External Interface Guide

to
SET Secure Electronic Transaction

September 24, 1997

Preface

SET Secure Electronic Transaction

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

External Interface Guide

Several issues outside the scope of the protocol exist in developing interoperable SET implementations. This document addresses those issues and provides a way for all SET-enabled applications to interoperate at more than just the SET message level.

This document is not a part of the SET specification, but is provided to assist vendors in developing interoperable applications. A given SET implementation may choose to ignore this document altogether or to implement only a subset of the protocols described, but once a vendor has decided to use this document for a given protocol, the processing steps for that protocol are requirements.

Audience

This document is intended for readers who are familiar with the SET Specification, in particular Book 2: Programmer's Guide and Book 3: Formal Protocol Definition.

Continued on next page

Preface, continued

Related documentation

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

ANSI X3.4: 1986, Coded Character Sets – 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII).

ISO 639: 1988, *Code for the Representation of Names of Languages.*

ISO 3166: 1993, *Codes for the Representation of Names of Countries*.

ISO 10646-1: 1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.*

RFC 821: *Simple Mail Transfer Protocol*, J. Postel.

RFC 822: *Standard for the format of ARPA Internet text messages,* D. Crocker, 08/13/1982.

RFC 1521: *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, N. Borenstein, N. Freed.

RFC 1738: *Uniform Resource Locators (URL)*, T. Berners-Lee, L. Masinter, M. McCahill, 12/20/1994.

RFC 1766: *Tags for the Identification of Languages,* H. Alvestrand.

RFC 1945: *Hypertext Transfer Protocol – HTTP/1.0,* T. Berners-Lee, R. Fielding, H. Frystyk, 05/1996.

RFC 2045: *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, N. Freed, N. Borenstein, 12/02/1996.

RFC 2046: *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, N. Freed, N. Borenstein, 12/02/1996.

RFC 2047: *MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text*, K. Moore, 12/02/1996.

RFC 2048: *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*, N. Freed, J. Klensin, J. Postel, 01/28/1997.

RFC 2049: *Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples*, N. Freed, N. Borenstein, 12/02/1996.

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

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

Table of Contents

Table of Figures

Table of Tables

Part I
Introduction

Overview

SET Secure Electronic Transaction

SET Secure Electronic Transaction is a payment protocol. Its scope is therefore explicitly restricted to the core payment transaction (payment request or payment-related registration). However, SET has been designed with the larger electronic commerce picture in mind. Eventually, there will be consumer-to-merchant protocols that support shopping, negotiation, and payment selection. Rather than try to provide these functions internally, SET is designed to interoperate with other protocols that provide this functionality.

Document Purpose

Pending the availability of generalized e-commerce protocols, this document specifies the external interfaces that SET-enabled applications must implement in order to interoperate. A given SET implementation may choose to ignore this document altogether or to implement only a subset of the protocols described, but once a vendor has decided to use this document for a given protocol, the processing steps for that protocol are requirements.

Living document

This document is a “living” document. As more SET-enabled software is developed and interoperability testing is performed, this document will be revised to accommodate any required changes.

Assumption

It is assumed that the reader is familiar with the SET Specification, in particular Book 2: Programmer’s Guide and Book 3: Formal Protocol Definition. Terms defined in those books are not generally defined in this document.

Continued on next page

Overview, continued

Organization

This document includes the following parts:

Part	Purpose	Page
Part I: Introduction	Discusses the purpose of this document, notation and terminology used, and issues related to SET initiation.	1
Part II: SET Initiation Messages	Discusses the general structure of SET initiation messages, then the specific messages, including the initiation response message.	7
Part III: Transport Mechanisms	Describes the methods to be used when transporting SET messages via HTTP, SMTP, and TCP.	52
Appendix A: An Overview of MIME	MIME (Multipurpose Internet Mail Extensions) is used for the SET initiation messages described in Part II and in the transport mechanisms described in Part III. For the reader's convenience, Appendix A provides a brief overview of MIME. If you are not familiar with MIME, please read Appendix A before you read the rest of this document.	98
Appendix B: World Wide Web Operation	The mechanism described in this document supports initiation of SET transactions over the World Wide Web using HTTP. Appendix B discusses issues related to Web operation and provides a brief overview of user interaction on the Web.	104

Notation and Terminology

Notation

This document includes the following notation.

↵	CRLF (US-ASCII 0x0D 0x0A)

Table 1: Notation

Terms

This document uses the following terms.

CA	Certificate Authority
HTTP	HyperText Transfer Protocol, defined in RFC 1945
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
IPSEC	IP Security protocol
MIME	Multipurpose Internet Mail Extensions, defined in RFCs 2045-2049
OD	Order Description, the description of what the Cardholder is ordering (as described in SET Book 2: Programmer's Guide)
SSL	Secure Sockets Layer
SMTP	Simple Mail Transfer Protocol, defined in RFC 821
TCP	Transport Control Protocol
US-ASCII	The US ASCII character set, defined in ANSI X3.4: 1986

Table 2: Terms

SET Initiation

SET does not specify initiation

SET does not specify how a SET transaction is initiated. The introduction of other electronic commerce protocols will likely have the effect of interposing other applications between SET applications.

For example, users may shop using a shopping protocol and application that allows them to accumulate the OD on their local machine. Upon completing their shopping, they may negotiate the price or the payment method with the merchant via a payment selection protocol and application. In this scenario, the SET application would be invoked by some later application, which would pass it the necessary initiating purchase information.

Lack of supporting protocols

As of this publication, shopping and payment selection protocols have not been defined. In their absence, this document specifies a mechanism that provides SET Cardholder implementations with a common, interoperable methodology and allows them to work with currently available shopping mechanisms (such as World Wide Web dialogues).

This mechanism can also provide interoperability between other SET entities.

SET Initiation Issues

Overview

The following issues must be addressed in initiating the SET protocol:

- Order Description (OD)
- Initiation responses
- Language
- Character set
- Confidentiality

Order Description

SET is designed to transport financial data; it explicitly does not communicate other information such as the OD. The exchange of such information falls within the scope of other electronic commerce protocols (such as shopping protocols). SET defers the communication of shopping-related information to such protocols.

However, in order to initiate a SET payment, the Merchant and Cardholder must agree on the OD and the amount of the purchase. This data must be available to the SET application. At present, there is neither a standardized mechanism for accumulating the OD at the Cardholder nor one for transmitting the OD and purchase amount from the Merchant to the Cardholder.

Eventually, shopping applications and protocols may exist that allow the Cardholder to accumulate ODs on their local machine over the course of a shopping session or that allow them to exchange this information with the Merchant securely and confidentially. Until these electronic commerce protocols and applications become available, Merchants should coordinate purchase information with the Cardholder by sending it to the Cardholder in the "SET initiation" message described in Part II on page 7.

SET applications should display the OD and amount to the user so that the user may verify them before committing to the payment.

Order Description format

SET requires that all Cardholder and Merchant implementations support the OD in "text/plain" format. By mutual agreement, implementations can support alternate formats. Cardholder and Merchant implementations may use the initiation response mechanism to coordinate their choice of OD format.

Initiation responses

When an implementation receives a SET initiation message, it normally sends a SET message such as **PInitReq** in response. In some situations, the responding application (the responder) must exchange additional data with the application that sent the initiation message (the initiator) before actually generating the first SET message. Instead of sending a SET message, the responder may optionally send an initiation response message as described on page 45. Upon receiving the initiation response message, the initiator may choose to send another SET initiation message based on the data received from the responder. Another message may be warranted if, for example, the response indicated that an alternate character set should be used. This flow is illustrated in Figure 5 on page 45.

Continued on next page

SET Initiation Issues, continued

Language

The OD and other fields (such as extensions) may be expressed in a specific language. SET implementations display but do not interpret these text fields and are thus insensitive to the language.

Support for choosing a language is included for the convenience of the user, so that initiation information may be displayed intelligibly.

Character set

There are two primary character sets for use in initiation fields.

Common name(s)	MIME name	Comments
ASCII	us-ascii	All SET implementations must support the US-ASCII character set as a basis for interoperability.
ISO 10646 Basic Multilingual Plane BMPString Unicode	iso-10646-ucs-2	SET also permits the use of the ISO 10646 (Basic Multilingual Plane) character set for initiation fields. When appearing in an initiation message field, these characters use the RFC 2047 *encoded-word* mechanism.

Table 3: Character Sets

Additional character sets may used. In particular, the Cardholder and Merchant software may choose an alternate character set for the order description. Implementations may use the initiation response mechanism to synchronize their choice of character sets.

Confidentiality

The payment initiation mechanism does not provide confidentiality for payment parameters such as price and OD. The confidentiality of this information is beyond SET's scope, which is explicitly limited to payment data. Confidentiality should be achieved by appropriate use of channel encryption mechanisms such as SSL or IPSEC.

Part II
SET Initiation Messages

Overview

Purpose

Part II describes SET initiation and initiation response messages.

Organization

Part II includes the following topics:

Chapter	Topic		Page
1	SET Initiation Message Structure	Discusses the general structure of the SET initiation message.	8
2	Registration Initiation Messages	Describes the SET registration initiation message that triggers the SET protocol.	21
3	Payment Initiation Messages	Describes the SET payment initiation message that triggers the SET protocol.	32
4	Initiation Response Message	Describes an optional message that a SET entity can send in response to a registration or payment initiation message.	45

Chapter 1
SET Initiation Message Structure

Overview

Types of initiation messages

The following SET initiation messages are defined:

Registration-Initiation	initiates a registration transaction
Registration-Inquiry-Initiation	initiates an inquiry about a registration transaction
Payment-Initiation	initiates a payment transaction
Payment-Inquiry-Initiation	initiates an inquiry about a payment transaction

(An initiation response message is also defined, and is described on page 45.)

SET initiation message structure

Each SET initiation message, as well as the initiation response message, is defined as a well-formed MIME message (for details about MIME, see Appendix A on page 98).

Reminder: If you are not familiar with MIME, please read Appendix A before you read the rest of this chapter.

Continued on next page

Overview, continued

Initiation message header

The initiation message header includes a number of special SET-defined fields described below. The US-ASCII character set is normally used for these fields, but the field bodies may also be expressed in the ISO 10646 character set via the RFC 2047 *encoded-word* mechanism.

Fields that do not apply in the electronic mail context are noted.

It is expected that values for SET fields that are not present in the initiation message will be obtained through other means, such as requesting the user to enter the information or by obtaining the information from the local disk.

An unrecognized initiation header field should be ignored.

Display of non-ASCII characters

A SET implementation that cannot display all ISO 10646 characters in a message should substitute characters from the available character set(s) according to local custom, such as:

- replace the characters with question marks (or some other arbitrary character);
- display the hexadecimal representation; or
- display the ASCII character corresponding to the low-order seven bits.

Since any choice made will result in the display of useless nonsense to the average user, SET implementations should avoid sending characters that do not correspond to a negotiated character set.

Initiation message body

Only the payment request initiation message contains a message body (the OD). The message body is omitted for all other initiation messages. When the OD is present, its format and character set are indicated by the initiation message **Content-Type** header.

Continued on next page

Overview, continued

Additional message wrappings

The transport mechanisms described in Part III (page 52) may add additional message wrappings to the initiation messages. For example, HTTP (page 54) uses a header similar to a MIME header. When the transport mechanism adds a header (or wrapper), the header (or wrapper) may contain header fields with the same name as those in the initiation message, such as **Content-Type** and **Content-Length**.

Figure 1 shows an example of a payment initiation message with such header fields.

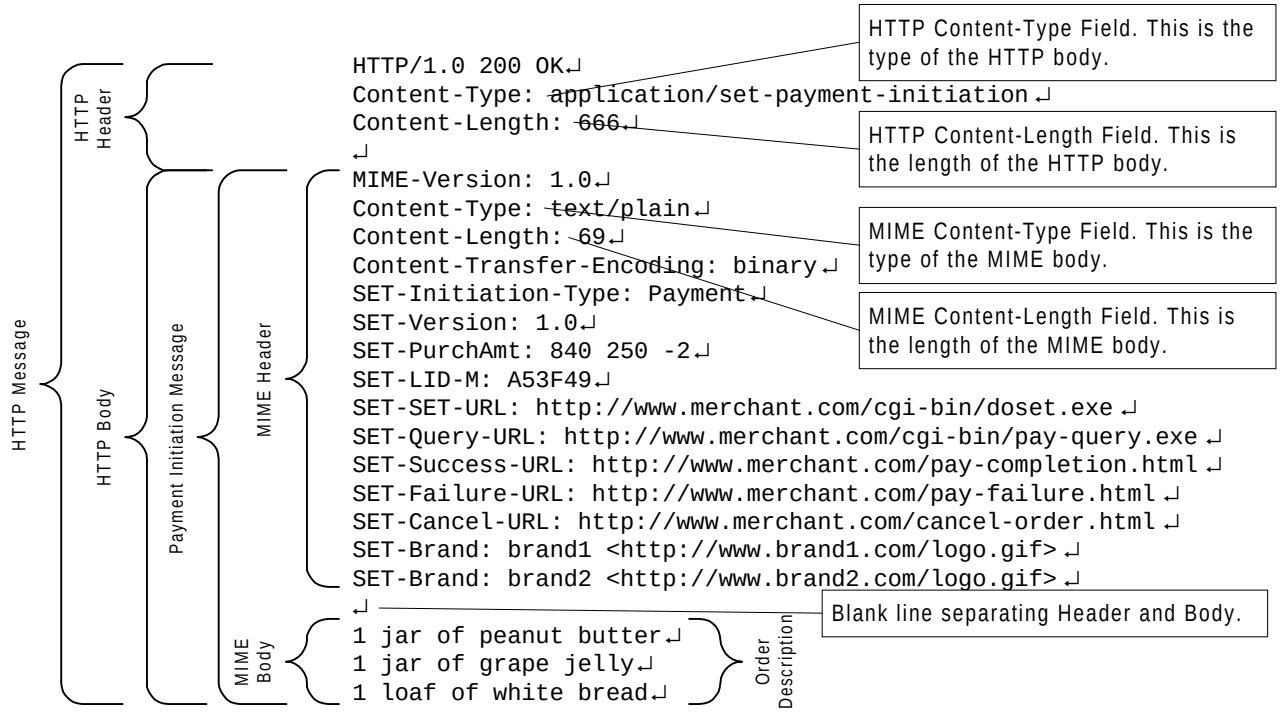

Figure 1: Additional Wrapping of Initiation Message

Content-Type and Content-Length fields

Note which **Content-Type** and **Content-Length** fields refer to which body. The "outer" header (the HTTP header) is processed by the transport layer, which passes the "outer" body (the HTTP body) on to the SET Initiation Processing layer. This "outer" body is the SET Initiation message, which itself consists of a MIME header and a MIME body.

