReq Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Password	**Merchant.password**	C	Merchant password provided by merchant's acquirer. **Edit Criteria** **Length:** 8 characters **Format:** alphanumeric Required if Merchant ID and Password is used as the authentication methodology, and omitted otherwise. The requirements for use of this field will be specific to the Payment Scheme. The Visa requirements are indicated in the acquirer Implementation Guide.	If the Payment Scheme or regional organization uses Merchant ID and Password for merchant authentication, DS must validate against values previously populated in the DS by the merchant's acquirer. If the password is invalid, the DS must send a **VERes** with **CH.enrolled** = N and: • If the element is missing when required, **iReqCode** = 52. • If it is not a valid password for the combination of **Merchant.acqBIN** and **Merchant.merID**, **iReqCode** = 53. Note: The DS validates the Merchant password. The ACS does not. Unless specifically stated otherwise for a Payment Scheme implementation, the DS must remove the Password before forwarding the **VEReq** to the ACS. (The DS may remove the field by any method defined by the Payment Scheme, such as removing element tags entirely, removing the value leaving an empty element, or replacing the contents with spaces or other masking characters.) The ACS must not reject the transaction on the basis of this element.

errata 40

Table 20: VEReq Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Device Category	**Browser. deviceCategory**	O	Indicates the type of device or channel being used for shopping. **Edit Criteria** **Length:** 0-2 characters **Value:** must be one of the following: 0 = The client environment is such that the full size messages (**PAReq/PARes**) will be used and the core protocol specification governs. For example, PC (HTML). (Default value) 1 = The client is a constrained device, such as WAP phone, where the condensed messages (**CPRQ/CPRS**) will be used and the **Extension for Mobile Internet Devices** must be followed. 2 = The client uses two-way messaging (SMS or USSD) and the **Extension for Voice and Messaging Channels** must be followed. 3 = The client uses the voice channel and the **Extension for Voice and Messaging Channels** must be followed. ~~This element may contain any non-negative integer, and additional values may be defined at any time.~~ If this element is omitted, a value of 0 is implied.	~~ACS must not treat an unrecognized value as an error.~~ If ~~value not recognized or supported~~ the ACS does not support the device category indicated, ACS must send a **VERes** with **PAN Authentication Available** = U.

errata 12

Table 20: VEReq Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Accept Headers	**Browser.accept**	C	The exact content of the HTTP accept header as sent to the merchant from the cardholder's user agent. Required if the cardholder's user agent supplied a value. **Edit Criteria** **Length:** 0-2048 characters **Format:** any characters Note: If the total length of the accept header sent by the browser exceeds 2048 characters, the MPI must truncate the excess portion.	ACS may use the contents to determine whether authentication is available, and the appropriate ACS URL to return.
User Agent	**Browser. userAgent**	C	The exact content of the HTTP user-agent header as sent to the merchant from the cardholder's user agent. Required if the cardholder's user agent supplied a value. **Edit Criteria** **Length:** 0-256 characters **Format:** any characters Note: If the total length of the user agent header sent by the browser exceeds 256 characters, the MPI must truncate the excess portion.	ACS may use the contents to determine whether authentication is available, and the appropriate ACS URL to return.
Message Extension	**Extension**	O	Any data necessary to support the requirements that are not otherwise defined in the **VEReq** message must be carried in an instance of **Message Extension**. See page 97 for additional information about this element.	ACS must process as defined on page 97 and in applicable protocol extension.[13]

errata 41

errata 41

Table 20: VEReq Fields, continued

[13] Currently only **3-D Secure: Protocol Specification – Extension for Voice and Messaging Channels** defines extensions in the **VEReq**.

VERes

Purpose

The **VERes** (Verify Enrollment Response) is sent

- by the ACS via the Directory Server, or
- by the Directory Server

to the Merchant Server Plug-in to advise the merchant whether authentication is available for a particular PAN.

VERes fields

Table 21 lists the defined fields for a **VERes** message.

"treat as an error"

~~In the "MPI Validation" column of Table 21,~~ For **VERes**, the term "treat as an error" indicates the following behavior:

errata 21

If the Directory Server discovers the error, it must:

- return an **Error** message to the ACS, and
- create a new **VERes** with **PAN Authentication Available** = "N" and send it to the MPI.

~~that~~ If the MPI discovers the error, it must:

- end transaction processing,
- indicate the error condition to the merchant, and
- optionally send an **Error** message to the Directory Server.

Required field missing

Unless explicitly noted otherwise in the table, if a required **VERes** field is missing, the recipient must treat as an error. This applies whether the field is always required or conditionally required.

See "Missing field" on page 52 for details.

If an **Error** message is returned in this situation, it must have **Error Code** = 3

Edit criteria

If a **VERes** field is present but its value does not conform to the edit criteria specified in the table, the recipient should treat as an error.

If an **Error** message is returned in this situation, it must have **Error Code** = 5.

Field Name	DTD Element	Inclusion	Description	Validation
Message Version Number	**version**	R	Version identifier; “1.0.2”. **Edit Criteria** **Length:** 3 or more characters **Format:** n+.n+[.n+]* where: • “n” represents a numeric digit • “+”represents “one or more” • “*”represents “zero or more” The square bracket is not part of the format, but encloses the optional portion of the string.	If the MPI does not support the returned version number, must treat as an error. ~~As stated in~~ Additional requirements are defined in “Versioning and parsing” on page 44.
PAN Authentication Available	**CH.enrolled**	R	Indicates whether the **Account Identifier** can be authenticated. **Edit Criteria** **Length:** 1 character **Value:** must be one of the following: Y = Authentication Available N = Cardholder Not Participating U = Unable To Authenticate “U” ~~applies~~ is used whether the Issuer’s inability to authenticate the account is due to technical difficulties or ~~non-inclusion in program~~ business reasons.	If the value is not “Y”, MPI must not create a **PAReq**.

Table 21: VERes Fields

Field Name	DTD Element	Inclusion	Description	Validation
Account Identifier	**CH.acctID**	C	The content of this field is a data string useful to the ACS; it must not reveal the PAN and must be generated using an algorithm that is likely to generate unique values, even if the same PAN is being presented. **Edit Criteria** **Length:** 1-28 characters **Format:** any characters Required if the value of **PAN Authentication Available** is "Y"; omitted otherwise. MPI developer should be aware that the contents of this field in a 1.0.1 **VERes** may be the actual PAN.	If absent when **PAN Authentication Available** = "Y", MPI must treat as an error.
ACS URL	**url**	C	URL of Access Control Server to which **PAReq** must be sent. **Edit Criteria** **Length:** 1-2048 characters **Format:** fully qualified https URL (https://domainname…) Required if the value of **PAN Authentication Available** is "Y".	If absent or not useable when **PAN Authentication Available** = "Y", MPI must treat as an error.

1.0.2

errata 25

errata 32

Table 4: VERes Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Payment Protocols	**protocol**	C	Indicates which payment protocols are supported by the issuer system for the **Cardholder PAN** specified in **VEReq**. The only defined value is “ThreeDSecure”. If the value of **PAN Authentication Available** is “Y”, ~~at least~~ one instance of this element must be included. Otherwise, the presence of this element is optional. **Edit Criteria** **Length:** 0-12 characters **Format:** any characters ~~Possible values are:~~ ~~ThreeDSecure~~ ~~indicates that the 3-D Secure protocol is supported by the issuer system for this PAN~~	~~If does not include an instance with the value “ThreeDSecure”, MPI may assume that 3-D Secure is supported, or, if **PAN Authentication Available** = “Y”, may treat as an error.~~

errata 42

Table 21: VERes Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Invalid Request Data	**IReq**	C	Required if the **VEReq** is syntactically correct, but business processing cannot be performed for one of the reasons defined in Table 25 on page 95. Note that when **IReq** is included, the value of **PAN Authentication Available** is always "N". **Invalid Request Data** consists of one required, one conditional, and one optional element as listed below.	If present, MPI must not create a **PAReq**. ~~If present and the value of **PAN Authentication Available** is "Y", MPI must treat as an error.~~
Invalid Request Code	**IReq.iReqCode**	~~C~~R	Code indicating the problem identified in the **VEReq**. Must be one of the values listed in "Invalid Request Data Values" on page 94. **Edit Criteria** **Length:** 1-3 characters	Note that additional values may be defined at any time. The recipient must accept any value.

Table 21: VERes Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Invalid Request Detail	**IReq.iReqDetail**	C	May ~~identify~~ provide supporting detail, such as the specific data elements that caused the **Invalid Request Code**. Table 25 on page 95 defines standard contents to be used. **Edit Criteria** **Length:** 0-2048 characters **Format:** any characters	
Vendor Code	**IReq. vendorCode**	O	Error code (or explanatory text) to be used for trouble shooting. **Edit Criteria** **Length:** ~~max~~ 0-256 characters **Format:** any characters	
Message Extension	**Extension**	O	Any data necessary to support requirements that are not otherwise defined in the **VERes** message must be carried in an instance of **Message Extension**. See page 97 for additional information about this element.	MPI must ~~validate~~ process as defined on page 97 and in applicable ~~Extension~~ protocol specification.

errata 38

Table 21: VERes Fields, continued

PAReq

Purpose

The **PAReq** (Payer Authentication Request) message is sent by the Merchant Server Plug-in to the ACS through the cardholder system, providing the data required to attempt cardholder authentication.

PAReq fields

Table 22 lists the defined fields for a **PAReq** message.

Required field missing

Unless explicitly noted otherwise in the table, if a required **PAReq** field is missing, the ACS must return an **Error** message with **Error Code** = 3. This applies whether the field is always required or conditionally required.

See "Missing field" on page 52 for details.

Edit criteria

If a **PAReq** field is present but its value does not conform to the edit criteria specified in the table, the ACS must return an **Error** message with **Error Code** = 5.

Field Name	DTD element	Inclusion	Description	Validation
Message Version Number	**version**	R	Version identifier; "1.0.2". **Edit Criteria** **Length:** 3 or more characters **Format:** n+.n+[.n+]* where: • "n" represents a numeric digit • "+"represents "one or more" • "*"represents "zero or more" The square bracket is not part of the format, but encloses the optional portion of the string.	If not the version number ~~requested~~ specified in the **VERes**, ACS ~~may~~ must send **PARes** with **iReqCode** = 55~~, or may process as stated in "Versioning and parsing" on page 46~~.
Acquirer BIN	**Merchant. acqBIN**	R	From **VEReq**. **Edit Criteria** **Length:** 1-11 characters **Value:** Same value as **VEReq.Merchant.acqBIN**	If value does not match corresponding **VEReq** (as identified by **Account Identifier**), ACS must send **iReqCode** = 55.
Merchant ID	**Merchant.merID**	R	From **VEReq**. **Edit Criteria** **Length:** 1-24 characters **Value:** Same value as **VEReq.Merchant.merID**	If value does not match corresponding **VEReq** (as identified by **Account Identifier**), ACS must send **iReqCode** = 55.

Table 22: PAReq Fields

Field Name	DTD element	Inclusion	Description	Validation
Merchant Name	**Merchant.name**	R	Merchant name to be displayed on Authentication Request Page. **Edit Criteria** **Length:** 1-25 characters **Format:** any characters	ACS must include value in Authentication Request Page.
Merchant Country Code	**Merchant. country**	R	Country Code of the Merchant. The same value must be used in the authorization request. **Edit Criteria** **Length:** 3 characters **Format:** numeric digits **Value:** ISO 3166 three digit country code, other than those listed in Table 25 on page 95	If not a valid three digit ISO country code, ACS must ~~send **Error** message (with **errorCode** = 5) or~~ send a **PARes** with **iReqCode** = 54.
Merchant URL	**Merchant.url**	R	Fully qualified URL of merchant website or customer care site (http(s)://domainname…). **Edit Criteria** **Length:** 1-2048 characters **Format:** any characters	

errata 43

errata 44

errata 32
errata 31

Table 22: PAReq Fields, continued

Field Name	DTD element	Inclusion	Description	Validation
Transaction Identifier	**Purchase.xid**	R	~~Unique~~ Transaction identifier determined by merchant. Contains a 20 byte ~~statistically unique~~ value that has been Base64 encoded, giving a 28 byte result. **Edit Criteria** **Length:** 28 characters **Format:** any character	
Purchase Date & Time	**Purchase.date**	R	Date and time of purchase expressed in GMT. **Edit Criteria** **Length:** 17 characters **Format:** YYYYMMDD HH:MM:SS where: YYYY 4 numeric digits MM 2 numeric digits with value 01-12 DD 2 numeric digits with value 01-31 a single space follows the date HH 2 numeric digits with value 00-24, followed by a colon (“:”) MM 2 numeric digits with value 00-59, followed by a colon (“:”) SS 2 numeric digits with value 00-59	ACS should show value in Authentication Request Page. ~~□If used, ACS must send **Error** message (with **errorCode** = 5) if not a valid date time or if format is not valid.~~ ~~□If NOT used, ACS may send **Error** message (with **errorCode** = 5) if not a valid date time or if format is not valid.~~

Table 22: PAReq Fields, continued

Field Name	DTD element	Inclusion	Description	Validation
Display Amount	**Purchase. amount**	R	This element must be present in the message (to ensure compatibility with the existing DTD). The content of this element is not used, and it may be empty. **Edit Criteria** **Length:** 0-20 characters **Format:** any characters	ACS must not use the contents of this field in any way.
Purchase Amount	**Purchase. purchAmount**	R	Purchase amount in minor units of currency with all punctuation removed. Example: If the purchase is for USD 123.45, the **purchAmount** element will contain the value 12345. **Edit Criteria** **Length:** 1-12 characters **Format:** numeric digits	~~If number of digits less than value of **Purchase.exponent**, ACS may send **Error** message with **errorCode** = 5.~~ ACS must use content in Authentication Request Page, as described on page 83.

errata 7

errata 7

errata 2

Table 22: PAReq Fields, continued

Field Name	DTD element	Inclusion	Description	Validation
Purchase Currency	**Purchase. currency**	R	Currency in which purchase amount is expressed. **Edit Criteria** **Length:** 3 characters **Format:** numeric digits **Value:** ISO 4217 three digit currency code, other than those listed in Table 25 on page 95	If not a valid ISO currency code, ACS must ~~send **Error** message (with **errorCode** = 5) or~~ send a **PARes** with **iReqCode** = 54. ACS must use value to display **Purchase Amount**, as described on page 83.
Currency Exponent	**Purchase. exponent**	R	The minor units of currency specified in ISO 4217. For example, US Dollars has a value of 2; Japanese Yen has a value of 0. **Edit Criteria** **Length:** 1 character **Format:** numeric digit **Value:** exponent defined for currency code in ISO 4217	If not a valid exponent for **Purchase.currency** per ISO 4217, ACS must ~~send **Error** message (with **errorCode** = 5) or~~ send a **PARes** with **iReqCode** = 55. ACS must use value to display **Purchase Amount**, as described on page 83.
Order Description	**Purchase.desc**	O	Brief description of items purchased. **Edit Criteria** **Length:** 0-125 characters **Format:** any characters ~~Maximum size is 125 characters, but merchant should consider the characteristics of the cardholder's device when creating the field.~~	

errata 49

errata 49

Table 22: PAReq Fields, continued

Field Name	DTD element	Inclusion	Description	Validation
Recurring Payment Data	**Purchase.Recur**	C	Required if the merchant and cardholder have agreed to recurring payments. The recurring payment data consists of two required data elements as listed below. **Edit Criteria** Both child elements must be present. Either both child elements must be empty, or both must contain valid contents.	
Recurring Frequency	**Recur. frequency**	R	Indicates the minimum number of days between authorizations. **Edit Criteria** **Length:** 0-4 characters **Format:** numeric digits See “Recurring Frequency” on page 84 for additional information.	
Recurring Expiry	**Recur. endRecur**	R	The date after which no further authorizations should be performed. **Edit Criteria** **Length:** 0 or 8 characters **Format:** YYYYMMDD where: YYYY 4 numeric digits MM 2 numeric digits with value 01-12 DD 2 numeric digits with value 01-31 ~~This date must be in the future.~~ See “Recurring Expiry” on page 84 for additional information.	~~If date not in the future, ACS must send **iReqCode** = 55.~~ ~~If date later than card expiry date, ACS must send **iReqCode** = 55.~~

Table 22: PAReq Fields, continued

Field Name	DTD element	Inclusion	Description	Validation
Installment Payment Data	**Purchase.install**	C	Indicates the maximum number of permitted authorizations for installment payments. Required if the merchant and cardholder have agreed to installment payments. **Edit Criteria** **Length:** 0-3 characters **Format:** numeric digits **Value:** must be >1 (if not empty)	
Account Identifier	**CH.acctID**	R	From **VERes**. **Edit Criteria** **Length:** 1-28 characters **Format:** any characters	If does not match a previous available **VEReq**, ACS must send a **PARes** with **Transaction Status** = "U" and **iReqCode** = 55 or 56. Note: If ACS is unable to sign the **PARes** (for example, because ACS responds on behalf of multiple issuers and therefore cannot select the correct signing certificate), ACS must send **Error** message with **errorCode** = 5.