Initiation Message Header Field Types

Globally defined data types

The following data types are defined for the values of initiation header fields. The values are always represented using US-ASCII characters. It may be necessary to convert the data before using it for other purposes.

date	A numeric string formatted as YYYYMMDD.
media-name	A media name that conforms to the requirements of RFC 2045. It is of the format *type* "/" *subtype* such as *text/plain*. The text is not case sensitive.
name	A name that conforms to the requirements of RFC 822. The text is not case sensitive.
language	A language name that corresponds to the requirements of RFC 1766. This is normally a 2-letter ISO 639 language code. The text is not case sensitive.
country	A country name that corresponds to the requirements of RFC 1766. This is normally a 2-letter ISO 3166 country code. The text is not case sensitive.
number	A string of digits representing 0 or a positive decimal value.
octet-string	A string consisting of the hexadecimal digits 0123456789ABCDEF. A pair of digits correspond to one octet. The string must contain an even number of hexadecimal digits. Octets may be separated by spaces; that is, 00 01 is equivalent to 0001. The text is case sensitive; only upper case characters are permitted.
text	A case-sensitive string of characters, normally in US-ASCII. The RFC 2047 *encoded-word* mechanism can be used for other character sets. For example, the ISO 10646 string for the Japanese characters ビッグ・デパート could be represented as =?iso-10646-ucs-2?Q?=30=D3=30=C3=30=B0=30=FB =30=C7=30=D1=30=FC=30=C8?.
url	A Uniform Resource Locator as defined in RFC 1738. The host name part of the *url* is not case sensitive; the path part is case sensitive.

Table 4: Globally-defined Data Types

Continued on next page

Initiation Message Header Field Types, continued

ASN.1 data types

The following additional data types are defined to correspond to types used in ASN.1.

BOOLEAN	"TRUE" or "FALSE". The text is case sensitive; only upper case characters are permitted.
BrandID	A *text* string that is optionally followed by a *url* enclosed with "<" and ">". The *text* string consists of a brand identifier and an optional product type. The *url* is the source for the logo associated with the brand. The full definition is: `brandId[:product] ["<" url ">"]` *brandId* and *product* are case sensitive. *url* is as defined in Table 4 on page 11.
Currency Amount	A list of three integers separated by spaces that correspond to currency code, amount, and exponent. For example, the value US$2.50 would be represented as 840 250 -2.
OBJECT-IDENTIFIER	A series of *numbers* separated by spaces.

Table 5: ASN.1 Data Types

Common Initiation Header Fields

Common fields

Header fields that are common to multiple initiation messages are described below. Additional details about the usage of these fields are provided in the description of each message.

The following header fields are discussed:

- **MIME-Version**
- **SET-Initiation-Type**
- **SET-Version**
- **Content-Language**
- Uniform Resource Locators
- **SET-Brand**
- Echo fields
- Extensions

MIME-Version must be the first header field. The other header fields may occur in any order.

MIME-Version

All initiation messages must have the following MIME version header:

```
MIME-Version: 1.0
```

Continued on next page

Common Initiation Header Fields, continued

SET-Initiation-Type

All initiation messages must have this field, which indicates the type of initiation message. The field is of type *name*. The following message types, which are defined in the chapters that follow, are defined:

	CCA, MCA, PCA	Payment Gateway	Merchant	Cardholder
INITIATION MESSAGES				
Registration-Initiation	**send**	**receive**	**receive**	**receive**
Registration-Inquiry-Initiation	**send**	**receive**	**receive**	**receive**
Payment-Initiation			**send**	**receive**
Payment-Inquiry-Initiation			**send**	**receive**
INITIATION RESPONSE MESSAGES				
Registration-Initiation-Response	*receive*	*send*	*send*	*send*
Registration-Inquiry-Initiation-Response	*receive*	*send*	*send*	*send*
Payment-Initiation-Response			*receive*	*send*
Payment-Inquiry-Initiation-Response			*receive*	*send*

Required action; this entity must be able to perform this action.
Optional action; this entity may be able perform this action, but is not required to do so.

Table 6: Initiation Message Support Requirements

Continued on next page

Common Initiation Header Fields, continued

SET-Version

This field is a list of version numbers separated by spaces. The list indicates the versions of SET messages supported by the system sending the message. Version numbers are expressed as:

```
version"."revision
```

where *version* is a number matching a major SET version, and *revision* is a number matching a minor SET revision. For example, "1.0" identifies the initial release of SET version 1 (with no revisions).

Content-Language

This field specifies the language used in the message. The language name may be useful in formatting non-ASCII characters contained either in header fields, the OD, or extensions. The **Content-Language** is specified in RFC 1766, "Tags for the Identification of Languages." Only one language should be specified. The format is:

```
language["-"country]
```

For example, to specify that the initiation message is expressed in Japanese:

```
Content-Language: ja
```

If unspecified, **Content-Language** is unknown; the SET application should format text according to the conventions of the user's native language.

Continued on next page

Common Initiation Header Fields, continued

Uniform Resource Locators

Each initiation message will include one or more URLs that correspond to actions that the application may take in response to conditions that arise while processing the message. The following fields, which are of type *url*, must be supported by all applications which receive initiation messages. Additional URLs may be defined to support special processing.

Field name:	Indicates the address to which the recipient should send:	Required or optional?
SET-SET-URL	a SET request message that corresponds to the initiation message. (Alternatively, a SET initiation response message may be sent to **SET-Response-URL**.)	Required if **SET-Response-URL** is omitted. If this field is omitted, the application should send a SET initiation response message to the **SET-Response-URL**.
SET-Response-URL	a SET initiation response message, as an alternative to sending a SET request message to the **SET-SET-URL**.	Required if **SET-SET-URL** is omitted. If this field is omitted, a SET initiation response message may not be sent.
SET-Query-URL	any subsequent SET inquiry request message.	Optional. If this field is omitted, the application should send the inquiry message to the **SET-SET-URL**.
SET-Success-URL	a standard HTTP request upon successful completion of the processing of the current exchange of SET messages.	Required.
SET-Failure-URL	a standard HTTP request upon communications failure or time-out that occurs before the final SET message in the current SET message exchange.	Optional. If this field is omitted, the application should send the HTTP request to **SET-Success-URL** instead.
SET-Cancel-URL	a standard HTTP request if the user indicates that the action corresponding the initiation message should be canceled.	Optional. If this field is omitted, the application should send the HTTP request to **SET-Success-URL** instead.
SET-Diagnostic-URL	a SET **Error** message according to the diagnostic log mechanism (see page 62).	Optional. If this field is omitted, the application should send the **Error** message to the **SET-SET-URL**. (Refer to SET Book 2: Programmer's Guide for restrictions on the frequency of such **Error** messages.)
SET-Service-URL	a standard HTTP request if the user needs customer service.	Required for Payment-Initiation and Payment-Inquiry-Initiation messages.

Table 7: Uniform Resource Locators (URLs)

Continued on next page

Common Initiation Header Fields, continued

Combinations of SET-SET-URL and SET-Response-URL

The presence or absence of **SET-SET-URL** and **SET-Response-URL** indicates whether the sender permits or demands a SET initiation response message:

If **SET-SET-URL** is:	and **SET-Response-URL** is:	Then:
present	absent	The sender cannot process a SET initiation response message, and the recipient must not send one.
present	present	The sender can accept either a SET request message or a SET initiation response message.
absent	present	The sender demands that the recipient send a SET initiation response message. For interoperability reasons, this demand should only be made when the sender is sure that the recipient can provide an initiation response (for example, as a result of receiving an initiation response message during a previous interaction).
absent	absent	This combination is not valid.

Table 8: Combinations of SET-SET-URL and SET-Response-URL

Repeated URL fields

The URL fields may be repeated with different values. If so, the different values are regarded as alternatives, which the recipient software may choose between as desired. The order of such alternatives does not imply an order of preference.

For example, a Merchant that supports SET through HTTP and SSL could specify:

```
SET-SET-URL: http://www.merchant.com/payment↵
SET-SET-URL: https://www.merchant.com/payment↵
```

Continued on next page

Common Initiation Header Fields, continued

SET-Brand

This field of type *BrandID* indicates a payment card brand that a Merchant or CA supports. This field may be repeated if multiple brands are supported. If so, the different values are regarded as alternatives, which the recipient software may choose between as desired. The order of such alternatives does not imply an order of preference.

Echo fields

Initiation messages and their responses support state information that is used by the system generating the message to identify the transaction. Either system involved in the transaction may include state information that the other system must repeat in any subsequent initiation message or response. These fields are of type *text*.

SET-Echo-In-Response	This field may optionally be included in an initiation message. If the field is present, the recipient must copy it to any initiation response message generated.
SET-Echo-In-Request	This field may optionally be included in an initiation response message. If the field is present, the recipient must copy it to the initiation message generated immediately after the initiation response message.

Table 9: Echo Fields

Continued on next page

Common Initiation Header Fields, continued

Extensions

Initiation messages may describe SET message extensions that apply to the processing of the message. A field that contains information about an extension has a name consisting of the following components:

- the prefix **SET-Ext-**
- one of the strings defined in Table 10 indicating which information about the extension is provided in the field,
- a name (unique for all extensions within a message) that is used to label related extension fields, and
- (for **SET-Ext-Data-...** only) an optional label preceded by a hyphen.

Implementations that support message extensions must also support these header fields.

Name	Description
OID-	This prefix designates a field of type *OBJECT-IDENTIFIER* that uniquely identifies the content of the message extension.
Mandatory-	This prefix designates a field of type *BOOLEAN* that indicates whether the receiving application must include the extension in the corresponding SET message. The default value of this field is FALSE. If this field is TRUE: • the originating SET application must include **SET-Response-URL** in the initiation message; and • if the receiving application does not recognize the extension, it must send a response message with the appropriate status code.
Data-	This prefix designates a field that contains some information about the value of the extension (or of a data element within the extension). The format of the field is defined for each message extension. The fields may use the optional label component of the field name described above. The label for each header field and their correspondence to the ASN.1 elements in the message extension are defined for each message extension.

Table 10: Extension Field Names

Continued on next page

Common Initiation Header Fields, continued

Extension example

As an example, a SET message extension called “promotional card name” might be defined as follows:

```
promotionalCardName EXTENSION ::= {
   SYNTAX PromotionalCardName
   IDENTIFIED BY { id-set-msgExt 0 }
}

PromotionalCardName ::= DirectoryString { 256 }
```

Then by abbreviating the name to “pcn”, the following extension header fields might be used:

```
SET-Ext-OID-pcn: 2 23 42 1 0↵
SET-Ext-Mandatory-pcn: FALSE↵
SET-Ext-Data-pcn: premium↵
```

Extension value

If the value of the extension is not defined in the initiation message (that is, if **SET-Ext-Data-...** is omitted), and the application supports the extension, it should determine the appropriate value to put into the SET message. In the example above, for instance, if the object identifier for the promotional card name extension is specified, the application should copy the promotional card name from the Cardholder certificate into the message extension.

Chapter 2
Registration Initiation Messages

Overview

Organization

This chapter includes the following topics:

Cardholder Registration

Initiation and registration flow

Figure 2 illustrates the usage of SET initiation messages and SET messages for Cardholder registration.

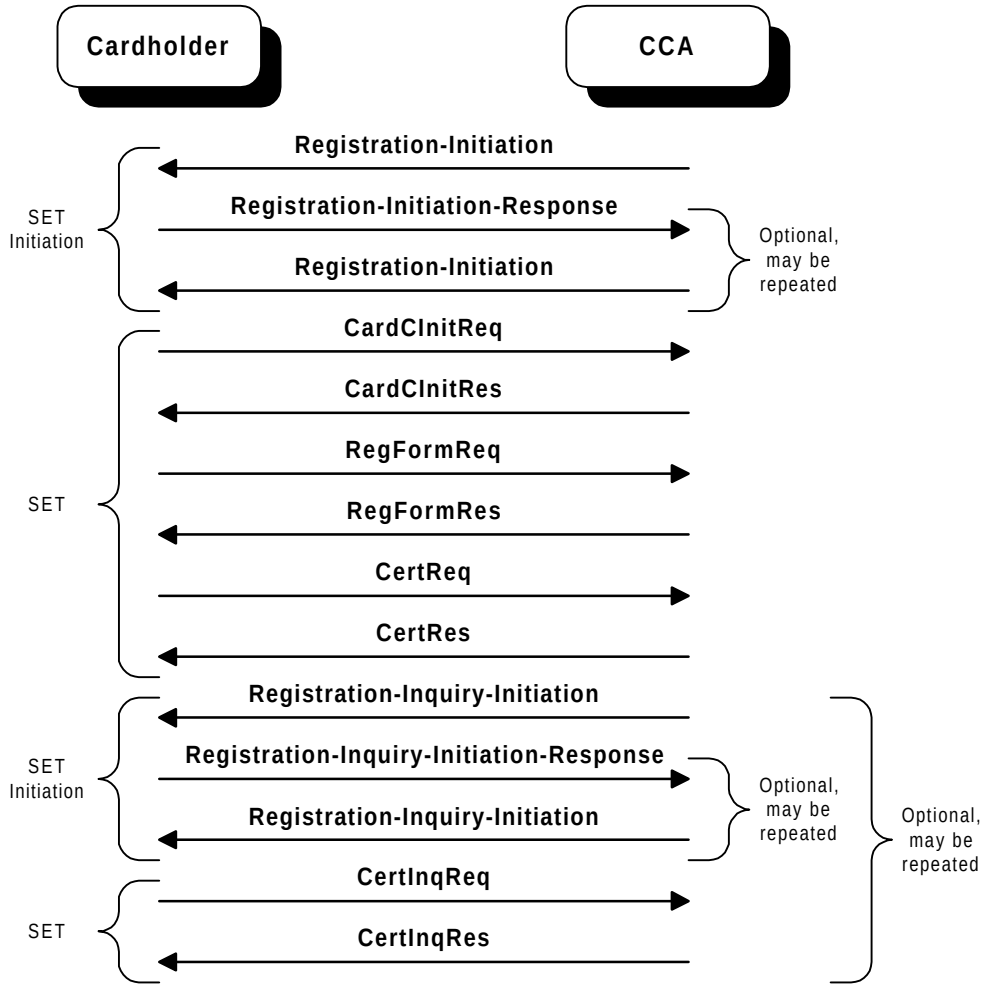

Figure 2: Cardholder Registration

Continued on next page

Cardholder Registration, continued

Initiation messages

In Figure 2, two (or more) Initiation messages are sent from the Cardholder Certificate Authority (CCA) to the Cardholder. These messages might be sent, for example, by a CCA Web server in response to a user's explicit request during an interactive Web session (such as by clicking a "Register" or "Inquiry" button on a Web page).

Cardholder registration messages

The Registration-Initiation message triggers the Cardholder application, which initiates the registration transaction by sending a **CardCInitReq** message to the CCA. Processing continues through the **CertRes**.

Cardholder registration inquiry messages

The Registration-Inquiry-Initiation message triggers the Cardholder application, which initiates a registration inquiry by sending a **CertInqReq** to the CCA.

For more detail

A more detailed discussion of SET registration flows may be found in Part II of SET Book 2: Programmer's Guide.

Merchant and Payment Gateway Registration

Initiation and registration flow

Figure 3 illustrates the usage of SET initiation messages and SET messages for Merchant and Payment Gateway registration.