Table 22: PAReq Fields, continued

Field Name	DTD element	Inclusion	Description	Validation
Card Expiry Date	**CH.expiry**	R	Expiration Date supplied to merchant by cardholder (YYMM). **Edit Criteria** **Length:** 4 characters **Format:** numeric digits	~~If does not match real expiry date, ACS may send **Transaction Status** = N.~~
Message Extension	**Extension**	O	Any data necessary to support the requirements that are not otherwise defined in the **PAReq** message must be carried in an instance of **Message Extension**. See page 97 for additional information about this element.	ACS must ~~validate~~ process as defined on page 97 and in applicable ~~Extension~~ protocol specification.

Table 22: PAReq Fields, continued

Displaying purchase amount

The ACS must format the transaction amount for display to the cardholder in the Authentication Request Page. The ACS must not use the **Display Amount** element (**Purchase.amount**). *errata 7*

In order to format the transaction amount for display, the ACS must use the **Purchase Amount** element and the associated currency code and exponent elements: **Purchase.purchAmount**, **Purchase.currency**, and **Purchase.exponent**.

The decimal position is indicated by the exponent. If, for example, the value of **exponent** is “2”, this indicates that there are two minor units of currency.

The currency element contains the ISO numeric currency code. The ACS may either convert this to one of the ISO alphabetic currency code (using the published ISO 4217 tables), or may use a standard currency symbol where appropriate (such as $, €, or ¥).

For example, if the value of **purchAmount** is “12345”, **currency** is “826”, and **exponent** is “2”, the ACS could display this as “GBP 123.45” or “£123.45”.

Note that the ACS must validate **Purchase Currency** to ensure that it is a valid ISO 4217 numeric currency code, as described on page 36.

Recurring Frequency

The value of **Recur.frequency** indicates the minimum number of days between authorizations. A frequency of monthly is indicated by a value of 28. The earliest possible date for each authorization is based on the actual date of the prior authorization.

Table 23 illustrates the earliest possible dates for a subsequent authorization when the value of **Recur.frequency** is 28. Later authorizations are acceptable (until **Recur.endRecur**).

if the most recent authorization was dated:	then the earliest possible date for the next authorization is:	but the next authorization typically occurs on:
December 31, 2000	January 28, 2001	January 31, 2001
January 28, 2001	February 25, 2001	February 28, 2001
January 31, 2001	February 28, 2001	February 28, 2001

Table 23: Recurring Frequency

Recurring Expiry

It is the responsibility of the ACS software (and the cardholder software, if any) to ensure that the value of **Recur.endRecur** is not later than the card expiration date.

Note: The card needs to be valid only at the time of authorization. It is not a problem if it expires between authorization and capture.

PARes

Purpose

The **PARes** (Payer Authentication Response) message is sent by the ACS in response to the **PAReq** regardless of whether authentication is successful.

PARes fields

Table 24 lists the defined fields for a **PARes** message.

Signature validation

The MPI must alert the merchant if the signature on the **PARes** cannot be validated using the Root certificate required by the Payment Scheme. This condition should be considered the same as a failed authentication.

"treat as an error"

In the "Validation" column of Table 24, the term "treat as an error" indicates that the MPI must:

errata 23

- end transaction processing,
- indicate the error condition to the merchant, and
- optionally send an **Error** message to the ACS.

Required field missing

Unless explicitly noted otherwise in the table, if a required **PARes** field is missing, the MPI should return an **Error** message with **Error Code** = 3. This applies whether the field is always required or conditionally required.

See "Message validation" on page 51 for details.

Edit criteria

If a **PARes** field is present but its value does not conform to the edit criteria specified in the table, the MPI should return an **Error** message with **Error Code** = 5.

Field Name	DTD Element	Inclusion	Description	Validation
Message Version Number	**version**	R	Version identifier; “1.0.2”. **Edit Criteria** **Length:** 3 or more characters **Format:** n+.n+[.n+]* where: • “n” represents a numeric digit • “+”represents “one or more” • “*”represents “zero or more” The square bracket is not part of the format, but encloses the optional portion of the string.	If the returned version number does not match that from **PAReq**, MPI must treat as an error.~~As stated in “Versioning and parsing” on page 46.~~
Acquirer BIN	**Merchant. acqBIN**	R	From **PAReq**.	If any of these fields does not match **PAReq**, MPI cannot rely on the contents of the **PARes**, and must treat as an error.
Merchant ID	**Merchant.merID**	R	From **PAReq**.	
Transaction Identifier	**Purchase.xid**	R	From **PAReq**.	
Purchase Date & Time	**Purchase.date**	R	From **PAReq**.	
Purchase Amount	**Purchase. purchAmount**	R	From **PAReq**.	
Purchase Currency	**Purchase. currency**	R	From **PAReq**.	
Currency Exponent	**Purchase. exponent**	R	From **PAReq**.	

Table 24: PARes Fields

Field Name	DTD Element	Inclusion	Description	Validation
Cardholder PAN	**pan**	R	Cardholder Account Number. **Edit Criteria** **Length:** 13-19 characters **Format:** numeric digits **Value:** When **Transaction Status** is “Y” or “A”, this field must include the last four digits of the PAN supplied in the **VEReq**, preceded by zeros: • 0000000001234 (13-digit PAN) • 0000000000001234 (16-digit PAN) ~~If authentication was unsuccessful,~~ When **Transaction Status** = “N” or “U”, this field must be all zeros: one for each digit of the original PAN in the **VEReq**.	If **Transaction Status** is “Y” or “A” and the last four digits do not match the PAN supplied in the **VEReq**, MPI must treat as an error. MPI developers should be aware that this element will contain the full PAN, without any “masking”, in version 1.0.1 **PARes** messages. Note that in some regional or Payment Scheme implementations, the full PAN may be provided without overlaying any of the digits.

1.0.2

Table 24: PARes Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Signature Date & Time	**TX.time**	R	Date and Time **PARes** message was signed by ACS. Value is expressed in GMT. **Edit Criteria** **Length:** 17 characters **Format:** YYYYMMDD HH:MM:SS where: YYYY 4 numeric digits MM 2 numeric digits with value 01-12 DD 2 numeric digits with value 01-31 a single space follows the date HH 2 numeric digits with values between 00-24, followed by a colon (":") MM 2 numeric digits with values between 00-59, followed by a colon (":") SS 2 numeric digits with values between 00-59	

Table 24: PARes Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Transaction Status	**TX.status**	R	Indicates whether a transaction qualifies as an authenticated transaction. **Edit Criteria** **Length:** 1 character **Value:** must be one of the following: Y = Authentication Successful Customer was successfully authenticated. All data needed for clearing, including the **Cardholder Authentication Verification Value**, is included in the message. N = Authentication Failed Customer failed or cancelled authentication. Transaction denied. U = Authentication Could Not Be Performed Authentication could not be completed, due to technical or other problem, as indicated in **PARes.IReq**. A = Attempts Processing Performed Authentication could not be completed, but a proof of authentication attempt (CAVV) was generated.	

errata 30

1.0.2

Table 24: PARes Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Cardholder Authentication Verification Value	**TX.cavv**	C	Contains a 20 byte value that has been Base64 encoded, giving a 28 byte result. **Edit Criteria** **Length:** 28 characters **Format:** Base64-encoded data Required when the value of **Transaction Status** is “Y” or “A”. See **3-D Secure: Functional Requirements – Access Control Server** for information about producing this value.	Note: MPI must make this data available to the merchant/acquirer. The merchant and acquirer may need to include the CAVV in the authorization in order to demonstrate that authentication occurred.
Electronic Commerce Indicator	**TX.eci**	C	This Payment Scheme-specific element represents the ~~default~~ value of the ECI, as determined by the ACS. **Edit Criteria** **Length:** 0 or 2 characters **Value:** numeric digits Required for Visa and MasterCard transactions when the value of **Transaction Status** is “Y” or “A”.	

1.0.2

errata 45

1.0.2

Table 7: PARes Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
CAVV Algorithm	**TX. cavvAlgorithm**	C	Indicates the algorithm used to generate the **Cardholder Authentication Verification Value**. If the CAVV field is included, the CAVV algorithm field must also be included. If the CAVV field is missing, the CAVV algorithm field must also be missing. ~~Current defined values are:~~ Value must be one of the following: 0 = HMAC (as per SET™ TransStain) (no longer in use for version 1.0.2) 1 = CVV (no longer in use for version 1.0.2) 2 = CVV with ATN 3 = MasterCard SPA algorithm **Edit Criteria** **Length:** 0-1 character	

Table 24: PARes Fields, continued

1.0.2

errata 13

Errata as of January 20, 2004

Field Name	DTD Element	Inclusion	Description	Validation
Invalid Request Data	**IReq**	C	Required if the **PAReq** is syntactically correct, but business processing cannot be performed for one of the reasons defined in Table 25 on page 95. Note that when **IReq** is included, the value of **Transaction Status** is always “U”. **Invalid Request Data** consists of one required, one conditional, and one optional element as listed below.	~~If present and if **Transaction Status** is “Y” or “A”, MPI must treat as an error.~~
Invalid Request Code	**IReq.iReqCode**	~~C~~R	Code indicating the problem identified in the **PAReq**. Must be one of the values listed in “Invalid Request Data Values” on page 94. **Edit Criteria** **Length:** 1-3 characters	Note that additional values may be defined at any time. The MPI must accept any ~~positive integer~~ value.
Invalid Request Detail	**IReq.iReqDetail**	C	May ~~identify~~ provide supporting detail, such as the specific data elements that caused the **Invalid Request Code**. Table 25 on page 95 defines standard contents to be used. **Edit Criteria** **Length:** 0-2048 characters **Format:** any characters	The contents of this field may be used for problem resolution.

errata 30

errata 38

Table 24: PARes Fields, continued

Field Name	DTD Element	Inclusion	Description	Validation
Vendor Code	**IReq. vendorCode**	O	Error code (or explanatory text) to be used for trouble shooting. **Edit Criteria** **Length:** ~~max~~ 0-256 characters **Format:** any characters	The contents of this field may be used for problem resolution, if necessary, by the operators of the ACS.
Message Extension	**Extension**	O	Any data necessary to support requirements that are not otherwise defined in the **PARes** message must be carried in an instance of **Message Extension**. See page 97 for additional information about this element.	MPI must ~~validate~~ process as defined on page 97 and in applicable ~~Extension~~ protocol specification.

Table 24: PARes Fields, continued

Invalid Request Data Values

Table 25 on page 95 lists and describes the currently defined values for **Invalid Request Code** (**iReqCode**), and specifies the contents of **Invalid Request Detail** (**iReqDetail**) when applicable. **Vendor Code** may also be included, at the discretion of the application developer.

Note that additional **iReqCode** values may be defined at any time. All components must accept any ~~positive integer~~ value.

Errata as of January 20, 2004

Invalid Request Code	Description	**Invalid Request Detail**
50	Acquirer not participating in 3-D Secure (based on **Acquirer BIN**). Issued only by the Directory Server (DS).	
51	Merchant not participating in 3-D Secure (based on **Acquirer BIN** and **Merchant ID**). Issued only by the DS.	
52	**Password** required, but no password was supplied. Issued only by the DS.	
53	Supplied password is not valid for combination of **Acquirer BIN** and **Merchant ID**. Issued only by the DS.	
54	ISO code not valid per ISO tables (for either country or currency), or code is one of the excluded values listed in Table 26 on page 96.	Name of invalid element(s); if more than one invalid element is detected, this is a comma-delimited list. If **PAReq.Purchase.currency** and **PAReq.Purchase.exponent** form an invalid pair, list both as **iReqDetail**.
55	Transaction data not valid. For example: • **PAReq.acctid** <> **VERes.acctid** • **PAReq.version** <> **VERes.version**	Name of invalid element(s); if more than one invalid element is detected, this is a comma-delimited list.
56	If in response to a **VEReq**: **Cardholder PAN** is not in a range belonging to issuer. If in response to a **PAReq**: **PAReq** was incorrectly routed; either: • the **PAReq** was received by the wrong ACS, or • the **PAReq** should never have been sent, based on the values in the **VERes**, or • a **PAReq** with this **Account Identifier** has already been received and processed.	Name of element(s) that caused the ACS to decide that the **VEReq** or **PAReq** was incorrectly routed; if more than one invalid element is detected, this is a comma-delimited list.
57	**Serial Number** cannot be located. Issued only by the Directory Server.	
58	Issued only by the Directory Server.	"Access denied, invalid endpoint."
98	Transient system failure.	A description of the failure
99	Permanent system failure.	A description of the failure

errata 50

errata 7

errata 16

errata 8

Table 25: Invalid Request Data Values

Excluded ISO code values

errata 24

ISO Code	Value Not Permitted for 3-D Secure	Definition
ISO 4217	955	European Composite Unit
	956	European Monetary Unit
	957	European Unit of Account-9
	958	European Unit of Account-17
	959	Gold
	960	I.M.F.
	961	Silver
	962	Platinum
	963	reserved for testing
	964	Palladium
	999	no currency is involved
ISO 3166	900-999	reserved by ISO to designate country names not otherwise defined

Table 26: Excluded ISO Code Values

Errata as of January 20, 2004

Message Extensions

Message extensions

Requirements may emerge that cannot be supported by elements in the 3-D Secure messages; any data necessary to support these requirements must be carried in a message extension.

Extension data

The party defining the message extension defines the format of the data. Examples of formats that might be chosen are:

- XML data
- Binary data that is Base64 encoded

Attributes

The **Message Extension** element includes the following attributes:

Attribute Name	Inclusion	Description
id	Required	A unique identifier for the extension. See additional description below.
critical	Optional	A Boolean value indicating whether the recipient must understand the contents of the extension in order to interpret the entire message. See additional description below. Values are lowercase: `true` `false` The recipient of a message may treat this as an optional attribute. If the attribute is missing from an extension, it may be assumed to have a default value of "`false`". To ensure interoperability, the sender of the message must include this attribute even when the value is "`false`".

errata 3

Table 27: Message Extension Attributes

Identification

Each message extension defined for use in 3-D Secure must have a unique identifier assigned to it. Examples of unique identifiers are:

- Object identifiers (OID)
- Uniform Resource Identifiers (URI)

The party defining the message extension specifies the format of the identifier (OID, URI, etc.) and the value.

Criticality

The data in a message extension may affect the meaning of the rest of the data such that the entire message can only be understood in the context of the extension data. When this occurs, the extension is deemed to be *critical* and the value of the **critical** attribute must be “true”.

When an extension is critical, recipients of the message must recognize and be able to process the extension. If a 3-D Secure application other than the DS receives a message containing a critical extension that it does not recognize, it must treat the message as invalid.[14]

errata 14

When an extension is non-critical, recipients that cannot process the extension ~~can safely~~ must ignore the data.

[14] Directory Server requirements for responding to an unrecognized critical **Extension** element are described in Table 17 on page 45.

Error Message

Purpose

The **Error** message must be returned when the incoming request or response cannot be successfully processed at a protocol level (such as bad XML).

Note: Implementations may limit the number of **Error** messages that are sent to a given requester in order to mitigate the effects of denial-of-service attacks.

Error message fields

Table 28 lists the defined fields for an **Error** message.

Field Name	DTD Element	Inclusion	Description
Message Version Number	**version**	R	Version identifier; “1.0.2”. **Edit Criteria** **Length:** 3 or more characters **Format:** n+.n+[.n+]* where: • “n” represents a numeric digit • “+”represents “one or more” • “*”represents “zero or more” The square bracket is not part of the format, but encloses the optional portion of the string.
Error Code	**errorCode**	R	Code indicating the problem identified in the message. ~~See~~ Must be one of the values listed in Table 29 on page 102 ~~for the list of currently defined values~~. **Edit Criteria** **Length:** 1-2 characters Note that additional values may be defined at any time. All components must accept any ~~positive integer~~ value.
Error Description	**errorMessage**	R	Text describing the problem identified in the message. See Table 29 on page 102. **Edit Criteria** **Length:** 0-2048 characters **Format:** any characters
Error Detail	**errorDetail**	R	May identify the specific data elements that caused the Error Code. See Table 29 on page 102.
Vendor Code	**vendorCode**	O	Error code (or explanatory text) to be used for trouble shooting. **Edit Criteria** **Length:** ~~max~~ 0-256 characters **Format:** any characters

Table 28: Error Message Fields

Errata as of January 20, 2004

Client may send Error message

A client device may send an **Error** message to the server that sends it an invalid response:

- A Merchant Server Plug-in may post an **Error** message to the Directory Server.
- A Merchant Server Plug-in or the Directory Server may send an **Error** message to an ACS, by posting it to the ACS URL.

Note: Developers are strongly encouraged to send these **Error** messages.

Response to Error message

An application which receives an **Error** message as an HTTP post must respond with an HTTP response code of "200 OK" and an empty body.

An application must never send an **Error** message in response to an **Error** message.

Message id

If the 3-D Secure component is able to determine the message **id** of the message in error, it must use the same **id** in the **Error** message. If an **id** cannot be determined from the message that is in error (such as when the root element is unrecognized or the **Message** element is missing), the 3-D Secure component must generate an **id** using an algorithm that is likely to generate unique **id**s.

Related elements

Table 29 lists and describes the currently defined values for **Error Code**, and specifies the contents of **Error Detail** when applicable.

Note that additional **Error Code** values may be defined at any time. All components must accept any ~~positive integer~~ value.

Error Code	Error Description	Explanation	Error Detail
1	Root element invalid.	Root element is not recognized.	The invalid root element.
2	Message element not a defined message.	Message is not **CRReq**, **CRRes**, **VEReq**, etc., or a valid message is sent to an inappropriate component (such as **PAReq** being sent to the Directory Server)	The invalid message element.
3	Required element missing.		Name of required element that was omitted.
4	Critical element not recognized.		Name of critical element that was not recognized.
5	Format of one or more elements is invalid according to the specification.	For example, not numeric, not in defined date format, etc.	Name of invalid element(s); if more than one invalid element is detected, this is a comma-delimited list.
6	Protocol version too old.		The oldest version supported.
98	Transient system failure	For example, a queue processing requests is full.	A description of the failure.
99	Permanent system failure	For example, the disk containing a critical database cannot be accessed.	A description of the failure.

Table 29: Error Code, Error Description, and Error Detail

Appendix A: 3-D Secure XML Message Format

Organization

The following topics are included:

Document element

"ThreeDSecure" is the document element (aka root) of all XML-based documents exchanged between various services and components participating in the 3-D Secure infrastructure.

Date and time format(s)

Date and time pairs are formatted as:

YYYYMMDD HH:MM:SS

With the exception of the Card Expiry Date, dates alone are formatted as:

YYYYMMDD

Card Expiry Date is formatted as:

YYMM

XML-Signature Syntax and Processing

Description

The 3-D Secure protocol uses the **detached** signature form where the signature is external to the signed element (**PARes**) but within the same document. The signed element is referenced using a local reference (for example, '#PARes11234'). The signed element includes:

> the opening angle bracket of the **PARes** start tag
> through the closing angle bracket of the **PARes** end tag

Figure 4 on page 106 illustrates the signature structure.

Profile

The generation of 3-D Secure signatures must correspond to the requirements for element content and algorithms specified in the tables that follow.

Note: Recipients of 3-D Secure messages should only enforce these requirements to the extent necessary to validate the signature; for example, the presence of a **CanonicalizationMethod** element should not cause the validation to fail, but the absence of **Signature.KeyInfo** must cause the validation to fail.

Element	Requirements
Signature	One instance of **KeyInfo**; zero instances of **Object**
CanonicalizationMethod	Element is EMPTY with **Algorithm** attribute present
SignatureMethod	Element is EMPTY with **Algorithm** attribute present
Transforms	Not present
DigestMethod	Element is EMPTY with **Algorithm** attribute present
KeyInfo	One instance of **X509Data**
X509Data	One instance of **X509Certificate** for each certificate to be included (see "Certificate chain" on page 105)

errata 5

Table 30: XML Signature Profile

Algorithm Type	Identifier
Canonicalization	http://www.w3.org/TR/2001/REC-xml-c14n-20010315
Digest	http://www.w3.org/2000/09/xmldsig#sha1
Encoding	http://www.w3.org/2000/09/xmldsig#base64
MAC	http://www.w3.org/2000/09/xmldsig#hmac-sha1
Signature	http://www.w3.org/2000/09/xmldsig#rsa-sha1
Transform	none (but see "Canonicalization requirements" on page 105)

Table 31: XML Signature Algorithms

Certificate chain

The ACS must include the entire chain of certificates, and not just the signing certificate, in the Signature.

At this time both Visa and MasterCard ~~is using~~ use a three-level certificate hierarchy, so there must be three (3) instances of **X509Certificate**, containing: *errata 5*

- the root certificate,
- one intermediate certificate, and
- the signing certificate.

Each participating Payment Scheme will determine requirements regarding certificate key sizes. Refer to the Payment Scheme documentation for more information.

Canonicalization requirements

Note that canonicalization is a requirement of "**XML-Signature Syntax and Processing, W3C Recommendation**", also known as *xmldsig*, and listed in "References" on page 3 of this document.

Specifically, *xmldsig* states that the computation of a digest over a same-document reference must use the required canonicalization method, which is also included in the "References" on page 3 of this document.

In addition, the ACS should return the signed **PARes** message in canonical form (both the **PARes** and **SignedInfo** elements).

XML namespaces for signatures

The detached signature of the message must be declared with a default namespace of "http://www.w3.org/2000/09/xmldsig#".

Example

```
<ThreeDSecure>
  <Message id="PAReq98765">
    <PARes id="PARes12345">...</PARes>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <Reference URI="#PARes12345">
        …
        </Reference>
      </SignedInfo>
      <SignatureValue>...</SignatureValue>
      <KeyInfo>...</KeyInfo>
    </Signature>
  </Message>
</ThreeDSecure>
```

errata 4

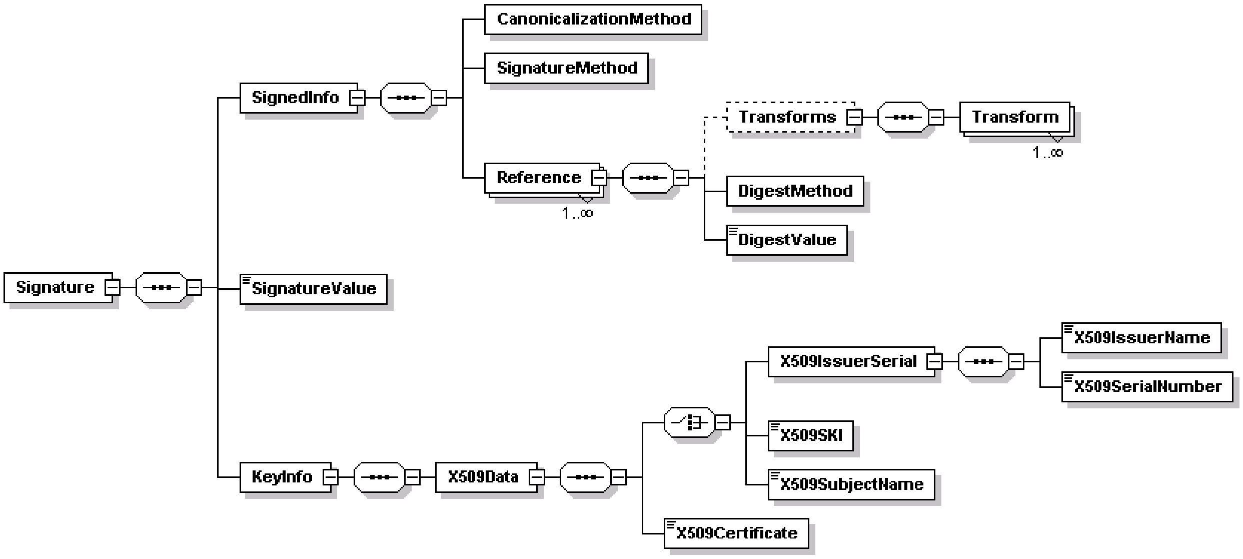


Figure 4: Signature Structure

Errata as of January 20, 2004

3-D Secure Messages

Introduction

Figure 5 provides an overview of 3-D Secure messages. The remaining figures in this section illustrate the messages individually:

Message	Page
CRReq	108
CRRes	108
VEReq	109
VERes	109
PAReq	110
PARes	111
Error Message	112

The DTD follows the diagrams.