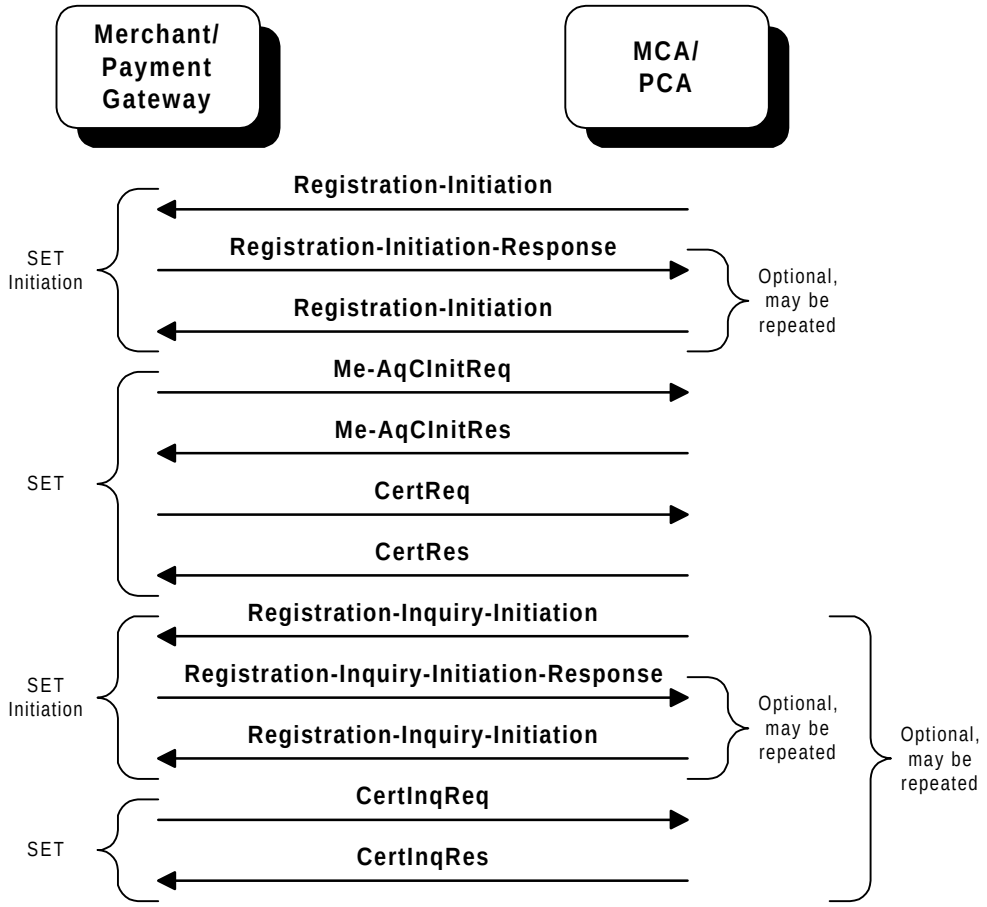

Figure 3: Merchant/Payment Gateway Registration

Continued on next page

Merchant and Payment Gateway Registration, continued

Initiation messages

In Figure 3, two (or more) initiation messages are sent from the Merchant/Payment Gateway Certificate Authority (MCA/PCA) to the Merchant/Payment Gateway. These messages might be sent, for example, by a MCA/PCA Web server in response to a Merchant/Payment Gateway's explicit request during an interactive Web session (such as by clicking a "Register" or "Inquiry" button on a Web page).

Merchant/ Payment Gateway registration messages

The Registration-Initiation message triggers the Merchant/Payment Gateway application, which initiates the registration transaction by sending a **Me-AqCInitReq** message to the MCA/PCA. Processing continues through the **CertRes**.

Merchant/ Payment Gateway registration inquiry messages

The Registration-Inquiry-Initiation message triggers the Merchant/Payment Gateway application, which initiates a registration inquiry by sending a **CertInqReq** to the MCA/PCA.

For more detail

A more detailed discussion of SET registration flows may be found in Part II of SET Book 2: Programmer's Guide.

Registration-Initiation Message Header

Header fields

The Registration-Initiation header consists of a subset of the common header fields defined on page 13. The following table lists the allowable fields in a Registration-Initiation:

Field Name	Required	Field Value
MIME-Version	Yes.	"1.0"
SET-Initiation-Type	Yes.	"Registration-Initiation"
SET-SET-URL	Only if SET-Response-URL is omitted.	Determined by the CA.
SET-Response-URL	Only if SET-SET-URL is omitted.	Determined by the CA.
SET-Query-URL	No.	Determined by the CA.
SET-Success-URL	Yes.	Determined by the CA.
SET-Failure-URL	No.	Determined by the CA.
SET-Cancel-URL	No.	Determined by the CA.
SET-Diagnostic-URL	No.	Determined by the CA.
SET-Service-URL	No.	Determined by the CA.
SET-Version	Yes.	Determined by the CA.
SET-Brand	Yes, at least one.	Determined by the CA.
SET-Echo-In-Response	No.	Determined by the CA.
SET-Echo-In-Request	Only if the previous initiation response message had one, then copied from that message.	Determined by the Cardholder, Merchant, or Payment Gateway.

Table 11: Registration-Initiation Message Header Fields

Registration-Initiation Message Body

No body

The Registration-Initiation message does not have a message body.

Sample Registration-Initiation Message

Sample message

The following is a sample Registration-Initiation message:

```
MIME-Version: 1.0↵
SET-Initiation-Type: Registration-Initiation↵
SET-SET-URL: http://www.CCA.com/cgi-bin/register.exe↵
SET-Query-URL: http://www.CCA.com/cgi-bin/query.exe↵
SET-Success-URL: http://www.CCA.com/register-completion.html↵
SET-Failure-URL: http://www.CCA.com/register-failure.html↵
SET-Cancel-URL: http://www.CCA.com/cancel-request.html↵
SET-Service-URL: http://www.CCA.com/cust-service.html↵
SET-Version: 1.0↵
SET-Brand: brand1 <http://www.brand1.com/logo/.>↵
SET-Brand: brand2 <http://www.brand2.com/logo/>↵
```

These messages are typically wrapped in a transport protocol such as HTTP or SMTP. The transport protocol generally adds its own header that is not shown in the sample. See, for example, page 59.

Registration-Inquiry-Initiation Message Header

Header fields

The Registration-Inquiry-Initiation header consists of a subset of the common header fields defined on page 13. The following table lists the allowable fields in a Registration-Inquiry-Initiation:

Field Name	Required	Field Value
MIME-Version	Yes.	"1.0"
SET-Initiation-Type	Yes.	"Registration-Inquiry-Initiation"
SET-SET-URL	Only if SET-Response-URL is omitted.	Determined by the CA.
SET-Response-URL	Only if SET-SET-URL is omitted.	Determined by the CA.
SET-Success-URL	Yes.	Determined by the CA.
SET-Failure-URL	No.	Determined by the CA.
SET-Cancel-URL	No.	Determined by the CA.
SET-Diagnostic-URL	No.	Determined by the CA.
SET-Service-URL	No.	Determined by the CA.
SET-Version	Yes.	Determined by the CA.
SET-LID-CA	No.	See below.
SET-Echo-In-Response	No.	Determined by the CA.
SET-Echo-In-Request	Only if the previous initiation response message had one, then copied from that message.	Determined by the Cardholder, Merchant, or Payment Gateway.

Table 12: Registration-Inquiry-Initiation Message Header Fields

SET-LID-CA

This optional field is of type *octet-string*. It specifies the CA's label for an ongoing registration transaction. If provided, this value must be converted to its binary equivalent and copied into the **LID-CA** field of the **CertInqReq**.

If this field is not specified, the application is assumed to have obtained the value of **LID-CA** through other means. For example, Cardholder software could ask the user to select the currently outstanding registration that should be the subject of the inquiry.

Registration-Inquiry-Initiation Message Body

No body

The Registration-Inquiry-Initiation message does not have a message body.

Sample Registration-Inquiry-Initiation Message

Sample message

The following is a sample Registration-Inquiry-Initiation message:

```
MIME-Version: 1.0↵
SET-Initiation-Type: Registration-Inquiry-Initiation↵
SET-SET-URL: http://www.CCA.com/cgi-bin/query.exe↵
SET-Success-URL: http://www.CCA.com/register-completion.html↵
SET-Failure-URL: http://www.CCA.com/register-failure.html↵
SET-Cancel-URL: http://www.CCA.com/cancel-request.html↵
SET-Service-URL: http://www.CCA.com/cust-service.html↵
SET-Version: 1.0↵
SET-LID-CA: 1234↵
```

These messages are typically wrapped in a transport protocol such as HTTP or SMTP. The transport protocol generally adds its own header that is not shown in the sample. See, for example, page 59.

Chapter 3
Payment Initiation Messages

Overview

Organization

This chapter includes the following topics:

Cardholder Payment

Initiation and payment flow

Figure 4 illustrates the usage of SET initiation messages and SET messages for payment transactions.

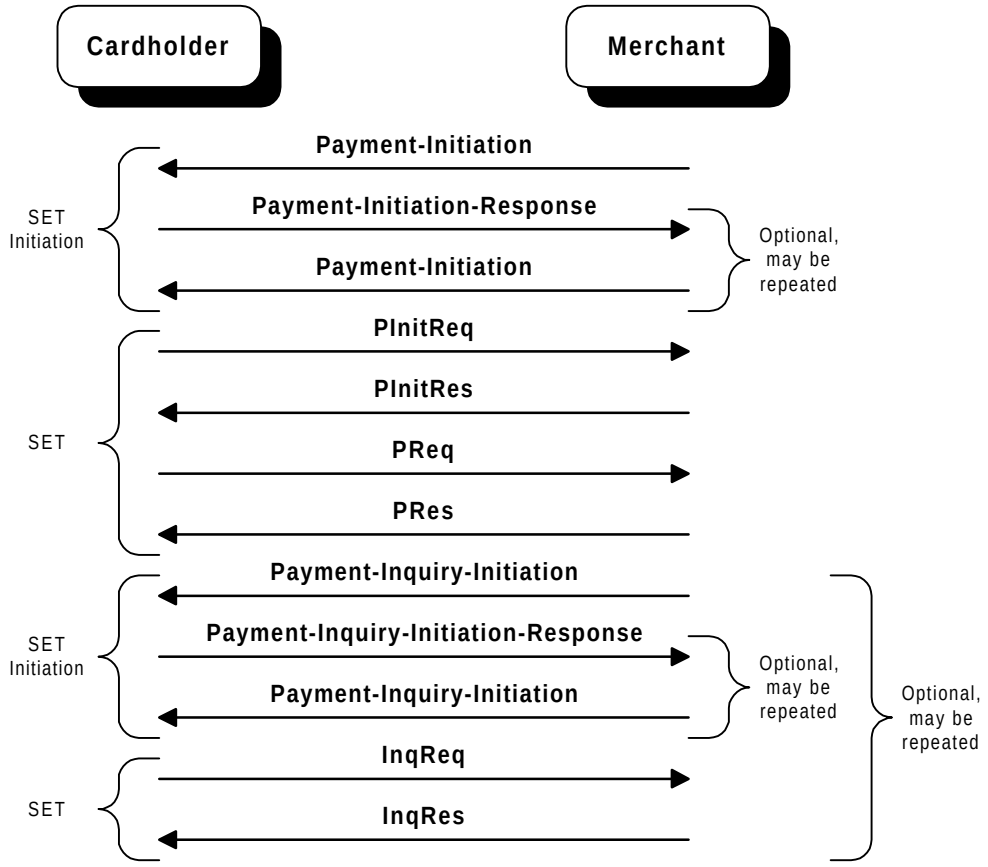

Figure 4: Cardholder Payment Transaction

Continued on next page

Cardholder Payment, continued

Initiation messages

In Figure 4, two (or more) initiation messages are sent from the Merchant to the Cardholder. These messages might be sent, for example, by a merchant Web server in response to a user's explicit request during an interactive Web session (such as by clicking a "Pay" button on a Web page).

Payment initiation messages

The Payment-Initiation message triggers the Cardholder application, which initiates the payment transaction by sending a **PInitReq** message to the Merchant. Processing continues through the **PRes**.

Payment inquiry initiation messages

The Payment-Inquiry-Initiation message triggers the Cardholder application, which initiates a payment inquiry by sending a **PInqReq** to the Merchant.

For more detail

A more detailed discussion of SET payment flows may be found in Part III of SET Book 2: Programmer's Guide.

Payment-Initiation Message Header

Header fields

The Payment-Initiation header consists of a subset of the common header fields defined on page 13, and the additional fields **SET-PurchAmt**, **SET-LID-M**, **SET-InstallTotalTrans**, and **SET-Recurring**, as well as MIME fields **Content-Type**, **Content-Length**, and **Content-Transfer-Encoding**. The following table lists the allowable fields in a Payment-Initiation:

Field Name	Required	Field Value
MIME-Version	Yes.	"1.0"
Content-Type	Yes.	See below.
Content-Length	No.	The length of the OD. See below.
Content-Transfer-Encoding	Yes.	"Binary"
Content-Language	No.	Determined by the Merchant.
SET-Initiation-Type	Yes.	"Payment-Initiation"
SET-SET-URL	Only if SET-Response-URL is omitted.	Determined by the Merchant.
SET-Response-URL	Only if SET-SET-URL is omitted.	Determined by the Merchant.
SET-Query-URL	No.	Determined by the Merchant.
SET-Success-URL	Yes.	Determined by the Merchant.
SET-Failure-URL	No.	Determined by the Merchant.
SET-Cancel-URL	No.	Determined by the Merchant.
SET-Diagnostic-URL	No.	Determined by the Merchant.
SET-Service-URL	Yes.	Determined by the Merchant.
SET-Version	Yes.	Determined by the Merchant.
SET-LID-M	No.	See below.
SET-PurchAmt	Yes.	See below.
SET-InstallTotalTrans	No. Mutually exclusive with SET-Recurring.	See below.
SET-Recurring	No. Mutually exclusive with SET-InstallTotalTrans.	See below.

Table 13: Payment-Initiation Message Header Fields

Continued on next page

Payment-Initiation Message Header, continued

Header fields (continued)

Field Name	Required	Field Value
SET-Brand	Yes, at least one.	Determined by the Merchant.
SET-Ext-OID	No.	Determined by the Merchant.
SET-Ext-Mandatory	No.	Determined by the Merchant.
SET-Ext-Data	No.	Determined by the Merchant.
SET-Echo-In-Response	No.	Determined by the Merchant.
SET-Echo-In-Request	Only if the previous initiation response message had one, then copied from that message.	Determined by the Cardholder.

Table 13: Payment-Initiation Message Header Fields, continued

Continued on next page

Payment-Initiation Message Header, continued

Content-Type

This MIME field specifies the media type and sub-type of the OD in the message body (see page 39).

All SET applications must support "text/plain". If the **charset** parameter is omitted for a text type, the default character set of US-ASCII will be assumed.

Content-Length

This MIME field specifies the number of bytes in the message body, which contains the OD. The count starts immediately after the empty line (double CRLF) separating the message header from the message body.

If the **Content-Length** field is omitted, the order body extends from immediately after the double CRLF to the end of the connection.

Content-Transfer-Encoding

This MIME field specifies the encoding used.

To accommodate the OD, the value of this field must be *binary*. If the transport mechanism does not support *binary*, then the entire payment-initiation message must be encoded (using base64, for example) before transmission.

SET-LID-M

This optional field is of type *octet-string*. It contains the Merchant's label identifying the transaction. It may be used by the Merchant as an order number to associate the payment with other information. If provided, this value must be converted to binary form and copied into the **LID-M** field of the payment initiation request (**PInitReq**) and subsequent payment messages.

SET-PurchAmt

This field is of type *CurrencyAmount*. It is expressed in terms of the same three components (*currency*, *amount*, and *amtExp10*) used in the SET payment messages. See "Amount Fields" in SET Book 2: Programmer's Guide.

Continued on next page

Payment-Initiation Message Header, continued

SET-Install TotalTrans

This field of type *number* is used to authorize payment in installments (when the Cardholder's account will be charged multiple times with the total of all these charges representing the purchase price).

The value specifies the maximum number of permitted authorizations.

This field is mutually exclusive with the **SET-Recurring** field.

If the **SET-InstallTotalTrans** field is specified, the **SET-PurchAmt** field specifies the total amount of all authorized installments.

SET-Recurring

This field is used to authorize recurring payments. It is expressed in terms of two components (*recurringFrequency* and *recurringExpiry*) used in the SET payment messages. The *recurringFrequency* component is of type *number* and the *recurringExpiry* component is of type *date*; the components are separated by one or more spaces. See "InstallRecurData" in SET Book 2: Programmer's Guide. An example **SET-Recurring** field is

```
SET-Recurring: 31 19960223
```

The **SET-Recurring** field is mutually exclusive with the **SET-InstallTotalTrans** field.

If the **SET-Recurring** field is specified, the **SET-PurchAmt** field specifies the total amount of all authorized installments.

SET-Brand

This field specifies a card brand accepted by the merchant. It is of type *BrandID*. It may be repeated for each brand that the merchant accepts. The SET application may read the **SET-Brand** fields, display to the user the names of cards that match a brand ID on the brand list, and allow the user to select a card for payment. The application may also display the brand logo beside or in place of each merchant-accepted card name. Clicking on the brand logo may be an easier interface for many users.

Payment-Initiation Message Body

Order Description

In a Payment-Initiation message, the OD is passed as the body of the message. The OD may take any format, including plain text and application-specific spreadsheets. The format is determined by the **Content-Type** field in the message header. All SET implementations must support “text/plain” in order to provide a least-common-denominator format for interoperability. Other formats may be supported.

If the OD format is text, the character set may be specified as a parameter to the **Content-Type** header, using the format defined in RFC 2045. SET implementations are required to support US-ASCII for basic interoperability. Support for ISO 10646 and other character sets is encouraged.

The OD must satisfy MIME requirements. In particular, ODs encoded as text must use CRLF as the line terminator regardless of native platform conventions. The final line of an OD encoded as text may be terminated with a CRLF but is not required to be.

SET ensures that the Cardholder and Merchant agree on this OD by including a hash of the OD in the **PReq** message. Specifically, all bytes in the initiation message body are hashed, from the first character after the double CRLF that ends the message header through the end of the body. If specified, the **Content-Length** field covers exactly the same bytes as are hashed.

Sample Payment-Initiation Message

Sample message

The following is a sample Payment-Initiation message:

```
MIME-Version: 1.0↵
Content-Type: text/plain↵
Content-Transfer-Encoding: Binary↵
SET-Initiation-Type: Payment-Initiation↵
SET-SET-URL: http://www.merchant.com/cgi-bin/doset.exe↵
SET-Query-URL: http://www.merchant.com/cgi-bin/pay-query.exe↵
SET-Success-URL: http://www.merchant.com/pay-completion.html↵
SET-Failure-URL: http://www.merchant.com/pay-failure.html↵
SET-Cancel-URL: http://www.merchant.com/cancel-order.html↵
SET-Service-URL: http://www.merchant.com/cust-service.html↵
SET-Version: 1.0↵
SET-PurchAmt: 840 250 -2↵
SET-LID-M: A53F49↵
SET-Brand: brand1 <http://www.brand1.com/logo/>↵
SET-Brand: brand2 <http://www.brand2.com/logo/>↵
↵
1 jar of peanut butter↵
1 jar of grape jelly↵
1 loaf of white bread↵
```

These messages are typically wrapped in a transport protocol such as HTTP or SMTP. The transport protocol generally adds its own header that is not shown in the sample. See, for example, page 60.

Payment-Inquiry-Initiation Message Header

Header fields

The Payment-Inquiry-Initiation header consists of a subset of the common header fields defined on page 13, and the additional fields **SET-LID-M**, **SET-LID-C**, **SET-XID**, and **SET-PaySysID**. The following table lists the allowable fields in a Payment-Inquiry-Initiation:

Field Name	Required	Field Value
MIME-Version	Yes.	"1.0"
SET-Initiation-Type	Yes.	"Payment-Inquiry-Initiation"
SET-SET-URL	Only if SET-Response-URL is omitted.	Determined by the Merchant.
SET-Response-URL	Only if SET-SET-URL is omitted.	Determined by the Merchant.
SET-Success-URL	Yes.	Determined by the Merchant.
SET-Failure-URL	No.	Determined by the Merchant.
SET-Cancel-URL	No.	Determined by the Merchant.
SET-Diagnostic-URL	No.	Determined by the Merchant.
SET-Service-URL	Yes.	Determined by the Merchant.
SET-Version	Yes.	Determined by the Merchant.
SET-LID-M	Yes.	See below.
SET-LID-C	No.	See below.
SET-XID	No.	See below.
SET-PaySysID	No.	See below.
SET-Ext-OID	No.	Determined by the Merchant.
SET-Ext-Mandatory	No.	Determined by the Merchant.
SET-Ext-Data	No.	Determined by the Merchant.
SET-Echo-In-Response	No.	Determined by the Merchant.
SET-Echo-In-Request	Only if the previous initiation response message had one, then copied from that message.	Determined by the Cardholder.

Table 14: Payment-Inquiry-Initiation Message Header Fields

SET-LID-M

This optional field is of type *octet-string*. It contains the Merchant's label identifying the transaction. It may be used by the Merchant as an order number to associate the payment with other information. If provided, this value must be converted to binary form and copied into the **LID-M** field of the payment inquiry request (**InqReq**).

If this field is not specified, the Cardholder software should use the value of **LID-M** obtained during the prior payment processing.

Continued on next page

Payment-Inquiry-Initiation Message Header, continued

SET-LID-C

This optional field is of type *octet-string*. It contains the Cardholder's label identifying the transaction. It may be used by the Cardholder as an order number to associate the payment with other information. If provided, this value must be converted to binary form and copied into the **LID-C** field of the payment inquiry request (**InqReq**).

If this field is not specified, the Cardholder software should use the value of **LID-C** obtained during the prior payment processing.

SET-XID

This optional field is of type octet-string. It contains a globally unique ID identifying the transaction. If provided, this value must be converted to binary form and copied into the **XID** field of the payment inquiry request (**InqReq**).

If this field is not specified, the Cardholder software should use the value of **XID** obtained during the prior payment processing.

SET-PaySysID

This optional field is of type *octet-string*. For some brands, it contains a unique identifier for the transaction from the time of authorization onward. If provided, this value must be converted to binary form and copied into the **PaySysID** field of the payment inquiry request (**InqReq**).

If this field is not specified, the Cardholder software should use the value of **PaySysID** obtained during the prior payment processing.

Payment-Inquiry-Initiation Message Body

No body	The Payment-Inquiry-Initiation message does not have a message body.

Sample Payment-Inquiry-Initiation Message

Sample Message

The following is a sample Payment-Inquiry-Initiation message:

```
MIME-Version: 1.0↵
SET-Initiation-Type: Payment-Inquiry↵
SET-SET-URL: http://www.merchant.com/cgi-bin/pay-query.exe↵
SET-Success-URL: http://www.merchant.com/pay-completion.html↵
SET-Failure-URL: http://www.merchant.com/pay-failure.html↵
SET-Cancel-URL: http://www.merchant.com/cancel-order.html↵
SET-Service-URL: http://www.merchant.com/cust-service.html↵
SET-Version: 1.0↵
SET-LID-M: A53F49↵
```

These messages are typically wrapped in a transport protocol such as HTTP or SMTP. The transport protocol generally adds its own header that is not shown in the sample. See, for example, page 60.

Chapter 4
Initiation Response Message

Overview

Usage

A SET entity that receives an initiation message may optionally send an initiation response message rather than a SET message. The initiation response message provides a method for the responding application (the responder) to defer the initiation of the SET protocol, and instead send additional information to the application that sent the initiation message (the initiator). The initiator may then choose to send another initiation message based on the information provided in the initiation response message. Another message may be warranted if, for example, the response indicated that an alternate character set should be used.

The discussions in this chapter focus on Cardholder–Merchant communications, but the initiation response message may be sent by any SET entity in response to any SET initiation message.

The initiation response process is illustrated in Figure 5.

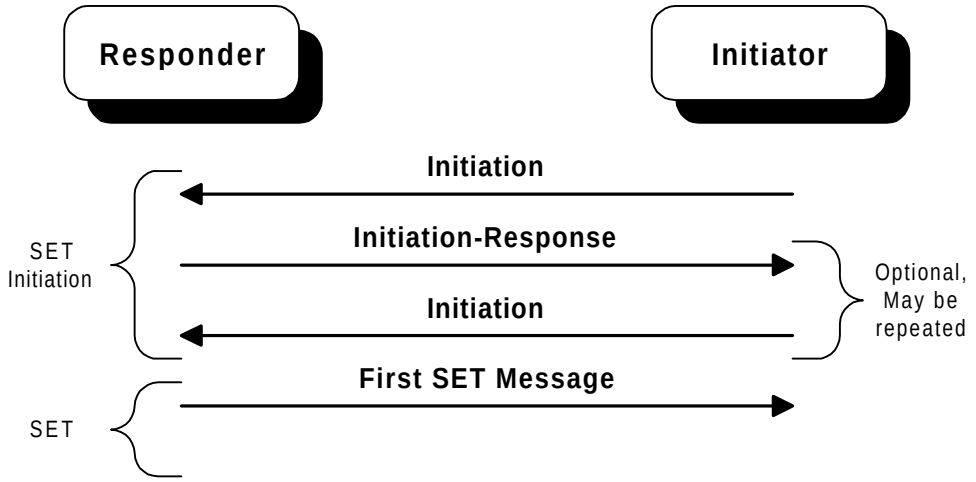

Figure 5: Initiation Response Message Flow

Iterations

To avoid loops, implementations should set a limit on the number of initiation messages sent for a single payment. This limit is determined by application requirements, but typically will be a small number such as 2 or 3.

Normal response

An initiation message generally contains sufficient information for a SET entity to generate the corresponding SET message. If the entity is able to send the SET message based on the information, it should do so.

Continued on next page

Overview, continued

Conditions for initiation response

An implementation may not be able to generate a SET message based on the information contained in the initiation message. Possible reasons include:

- There are no cards of the listed brands available in a Cardholder wallet.
- The application is unable to display the OD in the defined format or character set.
- The initiation message is missing critical information.
- The initiation message contained an unrecognized mandatory extension.
- The initiation message does not contain a required header field.
- The initiation message contains additional fields (defined outside this specification) that indicate a response message is necessary.
- The initiation message includes **SET-Response-URL** but not **SET-SET-URL**, indicating that an initiation response message is expected.
- For any other reason, the Cardholder is unable to generate a SET message.

In these cases, the application may send an initiation response message indicating the situation if and only if a **SET-Response-URL** is included in the initiation message.

If the application is unable to generate a SET message and the initiation message does not contain a **SET-Response-URL** field, the application must notify the user or log an error, and otherwise ignore the initiation message.

Optional implementation

Support for the initiation response message is optional for all SET entities. If an initiator can accept an initiation response message, it indicates that by providing a **SET-Response-URL** header in the initiation message. A responder may always choose not to generate the initiation response message.

Initiation Response Message Header

Header fields

An initiation response header consists of a subset of the common header fields defined at page 13, and the additional fields **SET-Status**, **SET-Error-Field**, **SET-Accept-Language**, **SET-Accept-Content-Type**, and **SET-Accept-Charset**. The following table lists the allowable fields in a initiation response message:

Field Name	Required	Field Value
MIME-Version	Yes.	"1.0"
SET-Initiation-Type	Yes.	See below.
SET-Version	Yes.	"1.0"
SET-Status	Yes.	See below.
SET-Error-Field	Only if the initiation message had an error.	Determined by the responder.
SET-Accept-Language	No.	See below.
SET-Accept-Content-Type	Only if **SET-Status** contains *UnsupContent*.	See below.
SET-Accept-Charset	Only if **SET-Status** contains *UnsupCharset*.	See below.
SET-Echo-In-Response	No.	Determined by the initiator.
SET-Echo-In-Request	Only if the previous initiation response message had one, then copied from that message.	Determined by the responder.

Table 15: Initiation Response Message Header Fields

SET-Initiation-Type

The **SET-Initiation-Type** field of a response message contains the value from the initiation message with a suffix of *-Response*. For example, if the initiation message had a message type of *Payment-Initiation*, the response message would be *Payment-Initiation-Response*.

Continued on next page

Initiation Response Message Header, continued

SET-Status

This field is a list of numbers that indicates the status of processing the initiation request. More than one number can be provided to indicate that multiple conditions apply. The following values may be used:

Status		Meaning
0	***Success***	*Success (see below)*
1	***UnrecMsg***	*Unrecognized message*
2	***BadMsg***	*Malformed message*
3	***UnsupContent***	*Unsupported Content-Type in message body*
4	***UnsupCharset***	*Unsupported character set in message header or body*
5	***UnsupMsgExt***	*Unsupported mandatory message extension*
6	***UnsupVersion***	*No common version of SET for transporting messages*
7	***UnsupLanguage***	*Unsupported language specified in* **Content-Language** *field*
100-199		*Reserved for failure codes to be defined for the specific initiation message*
200-299		*Failure codes defined by extensions to the protocol (defined outside this specification)*

Table 16: SET-Status Field Values

Status code of Success

The status code of *Success* may be sent only when all the following conditions are true:

- the responder processes the initiation message correctly; and
- the responder is ready to continue processing by sending either the first SET message or another initiation response.

Initiation Response Message Header, continued

SET-Error-Field	This field contains a list of *field-names* separated by spaces. The list defines the field(s) that produced an error condition. To indicate a problem with the type or character set, the encoding, or the length of the initiation message body, use **Content-Type**, **Content-Transfer-Encoding**, or **Content-Length**.
SET-Accept-Language	This field consists of a set of languages separated by spaces. The list defines the supported languages, in the format specified for the **Content-Language** field (see page 15). This field must be included if the status is *UnsupLanguage* and may be included for other status values.
SET-Accept-Content-Type	This field contains a list of *media-names* separated by spaces. The list defines the content types supported for the initiation message body. This field must be included if the status is *UnsupContent* and may be included for other status values. This field is only meaningful in a response to a payment-initiation message.
SET-Accept-Charset	This field contains a list of *names* separated by spaces. The list defines the supported characters sets. This field must be included if the status is *UnsupCharset* and may be included for other status values.
Additional fields	The definition of SET initiation message may include additional fields that apply to responses for that message.

Initiation Response Message Body

No body	The initiation response message does not have a message body.

Sample Initiation Response Message

Sample message

The following is a sample initiation response message (to a Payment-Initiation request):

```
MIME-Version: 1.0↵
SET-Initiation-Type: Payment-Initiation-Response↵
SET-Version: 1.0↵
SET-Status: 3↵
SET-Error-Field: Content-Type↵
SET-Accept-Content-Type: text/plain↵
SET-Accept-Charset: us-ascii↵
```

These messages are typically wrapped in a transport protocol such as HTTP or SMTP. The transport protocol generally adds its own header that is not shown in the sample. See, for example, page 60.

Part III
Transport Mechanisms

Overview

Introduction

SET has been designed to place minimal requirements on the transport channel. SET requires only that the communications environment be able to carry arbitrary-length messages and provide reasonable reliability.