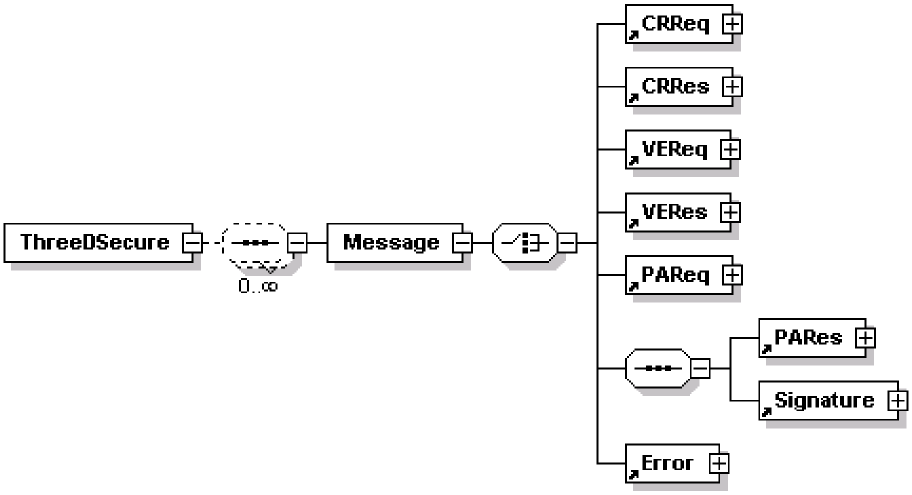


Figure 5: Overview of 3-D Secure Messages

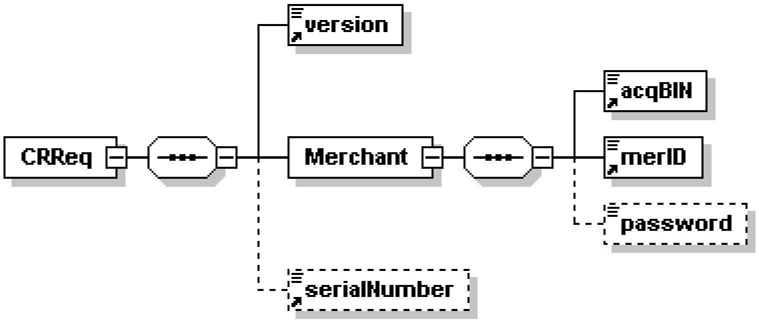


Figure 6: CRReq

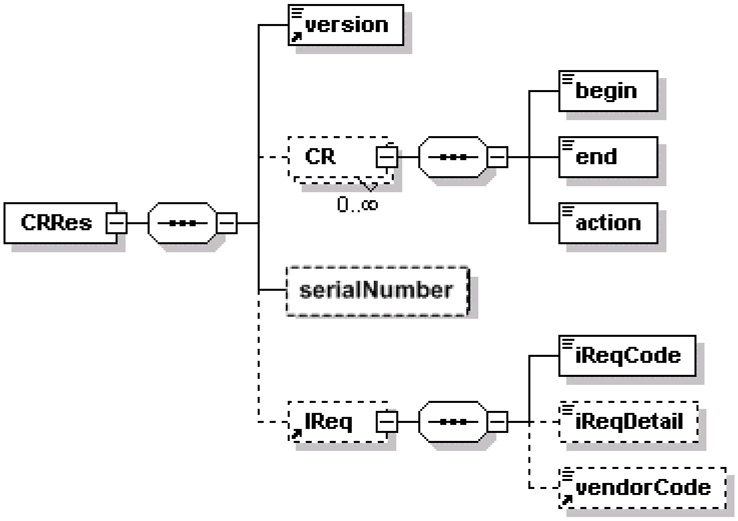


Figure 7: CRRes

Errata as of January 20, 2004

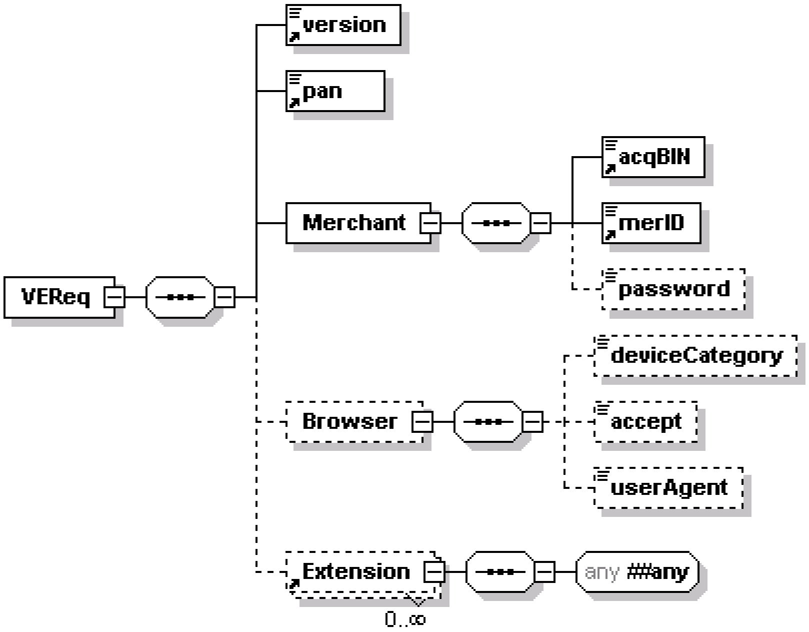


Figure 8: VEReq

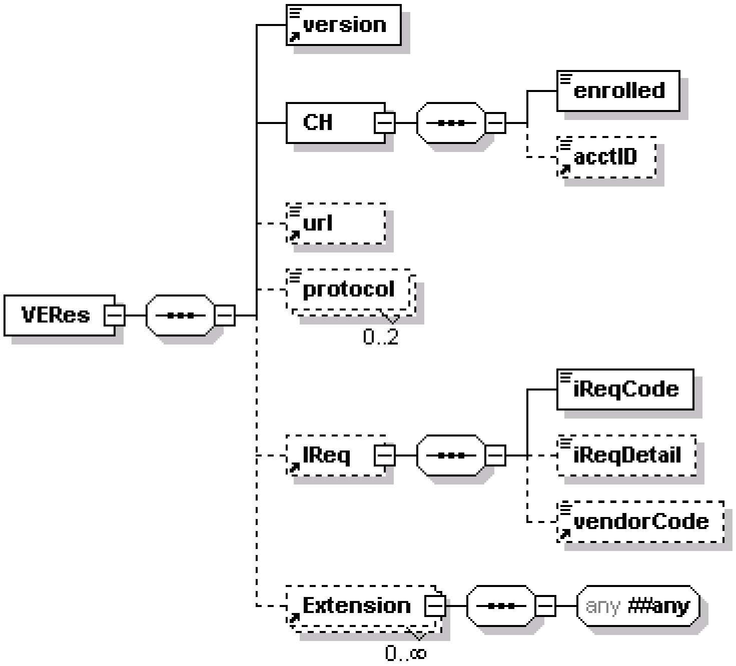


Figure 9: VERes

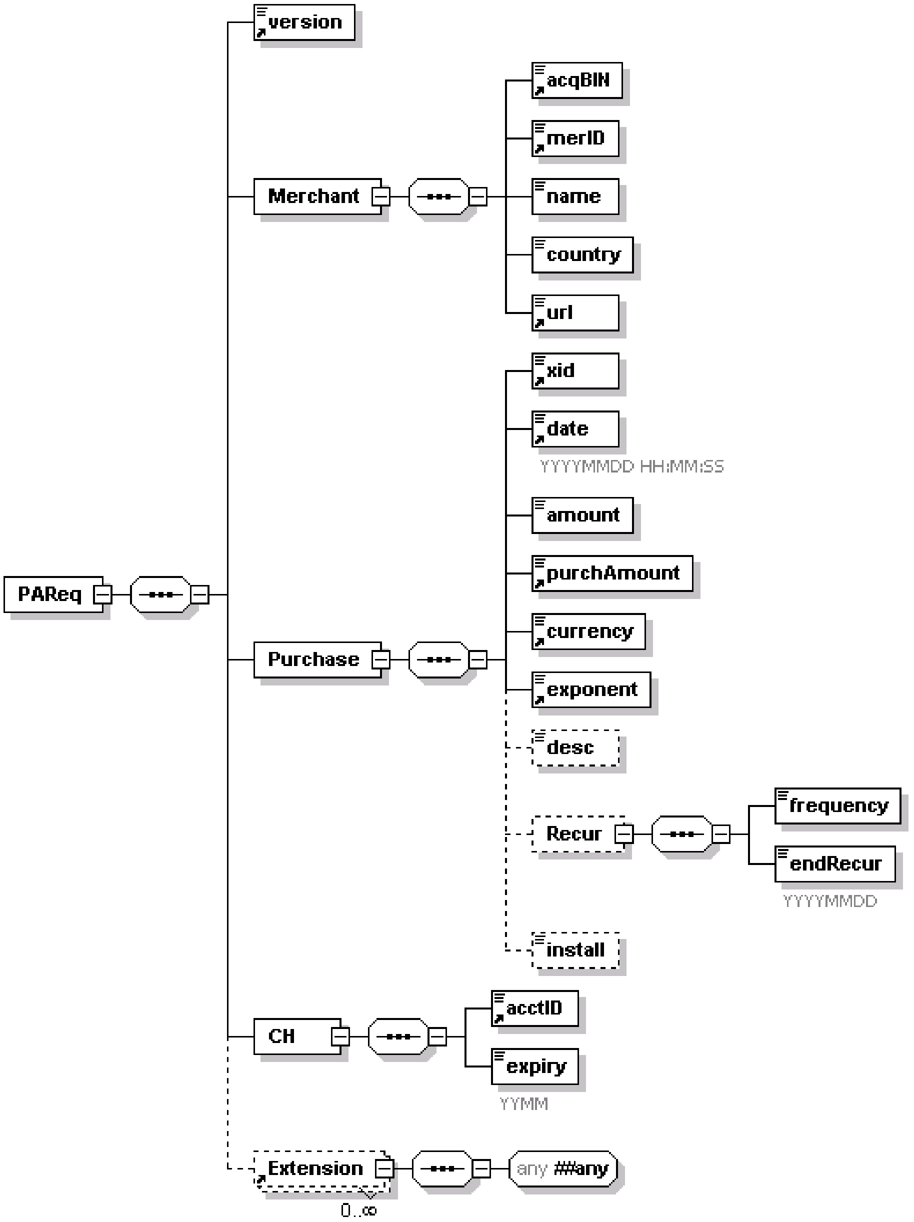


Figure 10: PAReq

Errata as of January 20, 2004

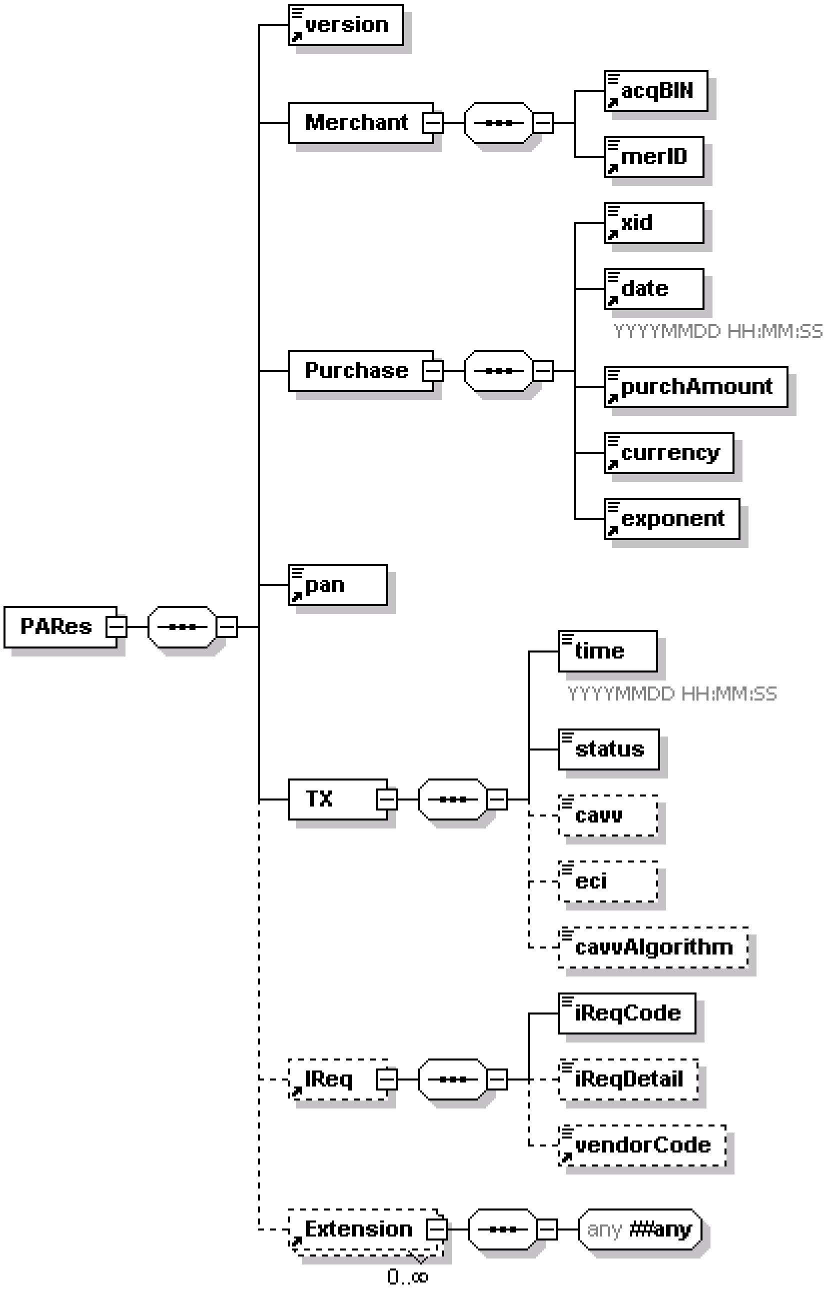


Figure 11: PARes

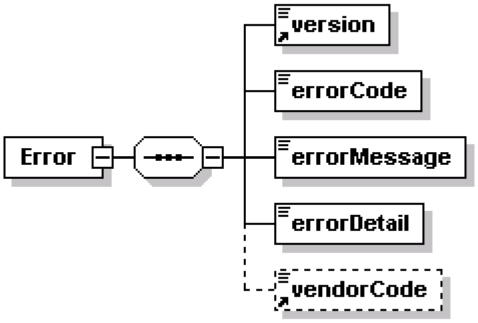


Figure 12: Error Message

3-D Secure DTD

```
<!--
**********************************************************************
   DTD for 3-D Secure Messages
   Version 1.0.2
**********************************************************************
-->
<!ELEMENT ThreeDSecure (Message)*>
<!ELEMENT Message ((CRReq | CRRes | VEReq | VERes | PAReq |
                  (PARes, Signature) | Error))>
<!ATTLIST Message id ~~CDATA ~~ID #REQUIRED >                            | errata 29

<!ELEMENT CRReq (version, Merchant, serialNumber?)>
<!ELEMENT CRRes (version, CR*, serialNumber?, IReq?)>                    1.0.2
<!ELEMENT VEReq (version, pan, Merchant, Browser?, Extension*)>
<!ELEMENT VERes (version, CH, url?, protocol*, IReq?, Extension*)>
<!ELEMENT PAReq (version, Merchant, Purchase, CH, Extension*)>
<!ELEMENT PARes (version, Merchant, Purchase, pan, TX, IReq?,
Extension*)>
<!ATTLIST PARes id ~~CDATA ~~ID #REQUIRED>                               | errata 29
<!ELEMENT Error (version, errorCode, errorMessage, errorDetail,
                vendorCode?)>

<!ELEMENT Browser  (deviceCategory?, accept?, userAgent?)>
<!ELEMENT CR       (begin, end, action)>
<!ELEMENT CH       (enrolled?, acctID?, expiry?)>
<!ELEMENT IReq     (iReqCode, iReqDetail?, vendorCode?)>
<!ELEMENT Merchant (acqBIN, merID, password?, name?, country?, url?)>
<!ELEMENT Purchase (xid, date, amount?, purchAmount, currency,
exponent,
                   desc?, Recur?, install?)>
<!ELEMENT Recur    (frequency, endRecur)>
<!ELEMENT TX       (time, status, cavv?, eci?, cavvAlgorithm?)>

<!ELEMENT Extension ANY>
<!ATTLIST Extension id CDATA #REQUIRED
                    critical (true | false) #REQUIRED >
```

```
<!ELEMENT accept         (#PCDATA)>
<!ELEMENT acctID         (#PCDATA)>
<!ELEMENT action         (#PCDATA)>
<!ELEMENT acqBIN         (#PCDATA)>
<!ELEMENT amount         (#PCDATA)>
<!ELEMENT begin          (#PCDATA)>
<!ELEMENT cavv           (#PCDATA)>
<!ELEMENT cavvAlgorithm  (#PCDATA)>
<!ELEMENT country        (#PCDATA)>
<!ELEMENT currency       (#PCDATA)>
<!ELEMENT date           (#PCDATA)>
<!ELEMENT desc           (#PCDATA)>
<!ELEMENT deviceCategory (#PCDATA)>
<!ELEMENT eci            (#PCDATA)>
<!ELEMENT end            (#PCDATA)>
<!ELEMENT endRecur       (#PCDATA)>
<!ELEMENT enrolled       (#PCDATA)>
<!ELEMENT errorCode      (#PCDATA)>
<!ELEMENT errorDetail    (#PCDATA)>
<!ELEMENT errorMessage   (#PCDATA)>
<!ELEMENT expiry         (#PCDATA)>
<!ELEMENT exponent       (#PCDATA)>
<!ELEMENT frequency      (#PCDATA)>
<!ELEMENT install        (#PCDATA)>
<!ELEMENT iReqCode       (#PCDATA)>
<!ELEMENT iReqDetail     (#PCDATA)>
<!ELEMENT merID          (#PCDATA)>
<!ELEMENT name           (#PCDATA)>
<!ELEMENT pan            (#PCDATA)>
<!ELEMENT password       (#PCDATA)>
<!ELEMENT protocol       (#PCDATA)>
<!ELEMENT purchAmount    (#PCDATA)>
<!ELEMENT serialNumber   (#PCDATA)>
<!ELEMENT status         (#PCDATA)>
<!ELEMENT time           (#PCDATA)>
<!ELEMENT url            (#PCDATA)>
<!ELEMENT userAgent      (#PCDATA)>
<!ELEMENT vendorCode     (#PCDATA)>
<!ELEMENT version        (#PCDATA)>
<!ELEMENT xid            (#PCDATA)>
```

```
<!--
**********************************************************************

  DTD for XML Signatures
  http://www.w3.org/TR/2001/CR-xmldsig-core-20010419

  3-D Secure XML-Signatures:
    * must declare XML-Signature namespace as the default namespace
      in the Signature element.
    * must use detached signatures.
    * must use X.509v3 certificates
    * must use following algorithms:
       Digest           - http://www.w3.org/2000/09/xmldsig#sha1
       Encoding         - http://www.w3.org/2000/09/xmldsig#base64
       MAC              - http://www.w3.org/2000/09/xmldsig#hmac-sha1
       Signature        - http://www.w3.org/2000/09/xmldsig#rsa-sha1
       Canonicalization - http://www.w3.org/TR/2001/REC-xml-c14n-20010315
       Transform        - none
    * xmlns must be set to XML-Signature namespace URI
**********************************************************************
-->
```

Appendix B: 3-D Secure Field Formats

This appendix combines the field definitions provided in the message descriptions that begin on page 53. Validation requirements for each field are discussed in the tables that begin on page 53, and are not duplicated in this appendix.