However, to encourage interoperability among vendors, it is desirable to have a standard approach for data transport between SET entities when they communicate over the Internet. It is strongly recommended that SET-enabled applications support these protocols in order to ensure interoperability.

This Part outlines the following communication mechanisms:

- HTTP — described in Chapter 1,
- SMTP — described in Chapter 2, and
- TCP — described in Chapter 3.

These methods are provided to ensure a guaranteed baseline set of interoperable transport mechanisms. Additional communication mechanisms may be defined in the future.

Continued on next page

Overview, continued

Approved content types

The following MIME content types have been approved by the IANA for use with SET:

application/set-registration-initiation	This media type is used for exchanging the SET registration initiation messages between a CA and an end entity: • Registration-Initiation • Registration-Inquiry-Initiation • Registration-Initiation-Response • Registration-Inquiry-Initiation-Response The SET registration initiation messages are described on page 21.
application/set-registration	This media type is used for exchanging SET messages related to registration: • **CardCInitReq/CardCInitRes** • **RegFormReq/RegFormRes** • **Me-AqCInitReq/Me-AqCInitRes** • **CertReq/CertRes** • **CertInqReq/CertInqRes** • **Error**
application/set-payment-initiation	This media type is used for exchanging the SET payment initiation messages between the Merchant and the Cardholder: • Payment-Initiation • Payment-Inquiry-Initiation • Payment-Initiation-Response • Payment-Inquiry-Initiation-Response The SET payment initiation messages are described on page 32.
application/set-payment	This media type is used for exchanging SET messages related to payment: • **PInitReq/PInitRes** • **PReq/PRes** • **InqReq/InqRes** • **AuthReq/AuthRes** • **AuthRevReq/AuthRevRes** • **CapReq/CapRes** • **CapRevReq/CapRevRes** • **CredReq/CredRes** • **CredRevReq/CredRevRes** • **PCertReq/PCertRes** • **BatchAdminReq/BatchAdminRes** • **Error**

Table 17: SET Content Types

Chapter 1
HTTP-based Transport

Overview

Introduction

The HyperText Transfer Protocol (HTTP) specifies an application level protocol designed for distributed, collaborative, hypermedia information systems. It is the protocol used by the World Wide Web. It is a request-response protocol that correlates very well to the request-response model of SET.

Purpose

This chapter specifies how an application must use the HTTP protocol for exchanging SET related messages between any two SET entities. The SET related messages are those described in Books 2 and 3 of the SET Specification and the initiation messages described on pages 21 and 32 of this document.

(For a description of user interaction on the Web, see Appendix B: World Wide Web Operation on page 104.)

Scope

The scope of the mechanism described here is limited to the exchange of SET related messages. It does not address the normal information exchange before SET is invoked and after SET has completed.

HTTP version

This document specifies the use of version 1.0 of the HTTP protocol as specified in RFC 1945.

Continued on next page

Overview, continued

Entities that can use HTTP

The mechanism described here is intended primarily for use on the link between the Cardholder and the Merchant, and between end entities and CAs. Consequently, the discussion and examples generally take the Cardholder–Merchant perspective. However, this mechanism should work equally well between Merchants and Acquirers with little or no change, and may be used on any SET transport link.

HTTP and MIME

HTTP follows the general outline for MIME but is more liberal in the details. For example, a MIME line must end with a CRLF pair. HTTP allows a line to end with CRLF, CR, or LF. Note that the HTTP variations from MIME only apply in the HTTP message wrapper. In particular, strict MIME is used within SET initiation messages. See RFC 1945 and Appendix A on page 98.

HTTP message

An HTTP message is very similar to a MIME message. It consists of a header, a blank line, and a body.

Character sets

For use in SET, US-ASCII must be used in the HTTP message header. When the HTTP body contains a SET initiation message, that initiation message may itself include alternate character sets as described on page 6.

Continued on next page

Overview, continued

Browser caching

HTTP service is often provided by means of a proxy mechanism. In the interest of performance, it is common practice for these proxy servers to cache data over many browser sessions. For SET, this means that the details of a private transaction may be mistakenly presented to other users of a proxy server that connect to the same Merchant.

A number of methods should be used to mitigate this risk. Merchants and Acquirers running HTTP servers offering SET services should take appropriate precautions to minimize this risk. Such precautions include:

- Setting page expiration to “Immediate” (very effective); and
- Using the “no-cache” pragma.

Below are samples of HTTP headers which implement the immediate expiration and no-cache methods:

```
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
```

HTTP Header

Header fields

HTTP uses the same format as MIME for its header fields. A field consists of:

field-name: *field-value*↵

Case sensitivity

HTTP field-names are not case sensitive. In SET, all field-bodies are not case sensitive unless otherwise specified.

Comments

Comments are *not* allowed within SET-defined fields; that is, any parentheses and the data contained between them is considered part of the value of the field.

Required header fields

In SET, the following HTTP header fields are required:

- **Content-Type**
- **Content-Length**

HTTP requests

In SET, all requests from a client to a server must use the HTTP "Full-Request" message format and the "POST" method of sending the message.

HTTP response

In SET, all responses from a server to a client must use the HTTP "Full-Response" message format. The response must begin with a status line, which consists of the following: "HTTP/1.0 *xxx*", where "xxx" is the 3 digit status code. More information may follow the 3 digit status code.

Continued on next page

HTTP Header, continued

Content-Type

This field specifies the type of contents being transported. When transporting a SET related message, the field value for **Content-Type** must be one of the following approved types:

- application/set-registration-initiation
- application/set-registration
- application/set-payment-initiation
- application/set-payment

See page 53 for more detailed information.

Content-Length

This field specifies the length in bytes (octets) of the body of the HTTP message. Each HTTP header must contain the following header field:

```
Content-Length: len↵
```

where “len” is the length of the body.

HTTP Samples

CCA registration initiation

Figure 6 is a sample registration initiation message sent from a CCA server to a Cardholder:

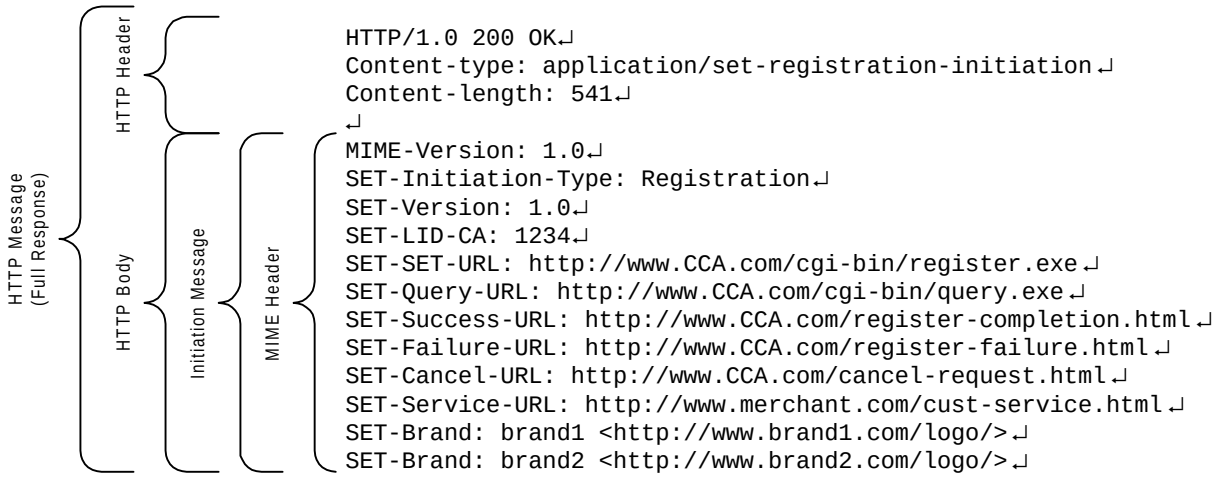

Figure 6: Sample HTTP Transport of Registration Initiation Message

Notice the following:

- The HTTP body is the Initiation message.
- The **Content-Length** is the length of the Initiation message.
- The Initiation message has a MIME header with an empty body.

Continued on next page

HTTP Samples, continued

Merchant payment initiation

Figure 7 is a sample payment initiation message sent from a Merchant server to a Cardholder:

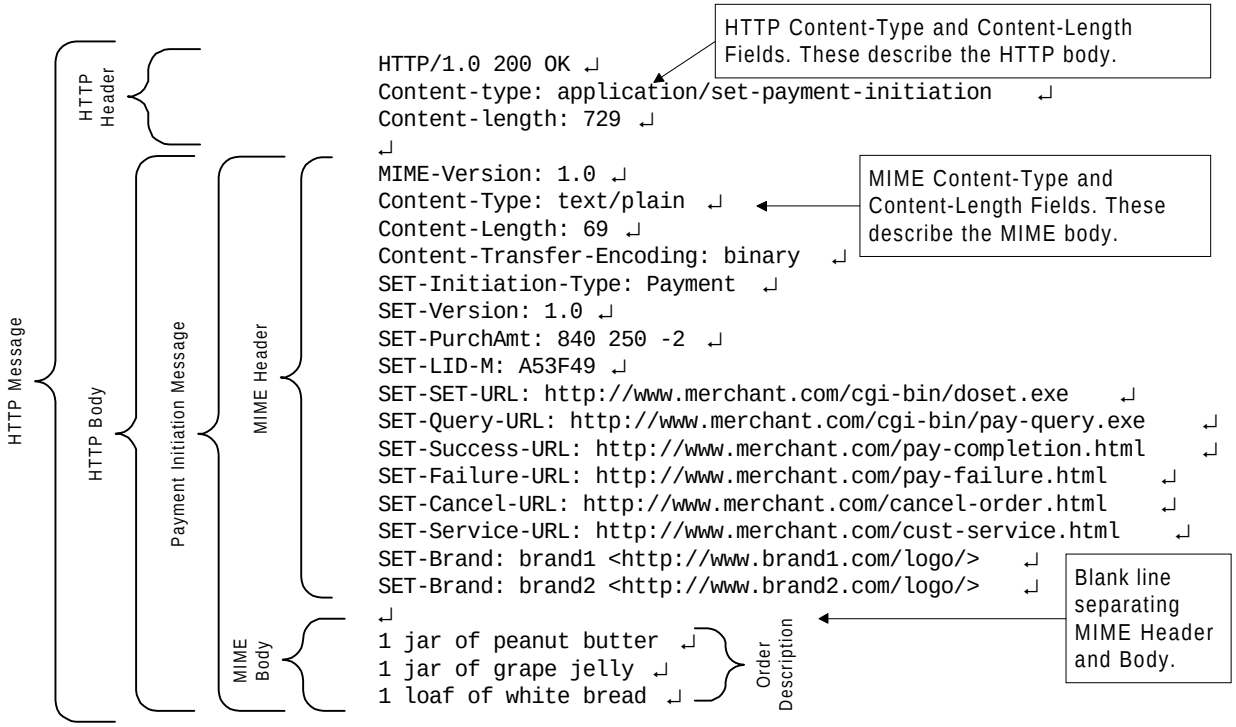

Figure 7: Sample HTTP Transport of Payment Initiation Message

Notice the following:

- The HTTP body is the Initiation message.
- The two **Content-Type** fields, two **Content-Length** fields, and what each one refers to.
- The Initiation message has a MIME header and a body that contains the OD.

Continued on next page

HTTP Samples, continued

Cardholder sending a SET request message

The following is a sample of how a Cardholder sends a SET request message to a Merchant:

```
POST uri HTTP/1.0↵
Content-Type: application/set-payment↵
Content-Length: xxx↵
↵
[DER-encoded SET-Request message goes here]
```

where "uri" is the Uniform Resource Identifier on the Merchant server (from the SET-SET-URL field in the initiation message), "xxx" is the length of the SET request message, and "SET-Request" is a DER-encoded SET message as described in Books 2 and 3 of the SET Specification.

Merchant sending a SET response message

The following is a sample of how a Merchant sends a SET response message to a Cardholder:

```
HTTP/1.0 200 OK↵
Content-Type: application/set-payment↵
Content-Length: xxx↵
↵
[DER-encoded SET-Response message goes here]
```

where "xxx" is the length of the SET response, and "SET-Response" is a DER-encoded SET message as described in Books 2 and 3 of the SET Specification.

Error messages

Use of Error messages

SET Error messages are used to indicate that a message cannot be reliably identified and processed.

Diagnostic log

If SET **Error** messages were used to signal errors in both directions of a connection, two SET systems might enter into an infinite loop of error messages. To avoid this, it is desirable to have a mechanism to communicate message failures happening in one of the two directions on the transport connection. The diagnostic log mechanism is intended to provide such a mechanism for the downstream connection.

When diagnostic log messages are sent via HTTP, the following method is specified:

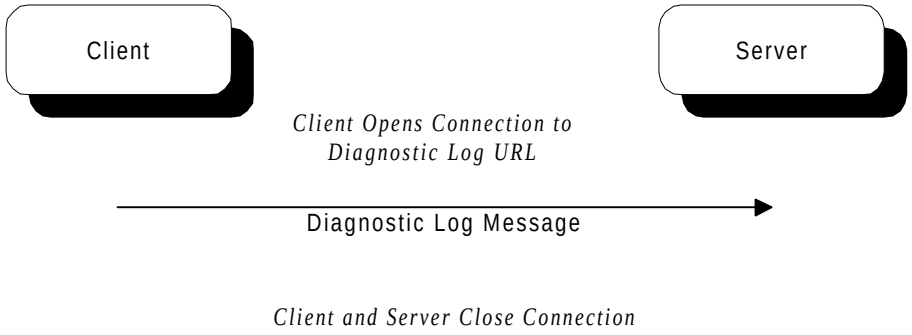

Figure 8: HTTP Diagnostic Log

When a SET entity receives an HTTP-transported SET message that fails basic SET message decoding (including DER-decoding errors and failure of signature verifications), the entity:

- generates a SET **Error** message;
- opens an HTTP connection to the server's diagnostic log URL (which was specified in the initiation message);
- posts the SET **Error** message; and
- closes the HTTP connection.

Note: Only one diagnostic log message may be sent per HTTP connection.

When a SET entity receives a diagnostic log message via HTTP:

- The entity does not respond, but simply closes the HTTP connection once the message is received.
- Processing of the diagnostic log message is implementation-dependent. It is strongly recommend that entities journal such messages in log files.

Chapter 2
SMTP-based Transport

Overview

Introduction

The Simple Mail Transport Protocol (SMTP) is an application level protocol designed for the reliable and efficient transfer of mail.

Purpose

This appendix specifies how the SMTP protocol is used for exchanging SET related messages between any two SET entities. The SET related messages are those described in Books 2 and 3 of the SET Specification and the initiation messages described on pages 21 and 32 of this document.

Scope

The scope of the mechanism described here is limited to the exchange of SET related messages. It does not address the normal information exchange before SET is invoked and after SET has completed.

SMTP version

This document specifies the use of the SMTP protocol as specified in RFC 821.

Issues

Issues

SMTP-based electronic mail has two significant limitations in its ability to support SET transactions:

No binary data	Not all SMTP servers support 8 bit and binary data.
Delayed delivery	Unlike HTTP, SMTP mail delivery does not always occur in a direct source-to-destination fashion. The SMTP protocol was designed to allow mail delivery even when it was not possible to find a route directly from one host to another or when there were transient network outages. Therefore, mail messages often pass through several intermediate "relay" hosts and sometimes encounter significant delays. This makes it difficult to support rapid interactive communication via e-mail.

Non-binary data

MIME encapsulation provides a standard method for resolving the binary issue: Each message has a specified *transfer-encoding*. Base64 is a standard MIME encoding for binary data in non-binary settings. In the e-mail case, the issue of non-binary data may be resolved by transmitting SET messages in Base64 encoding.

Delayed delivery

To mitigate the effects of delayed delivery, the two initial SET messages, **PInitReq** and **PInitRes**, may be skipped in the electronic mail case. To compensate:

- The Cardholder must provide the transaction-specific Merchant values normally provided in the **PInitRes** message. This data may be provided via some other media, such as CD-ROM or newspaper ad or created on the Cardholder's machine by custom Merchant software, which might be included on a CD-ROM.
- SET applications must provide a means to manually enter any Merchant information that cannot be automatically generated or that is provided via an alternative media. Please see the discussion of this transaction flow in Part III of SET Book 2: Programmer's Guide. All SET software that operates in the e-mail environment must support processing transactions without the **PInitReq** message in the purchase transaction flow.

Eliminating **PInitReq** and **PInitRes** reduces the number of messages required for a complete SET transaction, and thereby reduces the elapsed time to complete the transaction.

SMTP Interaction

Initial exchange

As in the Web scenario, the Merchant and the Cardholder must agree on the OD and the amount of the transaction. In the e-mail case, it is recommended that the Cardholder:

- include in the e-mail both the OD and the amount, using the initiation message described on page 32,
- format the data as a MIME-encapsulated message marked with the SET initialization MIME type *application/set-payment-initiation*, and
- if including the **PReq** message in the same e-mail, format the e-mail message as a *multipart/mixed* MIME message containing the SET message in one part and the information related to the order in another.

Continued on next page

SMTP Interaction, continued

Typical interaction

A typical SMTP interaction might proceed as follows:

Step	Action
1	Using custom-developed Merchant software, the user shops via a CD-ROM based catalog, selects goods, and selects SET as the payment method. The software develops the OD and total amount.
2	The user is allowed to choose a payment card account from his or her electronic wallet.
3	The custom-developed Merchant software creates a payment initiation message based on the OD and the amount.
4	The custom-developed software creates a **PReq** message based on the OD, amount, and the chosen payment card account.
5	The custom-developed Merchant software then passes the payment initiation message, **PReq** message, and electronic mail address of the Merchant to the customer's electronic mail application via a messaging API or similar channel.
6	Customer's mail software encodes the messages in base64 and creates a multipart electronic mail message with headers that contain the merchant's electronic mail address, message encoding type (that is, base64), and message MIME types.
7	Customer's mail software delivers the mail via the Internet.
8	Merchant receives the e-mail and detaches the payment initiation and SET messages. Merchant records the order information and amount from the initiation message, then decodes the SET message and processes it.
9	Merchant creates a **PRes** message, similarly attaches it to an electronic mail message and sends it back to the customer.
10	The customer's mail reader receives the mail, decodes the SET message portion back to pure binary form. Based on the message's MIME type, the mail software invokes the SET helper application on the SET message.
11	The SET application displays the results contained in the **PRes** message. It extracts the merchant's name and address from the merchant's certificate and displays them to the user so that the latter can know what organization signed the purchase response.

Table 18: Typical SET Interaction via SMTP

Chapter 3
TCP-based Transport

Overview

Organization

This chapter addresses the following topics:

Continued on next page

TCP-based Communication

Requirements

To ensure interoperability, Merchants and Payment Gateways using TCP for communications must use the method presented in this section. Although this method presumes communication between a Merchant and a Payment Gateway, nothing in this method, other than the security and performance considerations, precludes communications between any two SET parties. This method does not preclude multiple, parallel connection between the parties.

Benefits of TCP

TCP has a number of advantages over other low-level Internet protocols such as UDP.

- TCP is error-correcting. Most SET messages are signed, so errors are detected by SET implementations. However, error correction at the communications transport level will improve the overall performance and robustness of SET implementations in the face of error-prone communications environments.
- TCP permits arbitrary-length messages. Some SET messages may be longer than the 512-byte capacity guaranteed for UDP packets.

Using TCP for data communications according to the method described in this section offers several advantages compared to HTTP.

- The overhead of establishing and maintaining each TCP session may be amortized over multiple SET request/response pairs. The Merchant server and Payment Gateway may choose to hold a TCP session open across multiple requests, and multiple requests may be interleaved on one session.
- Session management functions are provided in a manner tailored to the needs of SET. These functions include echo, transport-level error indication, retry, and graceful close of sessions.
- A connection authentication mechanism is provided to deter one form of denial-of-service attack.

Data security

Note: TCP does not offer any security for the data transmitted within it. This is not an issue for SET messages, as the SET protocol provides all required security for message contents. However, any additional information exchanged via this TCP connection will not be protected.

Continued on next page

TCP-based Communication, continued

Interleaving

The TCP method specified in this section allows asynchronous communications. This allows the Merchant to interleave its requests. In other words, a Merchant may send another request right after the first without waiting for paired responses. The responses may come back in any order. If there is a communications failure, it is up to the Merchant to re-send requests that did not receive responses.

The maximum number of outstanding requests should be limited by a configuration parameter. (See page 88.)

The Merchant can have synchronous communications by always waiting for a response before sending the next request.

Merchant authentication (optional)

There is a certain amount of computational cost to processing even bogus or forged SET messages. Given the TCP method described here, one possible attack on the SET system is a denial of service attack in which a malicious entity floods the Payment Gateway with bogus messages with the intent of tying up its resources. The risk of this attack can be mitigated by authenticating the Merchant. Two simple authentication mechanisms are provided:

- TCP address authentication – authentication is accomplished by the Payment Gateway accepting connections only from certain TCP addresses, and rejecting all other connection attempts.
- Challenge authentication – authentication is accomplished via a challenge and response mechanism using shared secrets.

These two mechanisms may be optionally used in any combination and make it difficult for unauthorized parties to deliver messages to the Payment Gateway.

Connection States

Overview

This method defines the following connection states for a TCP connection between the Merchant and the Payment Gateway:

Connection State	Definition
Closed	No connection exists between the Merchant and the Payment Gateway.
Greeting	The Payment Gateway greets the Merchant, with an optional authentication challenge.
Authenticating	The Payment Gateway authenticates the Merchant.
Open	The Merchant and the Payment Gateway can exchange SET messages.
Closing	Either party has requested that the connection be closed.

Table 19: TCP Connection States

Diagram

Figure 9 illustrates these connection states.

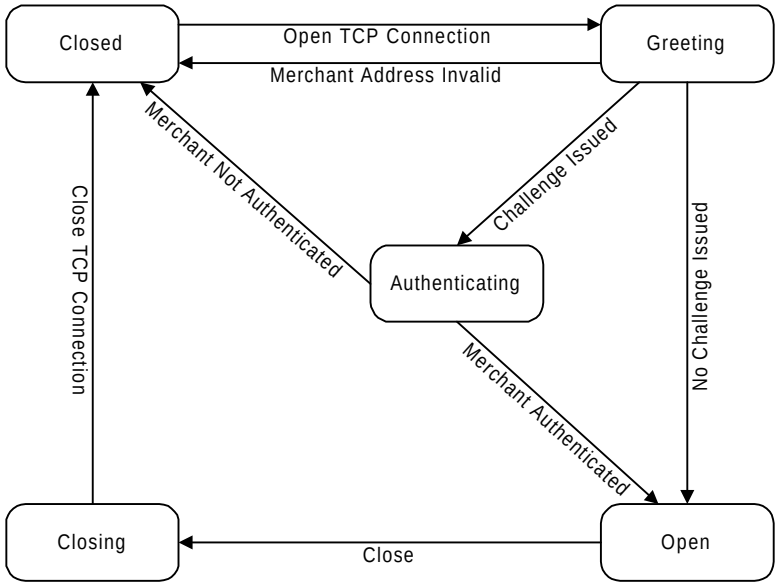

Figure 9: TCP Connection States

Figure 9 illustrates the expected operation of a connection. Network errors such as equipment failures will cause the state to transition to the *Closed* state.

Continued on next page

Connection States, continued

Connection events

The following events cause transition from one state to another:

Event	Transition
Open TCP Connection	Signals that the Merchant opened a TCP connection to the Payment Gateway.
Merchant Address Invalid	Signals that the Merchant's TCP address is invalid to the Payment Gateway.
Challenge Issued	Signals that the Payment Gateway issued a challenge in order to authenticate the Merchant.
No Challenge Issued	Signals that the Payment Gateway is not challenging the Merchant and has accepted the connection.
Close	Signals that either party sent a Close request, requesting that the connection be closed.
Close TCP Connection	Signals that both parties have terminated the connection between them.

Table 20: TCP Connection Events

Continued on next page

Closed State

Definition

This is the state when a connection does not exist between the Merchant and the Payment Gateway. For a connection to exist, the Merchant has to open a TCP connection, causing a transition to the *Greeting* state. The Merchant opens a TCP connection with the Payment Gateway if:

- the Merchant has a request to transmit,
- no available connection exists, and
- the configurable maximum number of outstanding TCP connections has not been exceeded (as discussed on page 97).

Greeting State

Definition

This state is entered when the Merchant opens a TCP connection with the Payment Gateway (this implies that the Payment Gateway is "listening" for a connection). If Merchant address authentication is required, the Payment Gateway will verify and only accept connections from specified TCP addresses. The Payment Gateway's TCP address and port, and the Merchant's TCP address (if required) are communicated out-of-band. (See page 97 for a list of networking attributes that must be agreed.)

Greeting message

When the Merchant opens a TCP connection, the Payment Gateway will respond by sending a Greeting message through the connection. The Greeting signals that the Payment Gateway is prepared to receive data. If the Greeting is not received by the Merchant in t_1 seconds, the connection will be closed. The default time-out value for t_1 is 30 seconds; this time-out value is configurable and communicated out-of-band. (See page 97.)

The Greeting consists of one of two messages:

Condition	Message
If the Merchant's address is valid, or if no TCP address authentication is required:	+OK|*text*|*challenge*↵
If the Payment Gateway performs TCP address authentication, and the Merchant's address is invalid:	-ERR|*text*↵

The fields of the message must always be present, and must consist of printable US-ASCII characters.

text	Optional text that the Payment Gateway may use to provide more information: any text the acquirer wants to send to the Merchant. It is recommended that, at a minimum, it include the Payment Gateway's host name or IP number.
|	field separator
challenge	The challenge the Payment Gateway sends to the Merchant for challenge authentication. If no challenge authentication is required, this field must be null.

If no challenge is issued (that is, if no authentication is required), the connection is established, and the state transitions to the *Open* state. If a challenge is issued (that is, if authentication is required), the state transitions to the *Authenticating* state.

Note: The Greeting message is not MIME-wrapped.

Continued on next page

Greeting State, continued

Example Greeting messages

OK message with a challenge:

```
"+OK|payment.gateway.com|12:34:56 01/23/45 payment.gateway.com↵"
```

OK message without a challenge:

```
"+OK|payment.gateway.com is ready|↵"
```

Error message with text:

```
"-ERR|Invalid merchant address↵"
```

Error message with no text:

```
"-ERR|↵"
```

Continued on next page

Greeting State, continued

Merchant in Greeting state

Figure 10 depicts the flow for the Merchant in the Greeting state:

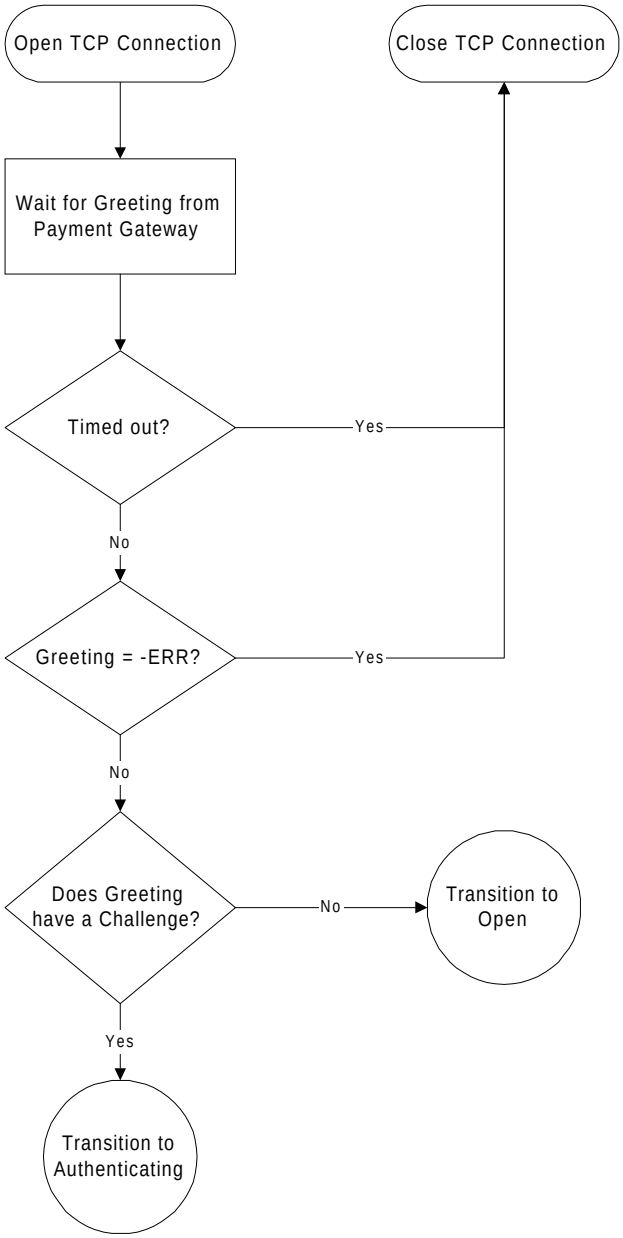

Figure 10: Merchant in TCP Greeting State

Continued on next page

Greeting State, continued

Payment Gateway in Greeting state

Figure 11 depicts the flow for the Payment Gateway in the Greeting state:

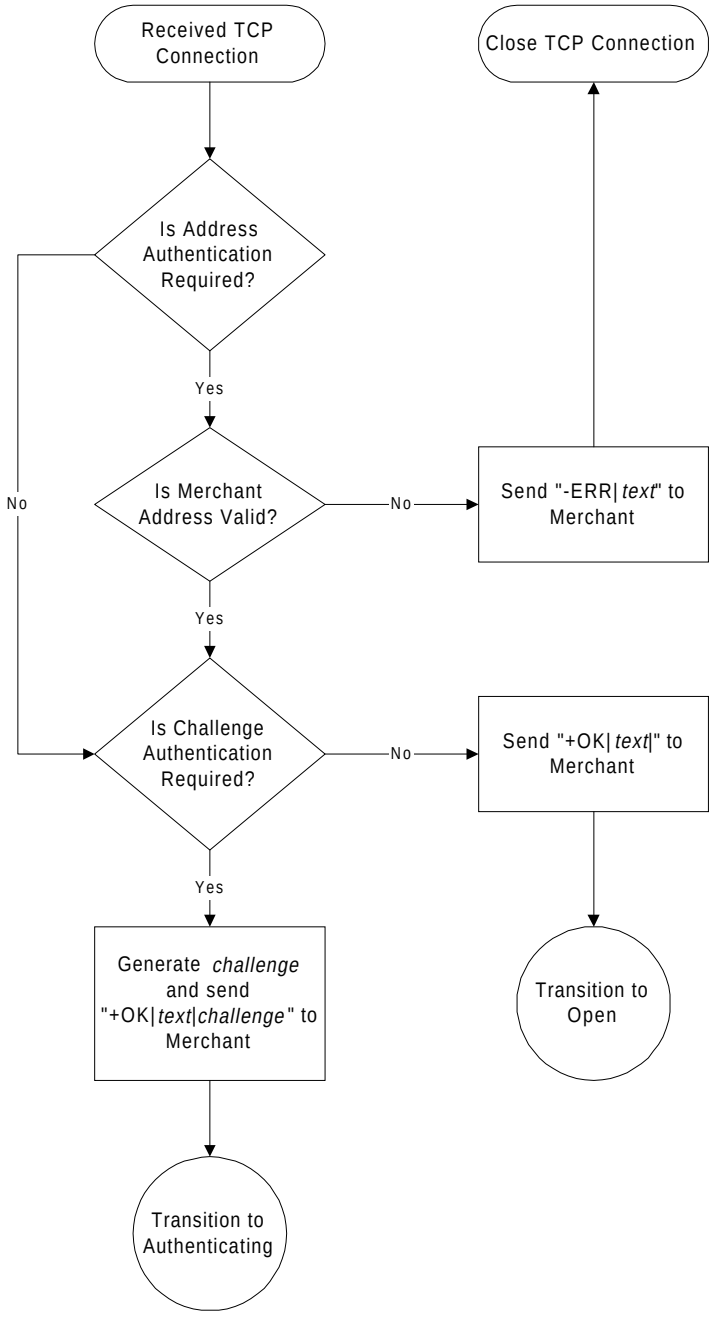

Figure 11: Payment Gateway in TCP Greeting State

Authenticating State

Definition

This is the state where the Merchant challenge authentication is performed. Authentication is accomplished via a challenge and response mechanism using a shared secret. This state is entered when the Payment Gateway issues a challenge to the Merchant in the Greeting message.