Field Name	DTD Element	Inclu-sion	Message	Description
Accept Headers	**Browser.accept**	C	**VEReq**	The exact content of the HTTP accept header as sent to the merchant from the cardholder's user agent. Required if the cardholder's user agent supplied a value. **Edit Criteria** **Length:** 0-2048 characters **Format:** any characters Note: If the total length of the accept header sent by the browser exceeds 2048 characters, the MPI must truncate the excess portion.
Account Identifier	**CH.acctID**	C	**VERes**	The content of this field is a data string useful to the ACS; it must not reveal the PAN and must be generated using an algorithm that is likely to generate unique values, even if the same PAN is being presented. **Edit Criteria** **Length:** 1-28 characters **Format:** any characters Required if the value of **PAN Authentication Available** is "Y"; omitted otherwise. MPI developer should be aware that the contents of this field in a 1.0.1 **VERes** may be the actual PAN.
Account Identifier	**CH.acctID**	R	**PAReq**	From **VERes**. **Edit Criteria** **Length:** 1-28 characters **Format:** any characters

Table 32: 3-D Secure Fields

Field Name	DTD Element	Inclu-sion	Message	Description
Acquirer BIN	**Merchant. acqBIN**	R	**CRReq** **VEReq**	Acquiring institution identification code. **Edit Criteria** **Length:** 1-11 characters **Format:** numeric digits Note: For Visa transactions, this is typically a 6-digit BIN assigned to the acquirer by Visa.
Acquirer BIN	**Merchant. acqBIN**	R	**PAReq**	From **VEReq**. **Edit Criteria** **Length:** 1-11 characters **Value:** Same value as **VEReq.Merchant.acqBIN**
Acquirer BIN	**Merchant. acqBIN**	R	**PARes**	From **PAReq**.
ACS URL	**url**	C	**VERes**	URL of Access Control Server to which **PAReq** must be sent. **Edit Criteria** **Length:** 1-2048 characters **Format:** fully qualified https URL (https://domainname…) Required if the value of **PAN Authentication Available** is “Y”.
Action	**CR.action**	R	**CRRes**	Indicates the action to take with the card range. **Edit Criteria** **Length:** 1 character **Value:** must be one of the following: A = Add the card range to the cache. D = Delete the card range from the cache. The card ranges must be processed in the order returned. Note: If the serial number was not included in the **CRReq** the action is *add* for all ranges returned.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Card Expiry Date	**CH.expiry**	R	**PAReq**	Expiration Date supplied to merchant by cardholder (YYMM). **Edit Criteria** **Length:** 4 characters **Format:** numeric digits
Card Range	**CR**	C	**CRRes**	If **CRReq** does not include **Serial Number**, a **CR** element is included for each card range populated in the Directory Server.[15] If **CRReq** includes **Serial Number**: • If the value of **CRReq.serialNumber** is current, indicating that the Directory Server data has not changed since the previous **CRReq** from this merchant, no **CR** elements are returned. • Otherwise, only **CR** elements that have been added or deleted since the previous **CRReq** are returned. **CR** is absent when **IReq** is present. The card range consists of three required data elements: • **First Number in Card Range – CR.begin** • **Last Number in Card Range – CR.end** • **Action – CR.action**
Cardholder Authentication Verification Value	**TX.cavv**	C	**PARes**	Contains a 20 byte value that has been Base64 encoded, giving a 28 byte result. **Edit Criteria** **Length:** 28 characters **Format:** Base64-encoded data Required when the value of **Transaction Status** is "Y" or "A". See **3-D Secure: Functional Requirements – Access Control Server** for information about producing this value.

Table 32: 3-D Secure Fields, continued

[15] The Directory Server will also provide all card ranges if a 1.0.1 MPI submits a **CRReq** with **Serial Number** = 0.

Field Name	DTD Element	Inclu-sion	Message	Description
Cardholder PAN	**pan**	R	**VEReq**	Account Number; it must be the same PAN that will be used in the authorization request. The value may be: • the account number on the card • a permanent account number that is only used online • produced by the wallet as a proxy • pulled from the merchant’s local wallet • or any other number that can be submitted for authorization. **Edit Criteria** **Length:** 13-19 characters **Format:** numeric digits
Cardholder PAN	**pan**	R	**PARes**	Cardholder Account Number. **Edit Criteria** **Length:** 13-19 characters **Format:** numeric digits **Value:** When **Transaction Status** is “Y” or “A”, this field must include the last four digits of the PAN supplied in the **VEReq**, preceded by zeros: • 0000000001234 (13-digit PAN) • 0000000000001234 (16-digit PAN) ~~If authentication was unsuccessful,~~ When **Transaction Status** = “N” or “U”, this field must be all zeros: one for each digit of the original PAN in the **VEReq**.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
CAVV Algorithm	**TX. cavvAlgorithm**	C	**PARes**	Indicates the algorithm used to generate the **Cardholder Authentication Verification Value**. If the CAVV field is included, the CAVV algorithm field must also be included. If the CAVV field is missing, the CAVV algorithm field must also be missing. ~~Current defined values are:~~ Value must be one of the following: 0 = HMAC (as per SET™ TransStain) (no longer in use for version 1.0.2) 1 = CVV (no longer in use for version 1.0.2) 2 = CVV with ATN 3 = MasterCard SPA algorithm **Edit Criteria** **Length:** 0-1 character
Currency Exponent	**Purchase. exponent**	R	**PAReq**	The minor units of currency specified in ISO 4217. For example, US Dollars has a value of 2; Japanese Yen has a value of 0. **Edit Criteria** **Length:** 1 character **Format:** numeric digit **Value:** exponent defined for currency code in ISO 4217
Currency Exponent	**Purchase. exponent**	R	**PARes**	From **PAReq**.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclusion	Message	Description
Device Category	**Browser. deviceCategory**	O	**VEReq**	Indicates the type of device or channel being used for shopping. **Edit Criteria** **Length:** 0-2 characters **Value:** must be one of the following: 0 = The client environment is such that the full size messages (**PAReq/PARes**) will be used and the core protocol specification governs. For example, PC (HTML). (Default value) 1 = The client is a constrained device, such as WAP phone, where the condensed messages (**CPRQ/CPRS**) will be used and the **Extension for Mobile Internet Devices** must be followed. 2 = The client uses two-way messaging (SMS or USSD) and the **Extension for Voice and Messaging Channels** must be followed. 3 = The client uses the voice channel and the **Extension for Voice and Messaging Channels** must be followed. ~~This element may contain any non-negative integer, and additional values may be defined at any time.~~ If this element is omitted, a value of 0 is implied.
Display Amount	**Purchase. amount**	R	**PAReq**	This element must be present in the message (to ensure compatibility with the existing DTD). The content of this element is not used, and it may be empty. **Edit Criteria** **Length:** 0-20 characters **Format:** any characters

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Electronic Commerce Indicator	**TX.eci**	C	**PARes**	This Payment Scheme-specific element represents the ~~default~~ value of the ECI, as determined by the ACS. **Edit Criteria** **Length:** 0 or 2 characters **Value:** numeric digits Required for Visa and MasterCard transactions when the value of **Transaction Status** is “Y” or “A”.
Error Code	**errorCode**	R	**Error**	Code indicating the problem identified in the message. ~~See~~ Must be one of the values listed in Table 29 on page 102 ~~for the list of currently defined values~~. **Edit Criteria** **Length:** 1-2 characters Note that additional values may be defined at any time. All components must accept any ~~positive integer~~ value.
Error Description	**errorMessage**	R	**Error**	Text describing the problem identified in the message. See Table 29 on page 102. **Edit Criteria** **Length:** 0-2048 characters **Format:** any characters
Error Detail	**errorDetail**	R	**Error**	May identify the specific data elements that caused the Error Code. See Table 29 on page 102.
First Number in Card Range	**CR.begin**	R	**CRRes**	Starting Account Number from Directory Server. **Edit Criteria:** **Length:** 13-19 characters **Format:** numeric digits

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Installment Payment Data	**Purchase.install**	C	**PAReq**	Indicates the maximum number of permitted authorizations for installment payments. Required if the merchant and cardholder have agreed to installment payments. **Edit Criteria** **Length:** 0-3 characters **Format:** numeric digits **Value:** must be >1 (if not empty)
Invalid Request Code	**IReq.iReqCode**	R	**CRRes** **VERes** **PARes**	Code indicating the problem identified in the request message. Must be one of the values listed in "Invalid Request Data Values" on page 94. **Edit Criteria** **Length:** 1-3 characters
Invalid Request Data	**IReq**	C	**CRRes**	Required if the **CRReq** is syntactically correct, but business processing cannot be performed for one of the reasons defined in Table 25 on page 95. **Invalid Request Data** consists of one required, one conditional, and one optional element: • **Invalid Request Code – IReq.iReqCode** • **Invalid Request Detail – IReq.iReqDetail** • **Vendor Code – IReq.vendorCode**

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Invalid Request Data	**IReq**	C	**VERes**	Required if the **VEReq** is syntactically correct, but business processing cannot be performed for one of the reasons defined in Table 25 on page 95. Note that when **IReq** is included, the value of **PAN Authentication Available** is always "N". **Invalid Request Data** consists of one required, one conditional, and one optional element: • **Invalid Request Code – IReq.iReqCode** • **Invalid Request Detail – IReq.iReqDetail** • **Vendor Code – IReq.vendorCode**
Invalid Request Data	**IReq**	C	**PARes**	Required if the **PAReq** is syntactically correct, but business processing cannot be performed for one of the reasons defined in Table 25 on page 95. Note that when **IReq** is included, the value of **Transaction Status** is always "U". **Invalid Request Data** consists of one required, one conditional, and one optional element: • **Invalid Request Code – IReq.iReqCode** • **Invalid Request Detail – IReq.iReqDetail** • **Vendor Code – IReq.vendorCode**
Invalid Request Detail	**IReq.iReqDetail**	C	**CRRes** **VERes** **PARes**	May ~~identify~~ provide supporting detail, such as the specific data elements that caused the **Invalid Request Code**. Table 25 on page 95 defines standard contents to be used. **Edit Criteria** **Length:** 0-2048 characters **Format:** any characters

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclusion	Message	Description
Last Number in Card Range	**CR.end**	R	**CRRes**	Ending Account Number from Directory Server. **Edit Criteria:** **Length:** same length as **First Number in Card Range** **Format:** numeric digits
Merchant Country Code	**Merchant. country**	R	**PAReq**	Country Code of the Merchant. The same value must be used in the authorization request. **Edit Criteria** **Length:** 3 characters **Format:** numeric digits **Value:** ISO 3166 three digit country code, other than those listed in Table 25 on page 95
Merchant ID	**Merchant.merID**	R	**CRReq** **VEReq**	Acquirer-defined merchant identifier. **Edit Criteria** **Length:** 1-24 characters **Format:** any characters Note: Individual Payment Schemes may impose specific format and character requirements on the contents of this field. For Visa, these requirements are defined in the acquirer Implementation Guide (see page 4) and are enforced at the time that the Merchant ID is populated into the DS.
Merchant ID	**Merchant.merID**	R	**PAReq**	From **VEReq**. **Edit Criteria** **Length:** 1-24 characters **Value:** Same value as **VEReq.Merchant.merID**
Merchant ID	**Merchant.merID**	R	**PARes**	From **PAReq**.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclusion	Message	Description
Merchant Name	**Merchant.name**	R	**PAReq**	Merchant name to be displayed on Authentication Request Page. **Edit Criteria** **Length:** 1-25 characters **Format:** any characters
Merchant URL	**Merchant.url**	R	**PAReq**	Fully qualified URL of merchant website or customer care site (http(s)://domainname…). **Edit Criteria** **Length:** 1-2048 characters **Format:** any characters
Message Extension	**Extension**	O	**VEReq** **VERes** **PAReq** **PARes**	Any data necessary to support the requirements that are not otherwise defined in the message must be carried in an instance of **Message Extension**. See page 97 for additional information about this element.
Message Version Number	**version**	R	all	Version identifier; "1.0.2". **Edit Criteria** **Length:** 3 or more characters **Format:** n+.n+[.n+]* where: • "n" represents a numeric digit • "+"represents "one or more" • "*"represents "zero or more" The square bracket is not part of the format, but encloses the optional portion of the string.
Order Description	**Purchase.desc**	O	**PAReq**	Brief description of items purchased. **Edit Criteria** **Length:** 0-125 characters **Format:** any characters ~~Maximum size is 125 characters, but merchant should consider the characteristics of the cardholder's device when creating the field.~~

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclusion	Message	Description
PAN Authentication Available	**CH.enrolled**	R	**VERes**	Indicates whether the **Account Identifier** can be authenticated. **Edit Criteria** **Length:** 1 character **Value:** must be one of the following: Y = Authentication Available N = Cardholder Not Participating U = Unable To Authenticate “U” ~~applies~~ is used whether the Issuer’s inability to authenticate the account is due to technical difficulties or ~~non-inclusion in program~~ business reasons.
Password	**Merchant. password**	C	**CRReq** **VEReq**	Merchant password provided by merchant’s acquirer. **Edit Criteria** **Length:** 8 characters **Format:** alphanumeric Required if Merchant ID and Password is used as the authentication methodology, and omitted otherwise. The requirements for use of this field will be specific to the Payment Scheme. The Visa requirements are indicated in the acquirer Implementation Guide.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Payment Protocols	**protocol**	C	**VERes**	Indicates which payment protocols are supported by the issuer system for the **Cardholder PAN** specified in **VEReq**. The only defined value is “ThreeDSecure”. If the value of **PAN Authentication Available** is “Y”, ~~at least~~ one instance of this element must be included. Otherwise, the presence of this element is optional. **Edit Criteria** **Length:** 0-12 characters **Format:** any characters ~~Possible values are:~~ ~~ThreeDSecure~~ ~~indicates that the 3-D Secure protocol is supported by the issuer system for this PAN~~
Purchase Amount	**Purchase. purchAmount**	R	**PAReq**	Purchase amount in minor units of currency with all punctuation removed. Example: If the purchase is for USD 123.45, the **purchAmount** element will contain the value 12345. **Edit Criteria** **Length:** 1-12 characters **Format:** numeric digits
Purchase Amount	**Purchase. purchAmount**	R	**PARes**	From **PAReq**.
Purchase Currency	**Purchase. currency**	R	**PAReq**	Currency in which purchase amount is expressed. **Edit Criteria** **Length:** 3 characters **Format:** numeric digits **Value:** ISO 4217 three digit currency code, other than those listed in Table 25 on page 95
Purchase Currency	**Purchase. currency**	R	**PARes**	From **PAReq**.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Purchase Date & Time	**Purchase.date**	R	**PAReq**	Date and time of purchase expressed in GMT. **Edit Criteria** **Length:** 17 characters **Format:** YYYYMMDD HH:MM:SS where: YYYY 4 numeric digits MM 2 numeric digits with value 01-12 DD 2 numeric digits with value 01-31 a single space follows the date HH 2 numeric digits with value 00-24, followed by a colon (“:”) MM 2 numeric digits with value 00-59, followed by a colon (“:”) SS 2 numeric digits with value 00-59
Purchase Date & Time	**Purchase.date**	R	**PARes**	From **PAReq**.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Recurring Expiry	**Recur. endRecur**	R	**PAReq**	The date after which no further authorizations should be performed. **Edit Criteria** **Length:** 0 or 8 characters **Format:** YYYYMMDD where: YYYY 4 numeric digits MM 2 numeric digits with value 01-12 DD 2 numeric digits with value 01-31 ~~This date must be in the future.~~ See "Recurring Expiry" on page 84 for additional information.
Recurring Frequency	**Recur. frequency**	R	**PAReq**	Indicates the minimum number of days between authorizations. **Edit Criteria** **Length:** 0-4 characters **Format:** numeric digits See "Recurring Frequency" on page 84 for additional information.
Recurring Payment Data	**Purchase.Recur**	C	**PAReq**	Required if the merchant and cardholder have agreed to recurring payments. The recurring payment data consists of two required data elements: • **Recurring Frequency – Recur.frequency** • **Recurring Expiry – Recur.endRecur** **Edit Criteria** Both child elements must be present. Either both child elements must be empty, or both must contain valid contents.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Serial Number	**serialNumber**	O	**CRReq**	A value returned in a previous **CRRes**. If this element is present, the Directory Server returns card ranges that have been updated since the time of the **CRRes**; if this element is absent, the Directory Server returns all participating card ranges. **Edit Criteria** **Length:** 1-20 characters **Format:** numeric digits (representing a maximum 64-bit unsigned integer)
Serial Number	**serialNumber**	C	**CRRes**	Indicates the current state of the card range database. (The specific value is meaningful only to the Directory Server.) The MPI should retain this value for submission in a future **CRReq** to request only changes that have been made to the participating card range list since this message was generated. **Edit Criteria** **Length:** 1-20 characters **Format:** numeric digits (representing a maximum 64-bit unsigned integer) If **Invalid Request Code** is included, **Serial Number** element must be omitted.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Signature Date & Time	**TX.time**	R	**PARes**	Date and Time **PARes** message was signed by ACS. Value is expressed in GMT. **Edit Criteria** **Length:** 17 characters **Format:** YYYYMMDD HH:MM:SS where: YYYY 4 numeric digits MM 2 numeric digits with value 01-12 DD 2 numeric digits with value 01-31 a single space follows the date HH 2 numeric digits with values between 00-24, followed by a colon (":") MM 2 numeric digits with values between 00-59, followed by a colon (":") SS 2 numeric digits with values between 00-59
Transaction Identifier	**Purchase.xid**	R	**PAReq**	~~Unique~~ Transaction identifier determined by merchant. Contains a 20 byte ~~statistically unique~~ value that has been Base64 encoded, giving a 28 byte result. **Edit Criteria** **Length:** 28 characters **Format:** any character
Transaction Identifier	**Purchase.xid**	R	**PARes**	From **PAReq**.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Transaction Status	**TX.status**	R	**PARes**	Indicates whether a transaction qualifies as an authenticated transaction. **Edit Criteria** **Length:** 1 character **Value:** must be one of the following: Y = Authentication Successful Customer was successfully authenticated. All data needed for clearing, including the **Cardholder Authentication Verification Value**, is included in the message. N = Authentication Failed Customer failed authentication. Transaction denied. U = Authentication Could Not Be Performed Authentication could not be completed, due to technical or other problem, as indicated in **PARes.IReq**. A = Attempts Processing Performed Authentication could not be completed, but a proof of authentication attempt (CAVV) was generated.
User Agent	**Browser. userAgent**	C	**VEReq**	The exact content of the HTTP user-agent header as sent to the merchant from the cardholder's user agent. Required if the cardholder's user agent supplied a value. **Edit Criteria** **Length:** 0-256 characters **Format:** any characters Note: If the total length of the user agent header sent by the browser exceeds 256 characters, the MPI must truncate the excess portion.

Table 32: 3-D Secure Fields, continued

Field Name	DTD Element	Inclu-sion	Message	Description
Vendor Code	**IReq. vendorCode**	O	**CRRes** **VERes** **PARes**	Error code (or explanatory text) to be used for trouble shooting. **Edit Criteria** **Length:** ~~max~~ 0-256 characters **Format:** any characters
Vendor Code	**vendorCode**	O	**Error**	Error code (or explanatory text) to be used for trouble shooting. **Edit Criteria** **Length:** ~~max~~ 0-256 characters **Format:** any characters

Table 32: 3-D Secure Fields, continued

Appendix C: [deleted]

Appendix D: Compression

Impact of Base64 encoding

PAReq and **PARes** are Base64 encoded prior to being inserted in the page sent to the browser. This encoding enables the messages to transit through the browser without interpretation and without change. Unfortunately, the encoding expands the message sizes by a ratio of 4 to 3. To counter this expansion, the messages are compressed prior to encoding. (Base64 encoding is defined in IETF RFC 2045. Please see “Other documents” on page 4).

Compression

The algorithm used for compression must be the DEFLATE algorithm, as specified in RFC1951. The resulting data stream must be represented in the ZLIB compressed data format, as specified by RFC1950. The compression method must be “deflate” and the compression level should be “default” or “most compressed.” However, decompressors should be prepared to accept any compression level.

No other transformation or padding is to be done on the data stream. Thus in order to send **PAReq**, the following sequence occurs:

1) The MPI builds the XML **PAReq**, in canonical format according to the DTD.

2) It passes the XML stream to an RFC1951-compliant compressor, which produces an RFC1950-compliant output stream.

3) The output stream is Base64 encoded.

4) The Base64 data is passed to the ACS through the browser as specified earlier.

5) The ACS decodes the Base64 data into an RFC1950 compliant stream.

6) The RFC1950 stream is passed to an RFC1951 compliant de-compressor, which generates the original XML.

PARes is returned using a similar mechanism.

The relevant RFCs are available at:

http://www.ietf.org/rfc/rfc1950.txt
http://www.ietf.org/rfc/rfc1951.txt

Additional information, including software implementations, may be found at:

http://www.info-zip.org
ftp://ftp.info-zip.org/pub/infozip/src/
http://www.gzip.org/zlib/

Glossary

Overview

This section includes selected terms and acronyms related to 3-D Secure.

An extensive 3-D Secure glossary is available in **3-D Secure: System Overview**, available through the “Vendors & Merchants” link on http://corporate.visa.com.

3-D Secure

An e-commerce protocol that enables the secure processing of payment card transactions in the remote environment; one of the supported protocols of the Visa Authenticated Payment Program.

3-D Secure specifications

See “References” on page 3.

absent

An element is absent if its tags do not occur in the message. For example, element **c** is absent from the following XML instance:

```
<a><b>some data</b></a>
```

Contrast *empty*. See also *missing*.

Access Control Server

A component that operates in the Issuer Domain, verifies whether authentication is available for a card number and device type, and authenticates specific transactions.

acquirer

A Member financial institution that establishes a contractual service relationship with a merchant for the purpose of accepting payment cards. In 3-D Secure, determines whether merchant is eligible to participate. Performs traditional role of receiving and forwarding authorization and settlement messages (enters transaction into interchange).

Acquirer Domain

Contains the systems and functions of the acquirer and its customers, such as merchants

ACS

See *Access Control Server*.

Attempts functionality

For Visa implementations: The process by which the proof of an authentication attempt is generated, when payment authentication is not available. Described in **3-D Secure: Functional Requirements – Access Control Server**, Visa Publication 70002-01. Effective with 3-D Secure protocol version 1.0.2.

Authenticated Payment Program

One of the programs of the Visa Secure e-Commerce Initiative

authentication

In the context of 3-D Secure, the process of verifying that the person making an e-commerce purchase is entitled to use the payment card.

Authentication History Server

A component that operates in the Interoperability Domain; archives authentication activity for use by acquirers and issuers for dispute resolution and other purposes.

authorization

A process by which an issuer, or a processor on the issuer's behalf, approves a transaction for payment.

authorization system

The systems and services through which a Payment Scheme delivers online financial processing, authorization, clearing and settlement services to Members.

See, for example, *VisaNet*.

Bank Identification Number

The first six digits of a payment card account number that uniquely identify the issuing financial institution.

Base64

Encoding applied to the **PAReq** and **PARes** messages before they are passed through the browser, and defined in RFC 2045.

BIN

See *Bank Identification Number*.

Brand Certificate Authority

See *Scheme Certificate Authority*.

browser

A client program that allows users to read hypertext documents on the World Wide Web and navigate between them. Examples are Netscape Navigator and Microsoft Internet Explorer.

In 3-D Secure, acts as a conduit to transport messages between the Merchant Server Plug-in (in the Acquirer Domain) and the Access Control Server (in the Issuer Domain).

Card Range Request

Message from the Merchant Server Plug-in to the Directory Server, requesting the list of participating card ranges in order to update the MPI's internal cache information.

Card Range Response

Message from the Directory Server to the Merchant Server Plug-in, providing the list of participating card ranges.

cardholder

Party that holds a payment card, shops, provides card number, and commits to payment.

Cardholder Authentication Verification Value

A cryptographic value generated by the ACS to provide a way during authorization processing for the authorization system to rapidly validate the integrity of certain values copied from the Payer Authentication Response to the authorization request and to prove that authentication occurred.

cardholder software

Optional cardholder software which may supplement the abilities of the browser. Chip card authentication, for example, requires cardholder software sometimes referred to as terminal software.

CAVV

See *Cardholder Authentication Verification Value*.

certificate

An electronic document that contains the public key of the certificate holder and which is attested to by a certificate authority and rendered unforgeable by cryptographic technology (signing with the private key of the certificate authority).

certificate authority

A trusted party that issues and revokes certificates.

See also *Scheme Certificate Authority*.

certificate chain

An ordered grouping of digital certificates, including the Root certificate, that are used to validate a specific certificate.

chip

An integrated circuit containing memory and logic where a copy of the VSDC application is stored and executed.

chip card

A payment card with an integrated circuit chip that stores information about the account and user.

compression

In the context of 3-D Secure, refers to the use of the DEFLATE algorithm to decrease the size of the **PAReq** or **PARes** before Base64 encoding. See Appendix D for details.

core protocol

The protocol described in this publication.

CPRQ

Condensed Payer Authentication Request, used for 3-D Secure transactions performed with mobile Internet devices. Described in **3-D Secure: Protocol Specification – Extension for Mobile Internet Devices**, Visa Publication 70006-01.

See *Payer Authentication Request*.

CPRS

Condensed Payer Authentication Response, used for 3-D Secure transactions performed with mobile Internet devices.

See *Payer Authentication Response*.

CRReq

See *Card Range Request*.

CRRes

See *Card Range Response*.

cryptography

The process of protecting information by transforming it into an unreadable format. The information is encrypted using a key, which makes the data unreadable, and is later decrypted when the information needs to be used again.

digital certificate

See *certificate*.

digital signature

An asymmetric cryptographic method whereby the recipient of the data can prove the origin and integrity of data, thereby protecting the sender of the data and the recipient against modification or forgery by third parties and the sender against forgery by the recipient. Contrast with *Message Authentication Code*.

digital wallet

A software component that allows a user to make an electronic payment with a financial instrument (such as a credit card) while hiding the low-level details of executing the payment protocol, including such tasks as entering an account number and providing shipping information and cardholder identifying information.

Directory Server

A server hardware/software entity operated in the Interoperability Domain; it maintains lists of card ranges for which authentication may be available and coordinates communication between Merchant Server Plug-ins and Access Control Servers, to determine whether authentication is available for a particular card number and device type.

empty

An element is empty if its tags occur in a message, but no content is defined. For example, element **c** is empty in the following XML instance:

```
<a><b>some data</b><c></c></a>
```

Contrast *absent*. See also *missing*.

EMV

The EMV Integrated Circuit Card Specifications for Payment Systems developed jointly by Europay, MasterCard, and Visa.

Enrollment Server

A server hardware/software entity operated in the Issuer Domain; it manages cardholder enrollment in 3-D Secure, for example by presenting a series of questions via a Web interface to be answered by the cardholder and verified by the issuer.

HTML

Hypertext Markup Language, a computer programming language used to define pages on the World Wide Web

HTTP

Hypertext Transport Protocol

HTTPS

Hypertext Transport Protocol, Secure, uses the TLS/SSL protocol to ensure the secure transmission of data over the Internet. Also called S-HTTP.

Interoperability Domain

Facilitates the transfer of information between the Issuer Domain and Acquirer Domain systems.

issuer

A Member financial institution that issues payment cards, contracts with cardholder to provide card services, determines eligibility of cardholder to participate in 3-D Secure, and identifies for the Directory Server card number ranges eligible to participate in 3-D Secure.

Issuer Domain

Contains the systems and functions of the issuer and its customers (cardholders)

key

In cryptography, the value needed to encrypt and/or decrypt something

key management

The handling of cryptographic keys and other security parameters during the entire lifetime of the keys, including generation, storage, entry and use, deletion or destruction, and archiving.

MAC

See *Message Authentication Code*.

merchant

Entity that contracts with an acquirer to accept payment cards. Manages the online shopping experience with the cardholder, obtains card number, then transfers control to the Merchant Server Plug-in, which conducts payment authentication.

Merchant Commerce Server

A server hardware/software entity that handles online transactions and facilitates communication between the merchant application and the Payment Scheme gateway.

Merchant Server Plug-in

A component that operates in the Acquirer Domain; incorporated into the merchant's Web storefront, it performs functions related to 3-D Secure on behalf of the merchant, such as determining whether authentication is available for a card number and validating the digital signature in a 3-D Secure message.

Message Authentication Code

A symmetric (secret key) cryptographic method that protects the sender and recipient against modification and forgery of data by third parties. Contrast with *digital signature*.

missing

An element is missing either if it is absent (that is, its tags do not occur in the message) or if it is present and empty. For example, element **c** is missing in both of the following XML instances:

```
<a><b>some data</b></a>              [element absent]
<a><b>some data</b><c></c></a>       [element empty]
```

MPI

See *Merchant Server Plug-in*.

PAReq

See *Payer Authentication Request*.

PARes

See *Payer Authentication Response*.

PATransReq

Payer Authentication Transaction Request; a record of authentication activity sent by the ACS to the Authentication History Server

For details, see **3-D Secure: Functional Requirements – Access Control Server**.

PATransRes

Payer Authentication Transaction Response; Authentication History Server response to **PATransReq**

For details, see **3-D Secure: Functional Requirements – Access Control Server**.

Payer Authentication Request

A message sent from the Merchant Server Plug-in to the Access Control Server via the cardholder device. Requests the issuer to authenticate its cardholder and contains the cardholder, merchant, and transaction-specific information necessary to do so.

See **PAReq** and **CPRQ**.