Authentication message

After issuing the challenge, the Payment Gateway waits for the Authentication message from the Merchant (with a time-out value t_1 as described on page 73).

The Authentication message consists of the following:

`Authentication|`*`transportId`*`|`*`authenticationResp`*`↵`

where:

transportId	a configurable transport layer ID for the Merchant that uniquely identifies the Merchant to the Payment Gateway. It may be anything from a pre-assigned value (for example, the Merchant's DNS name) to a certificate thumbprint.
authenticationResp	base64(HMAC(*Greeting*‖*transportID*, *shared-secret*))
Greeting	the complete Greeting message without the trailing CRLF
‖	the concatenation operator
transportID	the transportID in the Authentication message (without any surrounding whitespace)
shared-secret	the shared secret used as the key to the HMAC function (described in Part I of SET Book 2: Programmer's Guide). The shared secret may be anything but must be communicated out-of-band. (See page 97.)

Note: The Authentication message is not MIME-wrapped.

Continued on next page

Authenticating State, continued

Authentication-Reply message

After receiving the Authentication message, the Payment Gateway will verify the *authenticationResp* field and return the Authentication-Reply message to the Merchant. The Authentication-Reply consists of one of two messages.

Condition	Message	Action
If the authentication succeeded:	+OK|*text*↵	The state transitions to the *Open* state.
If the authentication failed:	-ERR|*text*↵	Both the Merchant and the Payment Gateway close the TCP connection and transition to the *Closed* state.

Table 21: Authentication-Reply Message

After sending the Authentication message, the Merchant waits for the Authentication-Reply message from the Payment Gateway (with a time-out value t_1 as above).

Note: The Authentication-Reply message is not MIME-wrapped.

Continued on next page

Authenticating State, continued

Merchant in Authenticating state

Figure 12 depicts the flow for the Merchant in the Authenticating state:

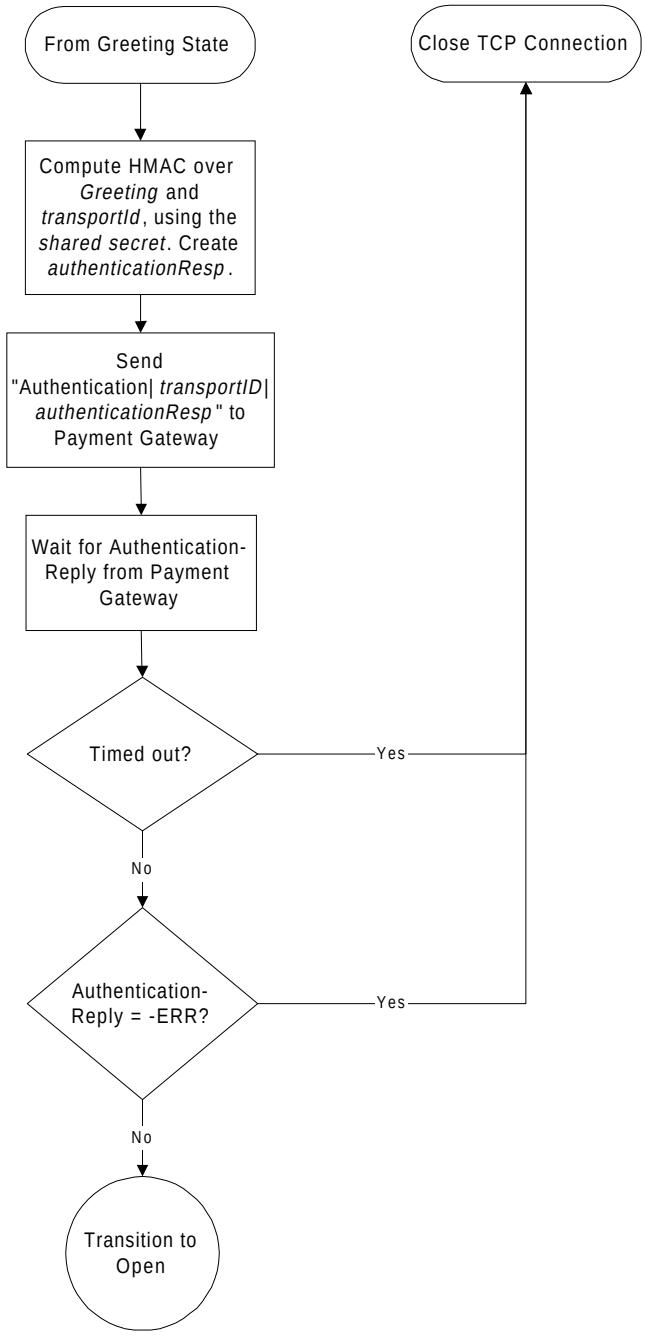

Figure 12: Merchant in TCP Authenticating State

Continued on next page

Authenticating State, continued

Payment Gateway in Authenticating state

Figure 13 depicts the flow for the Payment Gateway in the Authenticating state.

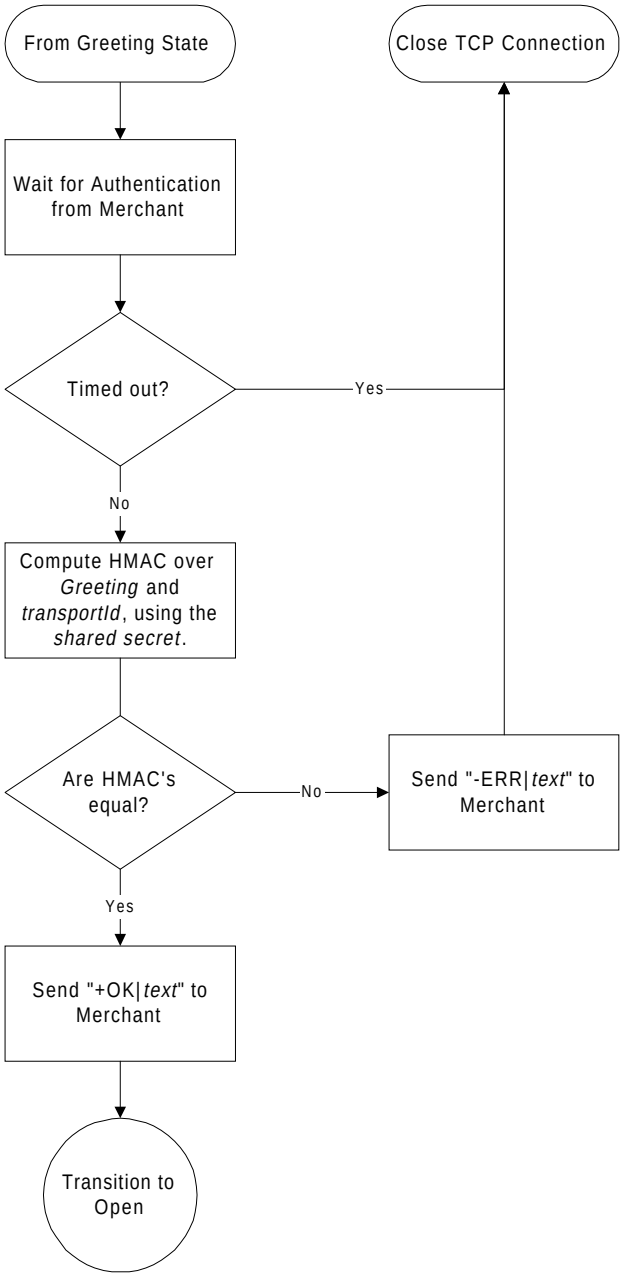

Figure 13: Payment Gateway in TCP Authenticating State

Open State

Definition

It is only in this state that the Merchant can send SET requests to the Payment Gateway. As discussed on page 69, the Merchant can interleave the requests to the Payment Gateway.

Whenever either party desires to close the communications, it issues the Close request that causes the state to transition to the *Closing* state.

All messages sent in this state are MIME-wrapped as described on page 84.

Closing State

Definition

This state allows the graceful closing of a connection between the Merchant and the Payment Gateway. This state is entered when either the Merchant or the Payment Gateway sends a Close request. After a Close request is sent (by either party), the Payment Gateway ignores any SET requests from the Merchant.

Merchant sends Close request

If the Merchant sends the Close request, it waits for the Close-Reply response (with a time-out value t_1 as described on page 73) from the Payment Gateway. When the Close-Reply response is received (or a time-out occurs), the Merchant closes the TCP connection and transitions to the *Closed* state. When the Payment Gateway receives the Close request, it responds with a Close-Reply. After sending the Close-Reply, both parties close the TCP connection. Any outstanding SET responses are lost.

Figure 14 illustrates the message flow when the Merchant sends the Close request.

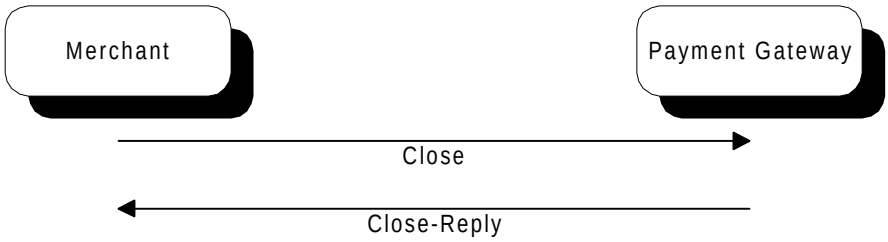

Figure 14: TCP - Merchant Sends Close Request

Continued on next page

Closing State, continued

Payment Gateway sends Close request

If the Payment Gateway sends the Close request, the Merchant can wait until it receives any outstanding SET responses before sending the Close-Reply response to the Payment Gateway. If the Payment Gateway does not receive the Close-Reply response within t_2 seconds (see page 97), the Payment Gateway closes the TCP connection and transitions to the *Closed* state.

Figure 15 illustrates the message flow when the Payment Gateway sends the Close request.

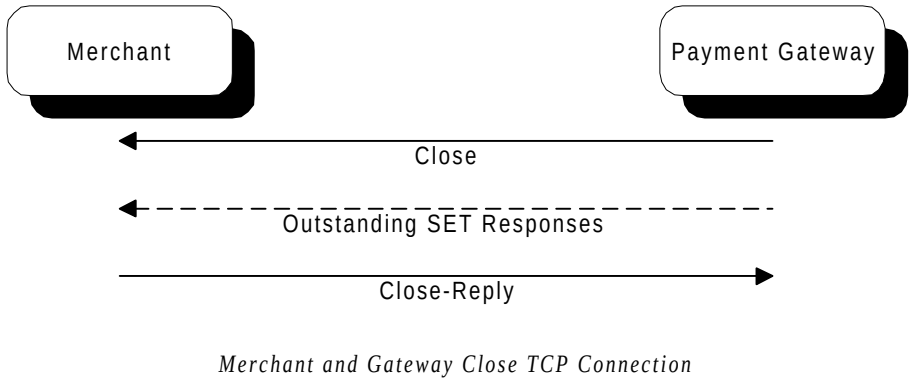

Figure 15: TCP - Payment Gateway Sends Close Request

The dotted line indicates that the Payment Gateway may or may not send any outstanding SET Responses.

Multiple Close requests

If a Close request is received while in this state, it is treated like a Close-Reply response.

MIME-wrapping

MIME header

When exchanged via this TCP method, SET messages are wrapped by a standard MIME header as follows:

```
MIME-Version: 1.0↵
Content-Type: application/set-payment;msg-tag="xxx"↵
Content-Transfer-Encoding: binary↵
Content-Length: yyy↵
↵
{DER-encoded SET message}
```

See Appendix A on page 98 for more information on MIME.

msg-tag parameter

The `msg-tag` parameter is optional, and if present in a SET request, it must be returned in the SET response. The following are reasons for using the `msg-tag`:

- When multiple merchants wish to communicate over the same link, as in an Internet Mall, and the responses need to be sent to the correct merchant.
- When it is desirable to observe messages passing over a physical link with a line monitor and easily see some plain text identification of the message.
- When the merchant wants to interleave messages, so that the responses can be matched to the request.
- When messages received at an acquirer's payment gateway communication front-end need to be efficiently dispatched to different gateway back-ends, as indicated by the merchant under prior agreement with the acquirer. This usage is outside the scope of this document.

All of these and possibly additional functions are enabled by the `msg-tag` parameter. The Merchant can add it to the **Content-Type** header in the MIME wrapper on SET requests. Its value is quoted string or token, up to 32 characters, as specified in RFC 1521. The value is specified by the Merchant and is opaque to and echoed back by the Payment Gateway. The Merchant is free to encode multiple "subfields" into the **msg-tag** value. For example the tag might be: "`merchantID.messageSequenceNo`", where "`merchantID`" identifies the merchant within a mall, and the "`messageSequenceNo`" is the number of the message so that the Merchant can correlate a response to a request.

Transport Layer Control Messages

Purpose

This TCP method provides for a number of transport layer conditions to be noted and transport layer actions to be available. Messages are needed for graceful close of the TCP connection, transport level status reporting, and a transport level test of the connection.

Transport layer conditions include:

- errors in the MIME wrapper, and
- transport-level status such as queues too full.

Structure

The general structure of these transport layer messages is as follows:

```
MIME-Version: 1.0↵
Content-Type: text/plain↵
Content-Transfer-Encoding: binary↵
Content-Length: vvv↵
Set-Transport: control=www;class=xxx;delay=yyy;msg-tag="zzz"↵
↵
{This is human readable text.}↵
```

The body of the message is intended to be human readable text, and its length is "vvv" bytes.