Payer Authentication Response	A message formatted, digitally signed, and sent from the Access Control Server to the Merchant Server Plug-in (via the cardholder device) providing the results of the issuer's 3-D Secure cardholder authentication. See **PARes** and **CPRS**.
Payment Scheme	A payment card system which defines the operating rules and conditions, and specifies the requirements for card issuance and merchant acceptance.
private key	Part of an asymmetric cryptographic system. The key that is kept secret and known only to an owner.
proof of attempt	See *Attempts functionality*.
public key	Part of an asymmetric cryptographic system. The key known to all parties.
public key pair	Two mathematically related keys – a public key and a private key – that are used with a public key (asymmetric) cryptographic algorithm to permit the secure exchange of information without the necessity for a secure exchange of a secret.
Scheme Certificate Authority	A component that operates in the Interoperability Domain on behalf of the Payment Scheme; generates and distributes selected digital certificates to entities participating in 3-D Secure.
secret key	A key used in a symmetric cryptographic algorithm such as DES which, if disclosed publicly, would compromise the security of the system.
Secure e-Commerce Initiative	A Visa initiative focused on increasing e-commerce transactions, promoting consumer confidence, and increasing Member and merchant profitability, and including the following programs: • Visa Account Information Security Program • Visa Authenticated Payment Program • Best Business Practices Program
Secure Sockets Layer	SSL: A cryptographic protocol developed by Netscape Communications Company to confidentially transmit information over open networks like the Internet. See also *Transport Layer Security*.
specifications	See *3-D Secure specifications*.
SSL	See *Secure Sockets Layer*.
Three-Domain Secure	See *3-D Secure*.
TLS	See *Transport Layer Security*.
Transport Layer Security	Successor protocol to SSL developed by the IETF (Internet Engineering Task Force)

Uniform Resource Locator	Address scheme for pages on the World Wide Web usually in the format http://address or https://address such as http://www.visa.com
URL	See *Uniform Resource Locator*.
VEReq	See *Verify Enrollment Request*.
VERes	See *Verify Enrollment Response*.
Verify Enrollment Request	Message from Merchant Server Plug-in to Directory Server or from Directory Server to ACS, asking whether authentication is available for a particular card number and device type
Verify Enrollment Response	Message from ACS or Directory Server, telling Merchant Server Plug-in whether authentication is available
VIS	Visa Integrated Circuit Card Specification
Visa Directory	See *Directory Server*.
VisaNet	The systems and services, including the V.I.P. and BASE II systems, through which Visa delivers online financial processing, authorization, clearing and settlement services to Members. VisaNet is a specific *authorization system*.
VSDC	Visa Smart Debit and Credit. The Visa service offerings for chip-based debit and credit programs which are based on EMV and VIS specifications and are support by VisaNet processing, as well as by Visa rules and regulations.
wallet	See *digital wallet*.
XML	Extensible Markup Language

Revision Log

Version	Date	Brief Description of Change	Affects
1.0	May 7, 2001	Initial issue.	Throughout.
1.0	June 12, 2001	Terms and conditions of use.	Legal agreements
1.0.1	November 1, 2001	Incorporated all errata from version 1.0. Added **Invalid Request Code** for invalid serial number. Aligned XML signature processing with W3C and IETF publications dated 20 August 2001.	**CRReq** processing by Directory Server **CRRes** processing by MPI **VEReq** processing by Directory Server **VEReq** processing by ACS **PARes** processing by ACS **PARes** processing by MPI
1.0.1	July 15, 2002	Added explicit language about message validation and sending **Error** messages in processing steps. Aligned **Account Identifier** handling in the processing steps with the conditions described in Table 21. Added new **Invalid Request Code** and **Error Code** values. Other general clarifications.	May affect all message processing if any clarifications differ from a developer's prior interpretation of the intent of the specification.
1.0.2	July 16, 2002	Adjusted to support generating proof of authentication attempt when authentication is not available.	**VEReq** and **PAReq** processing by ACS **PARes** processing by MPI
		Indicated that **Account Identifier** in **VERes** must not be the PAN.	Creation of **VERes** by ACS
		Specified that Cardholder PAN in the **PARes** must include the last four digits of the PAN supplied in the **VEReq**, preceded by zeros. Visa regions may require that the full PAN be used.	**PARes** processing by ACS **PARes** processing by MPI
		Clarified that values of codes are extensible.	Field tables
		Defined **serialNumber** as optional in **CRRes**; it is omitted when **Invalid Request Code** is included.	DTD

Note: Revision marks indicate changes made since January 16, 2003.

Version	Date	Brief Description of Change	Affects
1.0.2	January 16, 2003	Combined requirements for versions 1.0.1 and 1.0.2 into a single document for the convenience of developers.	Throughout
		Clarified treatment of messages with different version numbers. Adjusted description of **Purchase Amount**. Specified that the ACS is to rely only on the value of **Purchase Amount** and is not to make use of **Display Amount**. Specified that the critical attribute for a message extension is optional, with a default value of `"false"`. Clarified processing of critical message extensions. Clarified length of current certificate chain. Added a new **Invalid Request Code** value and a new **CAVV Algorithm** value. Clarified general message validation. Specified validation required for each field of each request or response message. Other general clarifications.	ACS handling of **PAReq** amount fields. Creation and processing of message extensions. May affect all message processing if any clarifications differ from a developer's prior interpretation of the intent of the specification.
1.0.2	January 20, 2004	Updated validation requirements: • Consolidated validation rules into a single column per message. • Enhanced field descriptions, providing consistent edit criteria and simplified validation requirements.	Chapter 6 message element tables Chapter 5 authentication processing descriptions
		Clarified message handling, including **id** attributes, Content-Length header, and Base53 decoding.	Message handling
		Other clarifications, including signature white space.	Throughout

3-D Secure: Protocol Specification
Errata
as of January 20, 2004

Number	Date Added	Location	Description
1	August 16, 2002	page 44	Adjust the sentence in “Versioning and Parsing”: Specifically, unrecognized ~~fields~~ elements in a message must be silently ignored (~~unless the field~~ except for an unrecognized **Extension** element that has a **critical** attribute with a value of “true”). *Note: See subsequent clarification in errata 51.*
		page 45	Adjust the sentence in the Directory Server/**CRReq** section of the “Versioning and Parsing” table (for version numbers higher than the DS supports, and for supported versions): If the message contains any unrecognized **Extension** element that has an attribute of **critical** with a value of “true”… Adjust the sentence in the Directory Server/**VEReq** section of the “Versioning and Parsing” table (for version numbers higher than the DS supports, and for supported versions): Ignore unrecognized elements, including unrecognized **Extension** elements.
2	August 16, 2002	page 78	Adjust the description of **Purchase Amount** in the “**PAReq** Fields” table: Up to 12-digit numeric amount in minor units of currency with all punctuation removed. … Note: The VisaNet systems do not accommodate amounts longer than 12 digits. *Note: All edit criteria were restated in the January 5, 2004, publication.*

<table>
<tr><th>Number</th><th>Date Added</th><th>Location</th><th>Description</th></tr>
<tr><td>3</td><td>August 16, 2002</td><td>page 97</td><td>Adjust the “Message Extension Attributes” table as follows:
<table>
<tr><td><b>critical</b></td><td>~~Required~~
Optional</td><td>A Boolean value indicating whether the recipient must understand the contents of the extension in order to interpret the entire message. See additional description below.
Values are lowercase: <code>true</code> <code>false</code>
The recipient of a message may treat this as an optional attribute. If the attribute is missing from an extension, it may be assumed to have a default value of <code>“false”</code>.
To ensure interoperability, the sender of the message must include this attribute even when the value is <code>“false”</code>.</td></tr>
</table></td></tr>
<tr><td>4</td><td>August 16, 2002</td><td>page 105</td><td>Adjust the ninth and twelfth lines of the example XML:
<code>&lt;/SignedInfo&gt;</code>
<code>&lt;/Signature&gt;</code></td></tr>
<tr><td>5</td><td>September 6, 2002</td><td>page 104

page 105</td><td>Adjust the <b>X509Data</b> entry in the “XML Signature Profile” table as follows:
<table>
<tr><td><b>X509Data</b></td><td>One ~~or more~~ instance~~s~~ of <b>X509Certificate</b> for each certificate to be included (see “Certificate chain” on page 105)</td></tr>
</table>
And add the referenced block:
<table>
<tr><td><b>Certificate chain</b></td><td>The ACS must include the entire chain of certificates, and not just the signing certificate, in the Signature.
At this time Visa is using a three-level certificate hierarchy, so there must be three (3) instances of <b>X509Certificate</b>, containing:
• the root certificate,
• one intermediate certificate, and
• the signing certificate.</td></tr>
</table>
<i>Note: Subsequently adjusted to indicate that both Visa and MasterCard use a three-level certificate hierarchy.</i></td></tr>
</table>

<table>
<tr><th>Number</th><th>Date Added</th><th>Location</th><th>Description</th></tr>
<tr><td>6</td><td>September 23, 2002</td><td></td><td>Obsolete. See errata 19 instead.</td></tr>
<tr><td>7</td><td>September 30, 2002</td><td>page 36</td><td>Adjust Step 7a as follows:
The ACS validates the <b>PAReq</b> data, ensuring that each of the following is true:
• …
~~□<b>Purchase Amount</b> equals <b>Display Amount</b>.~~
If any of these tests fails:
▪ …
▪ Continue with Step 8f on page 39.
Note that the ACS should not validate the contents of the <b>Display Amount</b> field, must not reject the <b>PAReq</b> based on its contents, and must not use the contents to create the cardholder display described in Step 8.</td></tr>
<tr><td></td><td></td><td>page 37</td><td>Adjust Step 8 as follows:
<table><tr><th>Data Item</th><th>From ACS</th><th>From <b>PAReq</b></th></tr><tr><td>Total amount and currency
Note: See "Displaying purchase amount" on page 83 for an explanation of how to display amount and currency.</td><td>X</td><td>~~X~~</td></tr></table></td></tr>
<tr><td></td><td></td><td>page 78</td><td>Adjust the description of <b>Display Amount</b> in the "<b>PAReq</b> Fields" table as follows:
~~Determined by merchant; must correspond to <b>Purchase Amount</b>.~~
~~Character string (up to 20 characters) that is displayable to the user.~~ This element must be present in the message (to ensure compatibility with the existing DTD). The content of this element is not used, and it may be empty.</td></tr>
<tr><td></td><td></td><td>page 78</td><td>Adjust the description of <b>Purchase Amount</b> in the "<b>PAReq</b> Fields" table as follows:
Example~~s~~: If the purchase is for USD 123.45, the <b>purchAmount</b> element will contain the value 12345.
~~<b>Display Amount</b> $123.45~~
~~<b>Purchase Amount</b> 12345~~</td></tr>
</table>

Number	Date Added	Location	Description
7, continued	September 30, 2002	page 83	Add the block referenced in Step 8: **Displaying purchase amount** The ACS must format the transaction amount for display to the cardholder in the ~~Purchase~~ Authentication Request Page. The ACS must not use the **Display Amount** element (**Purchase.amount**). In order to format the transaction amount for display, the ACS must use the **Purchase Amount** element and the associated currency code and exponent elements: **Purchase.purchAmount**, **Purchase.currency**, and **Purchase.exponent**. The decimal position is indicated by the exponent. If, for example, the value of **exponent** is "2", this indicates that there are two minor units of currency. The currency element contains the ISO numeric currency code. The ACS may either convert this to one of the ISO alphabetic currency code (using the published ISO 4217 tables), or may use a standard currency symbol where appropriate (such as $, €, or ¥). For example, if the value of **purchAmount** is "12345", **currency** is "826", and **exponent** is "2", the ACS could display this as "GBP 123.45" or "£123.45". Note that the ACS must validate **Purchase Currency** to ensure that it is a valid ISO 4217 numeric currency code, as described on page 36.
		page 95	In the "Invalid Request Data Values" table, change the Description of **Invalid Request Code** 55: Transaction data not valid. For example: ~~□purchase amount <> display amount~~ • **PAReq.acctid** <> **VERes.acctid**

<table>
<tr><th>Number</th><th>Date Added</th><th>Location</th><th>Description</th></tr>
<tr><td>8</td><td>October 23, 2002</td><td>Page 95</td><td>In the “Invalid Request Data Values” table, insert the line:
<table><tr><td>58</td><td>Issued only by the Directory Server</td><td>“Access denied, invalid endpoint.”</td></tr></table></td></tr>
<tr><td>9</td><td>January 16, 2003</td><td>page 26</td><td>Adjust Step 0b as follows:
The DS authenticates the merchant as described for **VEReq** in Step 3a. If authentication fails, the DS formats a **CRRes** with **Invalid Request Data** set to the corresponding value from Table 25 on page 95.
The DS formats a **CRRes** containing the participating ranges and sends it to the MPI. If the **CRReq** includes a value for **Serial Number**, the DS returns only those updates made since that value of **Serial Number** was current. If the DS cannot locate the **Serial Number** (for example, if the **Serial Number** is too old), the DS formats a **CRRes** with **iReqCode** set to “57”.</td></tr>
<tr><td>10</td><td>January 16, 2003</td><td>page 29</td><td>Adjust the second bullet of Step 3a as follows:
The endpoint submitting the transaction is a valid merchant endpoint. The **Merchant ID** may be used to perform this validation, by ensuring that it represents a participating merchant of the acquirer identified by **Acquirer BIN**.</td></tr>
<tr><td>11</td><td>January 16, 2003</td><td>page 42</td><td>In Step 12, delete the table “Merchant Payment System Follow-up” and adjust the text of the step as follows:
~~Depending on the results of the previous steps,~~ The Merchant Server Plug-in ~~must notify~~ notifies the merchant payment system of the ~~appropriate action to take from the following table~~ results of the authentication attempt, and provides data needed for further processing.</td></tr>
</table>

Number	Date Added	Location	Description
12	January 16, 2003	page 66	Adjust the description of **Device Category** in the “**VEReq** Fields” table: Indicates the type of ~~cardholder~~ device or channel being used for shopping. Current defined values are: 0 = The client environment is such that the full size messages (**PAReq/PARes**) will be used and the core protocol specification governs. For example, PC (HTML). 1 = The client is a constrained device, such as WAP phone, where the condensed messages (**CPRQ/CPRS**) will be used and the **Extension for Mobile Internet Devices** must be followed. 2 = The client uses two-way messaging (SMS or USSD) and the **Extension for Voice and Messaging Channels** must be followed. 3 = The client uses the voice channel and the **Extension for Voice and Messaging Channels** must be followed. This element may contain any non-negative integer, and additional values may be defined at any time. ~~All components must accept any positive integer value in this field.~~ If this element is omitted, ~~no value is provided,~~ a value of 0 is implied. *Note: This was slightly adjusted when edit criteria were restated in the January 5, 2004, publication.*
13	January 16, 2003	page 91	Add to the list of values for **CAVV Algorithm** in “**PARes** Fields” table: 3 = MasterCard SPA algorithm
14	January 16, 2003	page 98	Adjust the discussion of the criticality of message extensions: When an extension is critical, recipients of the message must recognize and be able to process the extension. If a 3-D Secure application other than the DS receives a message containing a critical extension that it does not recognize, it must treat the message as invalid.
15	January 16, 2003	page 91	Adjust the description of **CAVV Algorithm** in “**PARes** Fields” table: A ~~positive~~ non-negative integer indicating the algorithm used to generate the **Cardholder Authentication Verification Value**. *Note: This was slightly adjusted when edit criteria were restated in the January 5, 2004, publication.*

<table>
<tr><th>Number</th><th>Date Added</th><th>Location</th><th>Description</th></tr>
<tr><td>16</td><td>January 16, 2003</td><td>page 95</td><td>In the “Invalid Request Data Values” table, adjust the line:
<table>
<tr><td>56</td><td>If in response to a **VEReq**: **Cardholder PAN** is not in a range belonging to issuer.
If in response to a **PAReq**: **PAReq** was incorrectly routed; either:
• the **PAReq** was received by the wrong ACS, or
• the **PAReq** should never have been sent, based on the values in the **VERes**, or
• a **PAReq** with this **Account Identifier** has already been received and processed.</td><td>Name of element(s) that caused the ACS to decide that the **VEReq** or **PAReq** was incorrectly routed; if more than one invalid element is detected, this is a comma-delimited list.</td></tr>
</table>
</td></tr>
<tr><td>17</td><td>January 16, 2003</td><td>pages 55, 64</td><td>Adjust the description of the **Merchant ID** field in the “**CRReq** Fields” and “**VEReq** Fields” tables:
Acquirer-defined merchant identifier, up to 24 characters.~~:~~
~~□up to 15 byte alphanumeric Card Acceptor ID~~
~~□optionally followed by a hyphen and an up to 8 byte alphanumeric Card Acceptor Terminal ID~~
Note: Detailed information about the format of this field has been added to **3-D Secure: Functional Requirements – Merchant Server Plug-in**.</td></tr>
<tr><td>18</td><td>January 16, 2003</td><td>page 54</td><td>Add column clarifying expected handling by the Directory Server to “**CRReq** Fields” table.</td></tr>
<tr><td>19</td><td>January 16, 2003</td><td>pages 58-61</td><td>Add column clarifying expected handling by the MPI to “**CRRes** Fields” table.</td></tr>
<tr><td>20</td><td>January 16, 2003</td><td>pages 63-67</td><td>Add columns clarifying expected handling by the DS and ACS to the “**VEReq** Fields” table.
Note: These columns were merged into one when edit criteria were restated in the January 5, 2004, publication.</td></tr>
</table>

Number	Date Added	Location	Description
21	January 16, 2003	pages 69-73 page 68	Add column clarifying expected handling by the MPI to the “**VERes** Fields” table. Define “treat as an error” as used in the “**VERes** Fields” table: **“treat as an error”** In the “MPI Validation” column of Table 21, the term “treat as an error” indicates that the MPI must: • end transaction processing, • indicate the error condition to the merchant, and • optionally send an **Error** message to the Directory Server. *Note: “Treat as an error” was somewhat redefined when edit criteria were restated in the January 5, 2004, publication.*
22	January 16, 2003	pages 75-82	Add column clarifying expected handling by the ACS to “**PAReq** Fields” table.
23	January 16, 2003	pages 86-93 page 85	Add column clarifying expected handling by the MPI to the “**PARes** Fields” table. Define “treat as an error” as used in the “**PARes** Fields” table: **“treat as an error”** In the “MPI Validation” column of Table 24, the term “treat as an error” indicates that the MPI must: • end transaction processing, • indicate the error condition to the merchant, and • optionally send an **Error** message to the ACS.
24	January 16, 2003	page 96	Expand list of ISO code values that are not acceptable in 3-D Secure. See Table 26.
25	January 16, 2003	page 70	Adjust the description of **Account Identifier** in the “**VERes** Fields” table: The content of this field is a data string useful to the ACS; it must not reveal the PAN and must be generated using an algorithm that is likely to generate unique values, even if the same PAN is being presented.

Number	Date Added	Location	Description
26	January 16, 2003	page 51	Expand general discussion of message validation: **Message validation** ~~All 3-D Secure messages must be well formed XML and the fields of the messages must conform to the requirements described on the following pages.~~ The recipient of a 3-D Secure message must validate that: • The XML message is well-formed. • The Root Element is “ThreeDSecure”. • There is a “Message” Element inside of the Root Element. • There is an appropriate message in the “Message” Element. For example, a Directory Server expects to receive the following messages: **CRReq**, **VEReq**, **VERes**, or **Error**. Any other message is treated as an error. • Each required field is present. • For responses: Message ID matches that of request. The recipient of a 3-D Secure message ~~must validate that the message conforms to these requirements. If the message does not conform to the requirements, the recipient sends an **Error** message with an **Error Code** that corresponds to the reason for the failure.~~ should perform only those validations necessary to ensure that the message can be correctly processed. This includes validations necessary to ensure that the business context of the transaction is valid and those necessary to ensure that the message meets applicable technical requirements. The tables that follow define the required and optional validations for each data element.
27	January 16, 2003	page 90	~~Clarify that the ECI provided in the **PARes** may not be submitted in the authorization message:~~ [restated January 5, 2004:] Clarify that the ECI provided in the **PARes** is not necessarily the one that will be submitted in the authorization message: ~~Value to be passed in Authorization Message (exactly 2 decimal digits).~~ This is the default value of the ECI, as determined by ACS.

Number	Date Added	Location	Description
28	July 15, 2003	page 41	Add to Step 11: Note: Digital signature validation must fully comply with the underlying digital signature specification. In particular, the **PARes** signature must be validated over the entire contents of the **SignedInfo** element, including any inter-element white space.

29	July 15, 2003	pages 47, 113	Add to description of **id** attribute in **Message** element:
			The value of the **id** attribute in the request must be no longer than 128 characters. The ACS must be able to accept and process any **Message.id** up to 128 characters in length. If the value exceeds 128 characters, the ACS must respond with an **Error** message with **Error Code** = 5. The **Message.id** of the **Error** message must contain the first 128 bytes of the received **Message.id**.
			The MPI must generate **id** values that meet the requirements of the ID data type as defined in **Extensible Markup Language (XML), W3C Recommendation**.
			Add to description of **id** attribute in **PARes** element:
			The **PARes.id** value must be no longer than 128 characters. The MPI must be able to accept and process any **PARes.id** up to 128 characters in length. If the value exceeds 128 characters, the MPI must treat this as an error (as defined in "treat as an error" on page 85).
			The ACS must generate **PARes.id** values that meet the requirements of the ID data type as defined in **Extensible Markup Language (XML), W3C Recommendation**. Failure to do so may result in the MPI being unable to validate the Signature of the **PARes**.
			Adjust DTD:
			`<!ATTLIST Message id` ~~`CDATA`~~ `ID #REQUIRED >`
			`<!ATTLIST PARes id` ~~`CDATA`~~ `ID #REQUIRED>`

Number	Date Added	Location	Description
30	July 15, 2003	pages 36, 89, 92	Specify that when **PARes** contains **Invalid Request Data**, the value of **Transaction Status** must be set to “U”. Adjust Step 7a: If any of these tests fails: ▪ The ACS formats a Payer Authentication Response (**PARes**) message (as described in Table 24 on page 86) with **Transaction Status** set to ~~“N”~~ “U” and **Invalid Request Data** set to appropriate values as outlined in Table 25 on page 95. Adjust description of **Transaction Status**: U = Authentication Could Not Be Performed Authentication could not be completed, due to technical or other problems, as indicated in **PARes.IReq**. Include in description of **Invalid Request Data**: Note that when **IReq** is included, the value of **Transaction Status** is always “U”.
31	July 15, 2003	page 76	Clarify that although it is important that the Merchant URL be fully qualified, the ACS must not validate the Merchant URL. Add to description of **Merchant URL** in **PAReq** message: **Format:** any characters
32	December 5, 2003	pages 70, 76	Define length of URLs used in 3-D Secure messages. Add to description of **ACS URL** in **VERes** message: **Length:** 1-2048 characters Add to description of **Merchant URL** in **PAReq** message: **Length:** 1-2048 characters
33	December 5, 2003	page 48	Add a new topic, “Base64 decoding,” to the discussion of message handling: As specified in section 6.8 of the IETF RFC 2045, **Multipurpose Internet Mail Extensions (MIME) Part One**, Base64 decoding software must ignore any white space (such as carriage returns or line ends) within Base64 encoded data, and must not treat the presence of such characters as an error.

Number	Date Added	Location	Description
34	December 5, 2003	page 52	Add a new topic, "Optional field," to the discussion of message handling: When no data is to be sent for an optional element (including a conditional element that is not required based on the contents of the message), the element may be either absent, or present and empty. For example, a normal one-time purchase does not require (and should not contain) any installment payment information in the **PAReq** message. In this case, the MPI may omit the **install** element entirely from the message. Alternatively, the MPI may include an empty element: <**install**></**install**>

35	December 5, 2003	page 52	Add a new topic, "Missing field," to the discussion of message handling:
			A data field is missing either if the element tags are absent, or if the element is present and empty.
			Unless explicitly noted otherwise in the following tables, an empty element must be treated in the same manner as if the element tags were absent. For example, field **c** is missing in both of the following XML instances.
			`<a><b>some data</b></a>`
			`<a><b>some data</b><c></c></a>`

36	December 5, 2003	page 56	Correct the validation requirements for Password in **CRReq** message: …If the password is invalid, the DS must send a **CRRes** with **CH.enrolled** = N and: • If the element is missing when required, **iReqCode** = 52. • If the password is not valid for the combination of **Merchant.acqBIN** and **Merchant.merID**, **iReqCode** = 53.
37	December 5, 2003	page 57	Define "treat as an error" as used in the "**CRRes** Fields" table: **"treat as an error"** For **CRRes**, the term "treat as an error" indicates that the MPI: • must not store the contents of the **CRRes**, and • may optionally send an **Error** message to the Directory Server.

Number	Date Added	Location	Description
38	December 5, 2003	pages 61, 73, 92	Define maximum length of **Invalid Request Detail** in **CRRes**, **VERes**, and **PARes** messages: **Length:** 0-2048 characters
39	December 5, 2003	page 63	Expand the validation requirements for **Cardholder PAN** in **VEReq** message: If the PAN is not part of a participating range, the DS must send a **VERes** with **CH.enrolled** = N. If the PAN is not enrolled in the Payment Scheme's 3-D Secure program, the ACS must send a **VERes** with **CH.enrolled** = N. If the message has been misrouted (the PAN does not belong to one of the issuer's card ranges), the ACS must send a **VERes** with **CH.enrolled** = N and **iReqCode** = 56.
40	December 5, 2003	page 65	Clarify processing of Password in **VEReq**: The DS validates the Merchant password. The ACS does not. Unless specifically stated otherwise for a Payment Scheme implementation, the DS must remove the Password before forwarding the **VEReq** to the ACS. (The DS may remove the field by any method defined by the Payment Scheme, such as removing element tags entirely, removing the value leaving an empty element, or replacing the contents with spaces or other masking characters.)
41	December 5, 2003	page 67	Specify length of **Accept Headers** and **User Agent** fields in **VEReq** message. Add to description of **Accept Headers** in the **VEReq**: **Length:** 0-2048 characters **Format:** any characters Note: If the total length of the accept header sent by the browser exceeds 2048 characters, the MPI must truncate the excess portion. Add to description of **User Agent** in the **VEReq**: **Length:** 0-256 characters **Format:** any characters Note: If the total length of the user agent header sent by the browser exceeds 256 characters, the MPI must truncate the excess portion.

Number	Date Added	Location	Description
42	December 5, 2003	pages 34, 71	Remove requirement to validate value of **Payment Protocols** element. Delete Step 6b: ~~b) Examine the **Payment Protocols** and select the desired protocol to be used. If that protocol is “ThreeDSecure”, continue processing; if it is a protocol other than “ThreeDSecure”, execute the appropriate processing for the selected protocol.~~ ~~Note: The protocol selection is made based on the capabilities of the merchant systems as well as region, acquirer, and merchant policies.~~ Adjust **PAReq.protocol** description: Indicates which payment protocols are supported by the issuer system for the **Cardholder PAN** specified in **VEReq**. The only defined value is “ThreeDSecure”. If the value of **PAN Authentication Available** is “Y”, ~~at least~~ one instance of this element must be included. Otherwise, the presence of this element is optional. **Edit Criteria** **Length:** 0-12 characters **Format:** any characters ~~Possible values are:~~ ~~ThreeDSecure~~ ~~indicates that the 3-D Secure protocol is supported by the issuer system for this PAN~~
43	December 5, 2003	page 76	Specify edit criteria for **Merchant Name** in **PAReq**: **Format:** any characters
44	December 5, 2003	page 76	Correct validation requirements for **Merchant Country Code** in **PAReq**: If not a valid three digit ISO country code, ACS must ~~send **Error** message (with **errorCode** = 5) or~~ send a **PARes** with **iReqCode** = 54.
45	December 5, 2003	page 90	Define edit criteria for ECI in **PARes**: **Edit Criteria** **Length:** 0 or 2 characters **Value:** numeric digits

<table>
<tr><th>Number</th><th>Date Added</th><th>Location</th><th>Description</th></tr>
<tr><td>46</td><td>December 5, 2003</td><td>page 48</td><td>Specify an alternative to the ‘<b>Content-Length:</b>’ header for messages passed as XML document.
In discussion of HTTP POST, adjust first bullet:
• If chunked transfer coding is not used, the ‘<b>Content-Length:</b>’ header must be present (and set to the length of the message body).</td></tr>
<tr><td>47</td><td>December 5, 2003</td><td>page 46</td><td>Clarify versioning requirements for Access Control Servers:
<table>
<tr><th>Message Received</th><th>If version number is:</th><th>then:</th></tr>
<tr><td rowspan="2">~~any~~
<b>VEReq</b></td><td>higher than the ACS supports</td><td>Process the request and generate a response using the highest supported protocol version. Any element not defined for ~~that~~ the highest supported version must be ignored (provided it is not a “critical” extension as discussed on page 98).</td></tr>
<tr><td>supported version</td><td>Generate response using the version of the protocol indicated in the request. Any non-critical <b>Extension</b> element not supported by the ACS ~~defined for that version~~ must be ignored ~~(provided it is not a “critical” extension)~~.</td></tr>
<tr><td rowspan="2"><b>PAReq</b></td><td>equal to version number returned in <b>VERes</b></td><td>Process the request and generate a response under the specified protocol version. Any non-critical <b>Extension</b> element not supported by the ACS must be ignored.</td></tr>
<tr><td>different version number</td><td>Send <b>PARes</b> with <b>iReqCode</b> = 55.
If the <b>PAReq</b> version number is higher than supported, format the <b>PARes</b> using the highest supported version; otherwise, use the version number of the <b>PAReq</b>.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Number</th><th>Date Added</th><th>Location</th><th>Description</th></tr>
<tr><td>48</td><td>December 5, 2003</td><td>page 46</td><td>Clarify versioning requirements for Merchant Server Plug-in:
<table>
<tr><th>Message Received</th><th>If version number is:</th><th>then:</th></tr>
<tr><td><b>VERes</b></td><td>supported version</td><td>Generate subsequent <b>PAReq</b> message (if any) using the version of the protocol indicated in the <b>VERes</b>. Any non-critical <b>Extension</b> element not recognized by the MPI <s>defined for that version</s> must be ignored<s>(provided it is not a “critical” extension)</s>.</td></tr>
</table></td></tr>
<tr><td>49</td><td>December 5, 2003</td><td>page 79</td><td>Correct validation requirements for <b>Purchase Currency</b> and <b>Currency Exponent</b> in <b>PAReq</b>:
If not a valid ISO currency code, ACS must <s>send <b>Error</b> message (with <b>errorCode</b> = 5) or</s> send a <b>PARes</b> with <b>iReqCode</b> = 54.
If not a valid exponent for <b>Purchase.currency</b> per ISO 4217, ACS must <s>send <b>Error</b> message (with <b>errorCode</b> = 5) or</s> send a <b>PARes</b> with <b>iReqCode</b> = 55.</td></tr>
<tr><td>50</td><td>December 5, 2003</td><td>page 95</td><td>Clarify that the following <b>Invalid Request Code</b> values are issued only by the Directory Server:
50, 51, 52, 53, 57</td></tr>
<tr><td>51</td><td>December 5, 2003</td><td>page 44</td><td>Clarify requirements for dealing with unrecognized elements:
In order to support future versions of the protocol, implementations must use (or configure) XML parsers that do not validate strictly. <s>Specifically, unrecognized elements in a message must be silently ignored (except for an unrecognized <b>Extension</b> element that has a <b>critical</b> attribute with a value of “true”).</s> In particular:
• The ACS must silently ignore unrecognized elements in the <b>VEReq</b> message.
• All entities must silently ignore unrecognized non-critical <b>Extension</b> elements (that is, any <b>Extension</b> element that does not have a <b>critical</b> attribute with a value of “true”).
• All entities must silently ignore unrecognized child elements of any non-critical <b>Extension</b> element.
Note: This allows new elements to be defined in future versions of the protocol, while preserving downward compatibility, to facilitate migration to new protocol versions.</td></tr>
</table>