SET-Transport Parameters

For the `control` parameter, the only acceptable values are:

`close`	specifies a Close request message
`close-reply`	specifies a Close response message
`status`	specifies a Status message
`echo`	specifies an Echo request message
`echo-reply`	specifies an Echo response message

The **msg-tag** parameter is required for `control=status` messages, and optional for the other messages. If it is present, then it must be included in any response.

The **class** and **delay** parameters are only valid when `control=status`.

Graceful Close Message

Purpose

With multiple outstanding requests permitted on a TCP/IP connection, a graceful way to close is required. This process was described in the *Closing* state description on page 82.

Structure

The graceful close request is indicated by a `control=close` parameter and the response by a `control=close-reply`. The body of the close should indicate the reason for the close (idle-connection, equipment going out of service, etc.) and must be echoed in the response.

Sample Close request

The following is a sample Close request message:

```
MIME-version: 1.0↵
Content-Type: text/plain↵
Content-Length: 33↵
Content-Transfer-Encoding: binary↵
Set-Transport: control=close;msg-tag=“zzz”↵
↵
Equipment going out of service.↵
```

Sample Close-Reply response

The following is a sample Close-Reply message:

```
MIME-version: 1.0↵
Content-Type: text/plain↵
Content-Length: 33↵
Content-Transfer-Encoding: binary↵
Set-Transport: control=close-reply;msg-tag=“zzz”↵
↵
Equipment going out of service.↵
```

Status Messages

Overview

Status messages are indicated by a `control=status` parameter. There is no response to a status message.

Note: These are transport layer only messages. Any SET level errors are indicated through the SET **Error** and Response messages.

Failed message

If `class=failed`, a permanent error has occurred in the sense that the message causing the error should not be retried. The body contains an explanation. For example: “Messages too big”, “Service not available at this time”, “Bad content type”, “Bad content-transfer-encoding”, “Connection closing”, etc.

The following is a sample Failed message:

```
MIME-version: 1.0↵
Content-Type: text/plain↵
Content-Length: 25↵
Content-Transfer-Encoding: binary↵
Set-Transport: control=status;class=failed;msg-tag=“xxx”↵
↵
Service is unavailable.↵
```

Retry message

If `class=retry`, a transient error has occurred (such as “Busy”, “Too many outstanding requests”, etc.) and the message causing the error may be retried in not less than *n* number of seconds as specified by `delay=`*n*, where *n* is a number greater than zero.

The following is a sample Retry message:

```
MIME-version: 1.0↵
Content-Type: text/plain↵
Content-Length: 32↵
Content-Transfer-Encoding: binary↵
Set-Transport: control=status;class=retry;delay=5;msg-tag=“xxx”↵
↵
Too many outstanding requests!↵
```

Continued on next page

Status Messages, continued

Info message

If `class=info`, the message has been accepted and some status concerning it is being reported. The only anticipated use of this is to inform the sender that a response may be unusually delayed. It is possible to receive multiple such status messages for one request (identified by the `msg-tag`) and possible to receive one or more such status messages followed by a transport status message.

The following is a sample Info message:

```
MIME-version: 1.0↵
Content-Type: text/plain↵
Content-Length: 25↵
Content-Transfer-Encoding: binary↵
Set-Transport: control=status;class=info;msg-tag="xxx"↵
↵
Slow financial network.↵
```

Closing message

If `class=closing`, the message has not been accepted because the connection is closing. The Merchant should resend this message in another connection.

The following is a sample Closing message:

```
MIME-version: 1.0↵
Content-Type: text/plain↵
Content-Length: 24↵
Content-Transfer-Encoding: binary↵
Set-Transport: control=status;class=closing;msg-tag="xxx"↵
↵
Connection is closing.↵
```

Echo Messages

Purpose

Echo messages may be used to test transport layer connectivity, and can be issued by either party.

Structure

The request is indicated by `control=echo` and the response by `control=echo-reply`. The body of the echo message must be echoed as the body of the echo-reply.

Sample Echo request

The following is a sample Echo request message:

```
MIME-version: 1.0↵
Content-Type: text/plain↵
Content-Length: 16↵
Content-Transfer-Encoding: binary↵
Set-Transport: control=echo↵
↵
Are you there?↵
```

Sample Echo response

The following is a sample Echo response message:

```
MIME-version: 1.0↵
Content-Type: text/plain↵
Content-Length: 16↵
Content-Transfer-Encoding: binary↵
Set-Transport: control=echo-reply↵
↵
Are you there?↵
```

Non-SET Messages

Non-SET message summary

Table 22 summarizes the non-SET messages used in this TCP communications method.

Message Name	Allowed in States	Description	MIME-wrapped
Greeting	Greeting	Sent by Payment Gateway to accept connection or challenge Merchant.	No
Authentication	Authenticating	Sent by Merchant to respond to the Payment Gateway's challenge.	No
Authentication-Reply	Authenticating	Sent by the Payment Gateway to accept or deny the Merchant's challenge.	No
Close	Open	Sent by either party to request a graceful closing of the connection.	Yes
Close-Reply	Closing	Sent by the party receiving the Close request to acknowledge the Close.	Yes
Status	Open, Closing	Sent by the Payment Gateway to inform the Merchant of the status of the previous message.	Yes
Echo	Open, Closing	Sent by either party to request confirmation that the connection is still active.	Yes
Echo-Reply	Open, Closing	Sent by the party receiving the Echo request to acknowledge the Echo.	Yes

Table 22: Non-SET Messages for TCP

Diagnostic Log

Use of Error messages

SET **Error** messages are used to indicate that a message cannot be reliably identified and processed.

Method

When diagnostic log messages are sent from Merchant to Payment Gateway via TCP, the following method is specified:

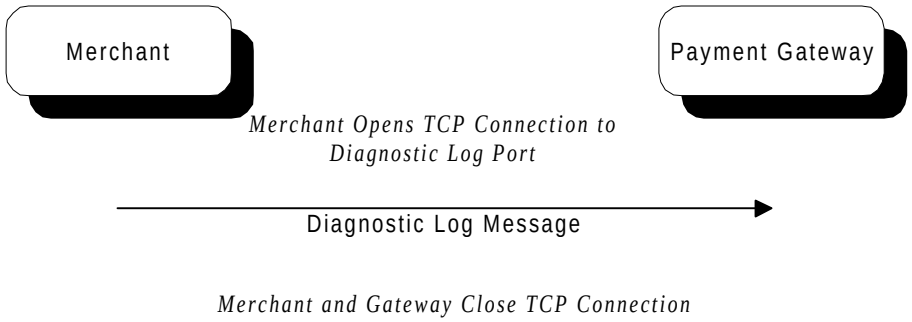

Figure 16: TCP Diagnostic Log

When a Merchant receives a SET response message that fails basic SET message decoding (including DER-decoding errors and failure of signature verification), it:

- generates a SET **Error** message,
- opens a TCP connection to the Payment Gateway's diagnostic log port,
- sends the MIME-encapsulated DER-encoded SET **Error** message, and
- closes the TCP connection.

Note: A Merchant must send only one diagnostic log message per TCP connection.

When the Payment Gateway receives a diagnostic log message:

- It does not respond, but simply closes the TCP connection once the message is received.
- Processing of the diagnostic log message is implementation dependent. It is strongly recommend that Payment Gateways journal such messages in log files.

Example Communication

Overview

This section provides examples of Merchant to Payment Gateway communications, including:

- Single SET Request/Response Pair
- Multiple Request/Response Pairs
- Merchant Sends Close Request
- Payment Gateway Sends Close Request
- Transport Errors Occur During Processing

Single SET request/ response pair

Figure 17 shows a single SET request/response pair with authentication and no errors.

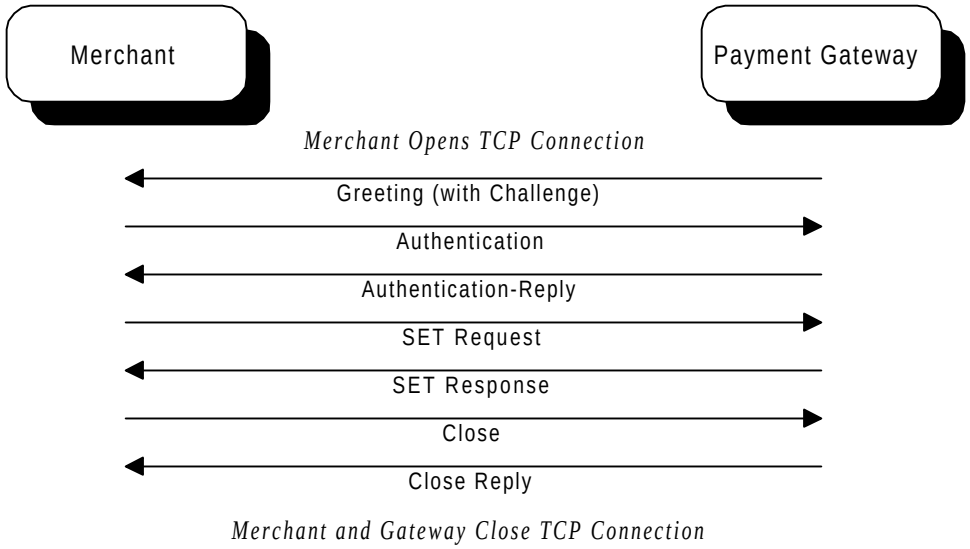

Figure 17: TCP - Single SET Request/Response Pair

If the Merchant does not want to interleave transactions, it may still send multiple requests on a single connection in serial mode, waiting for each response before sending the next request. See Figure 18 on page 93.

Multiple simultaneous connections, up to the agreed maximum (see page 97), may be in progress between the Merchant and the Payment Gateway.

Continued on next page

Example Communication, continued

Multiple request/ response pairs

Figure 18 shows multiple request/response pairs with authentication and no errors.

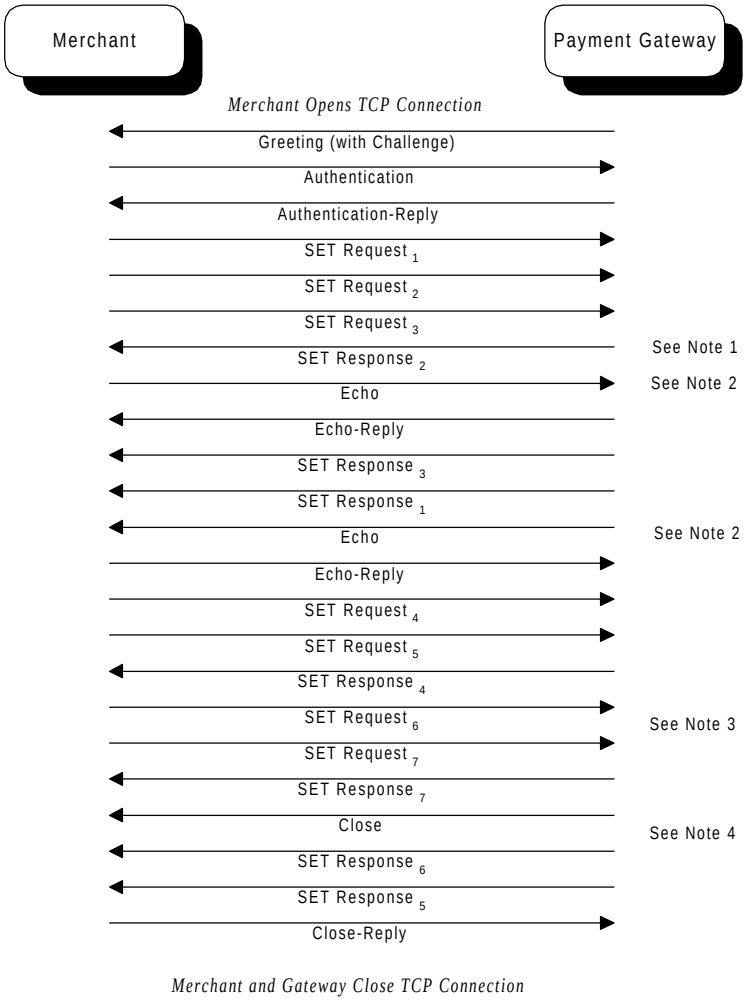

Figure 18: TCP - Multiple SET Request/Response Pairs

Notes:

1. Responses to SET requests may come back in any order.
2. An Echo request may be sent by either end of a connection while the connection is in the Open or Closing states.
3. The Payment Gateway may send a Close request while responses are outstanding.
4. The Merchant should send the Close-Reply only after all outstanding SET responses are received or the connection has timed-out.
5. Since TCP is full-duplex, responses to SET requests may be returned at any time.

Continued on next page

Example Communication, continued

Merchant sends the Close request

Figure 19 outlines the flow when the Merchant sends a Close request.

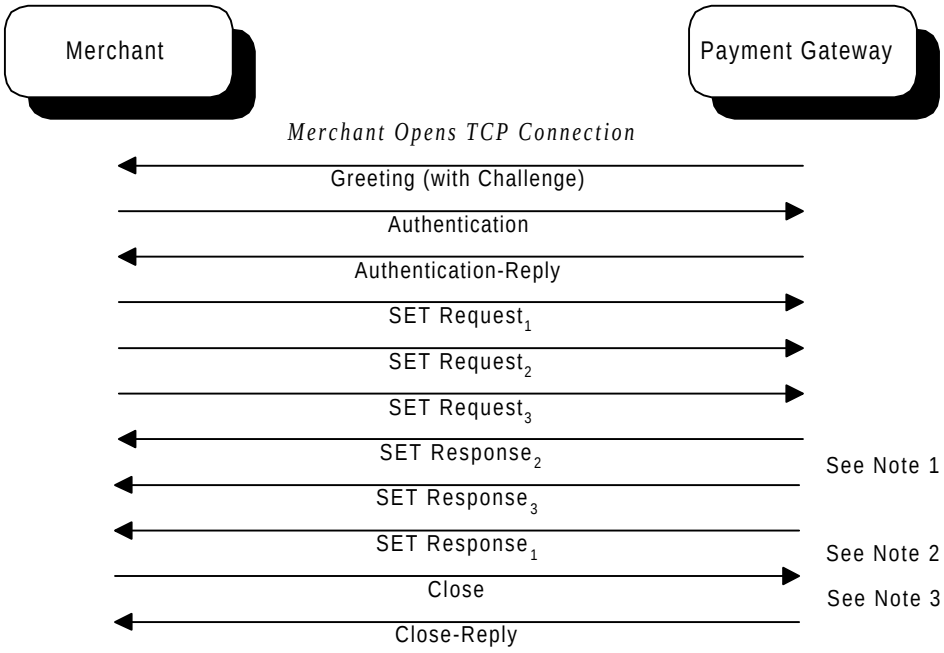

Figure 19: TCP - Merchant Sends Close Request

Notes:

1. Merchant decides to close the connection.
2. Merchant waits for all outstanding responses.
3. Merchant sends Close request. The Merchant must not send the Close request when SET responses are outstanding unless it is willing to lose the responses. Once the Payment Gateway receives the Close request, it will discard any outstanding responses.

Continued on next page

Example Communication, continued

Merchant sends request after Close request

Figure 20 outlines the flow when the Payment Gateway sends a Close request, and the Merchant sends a subsequent request.

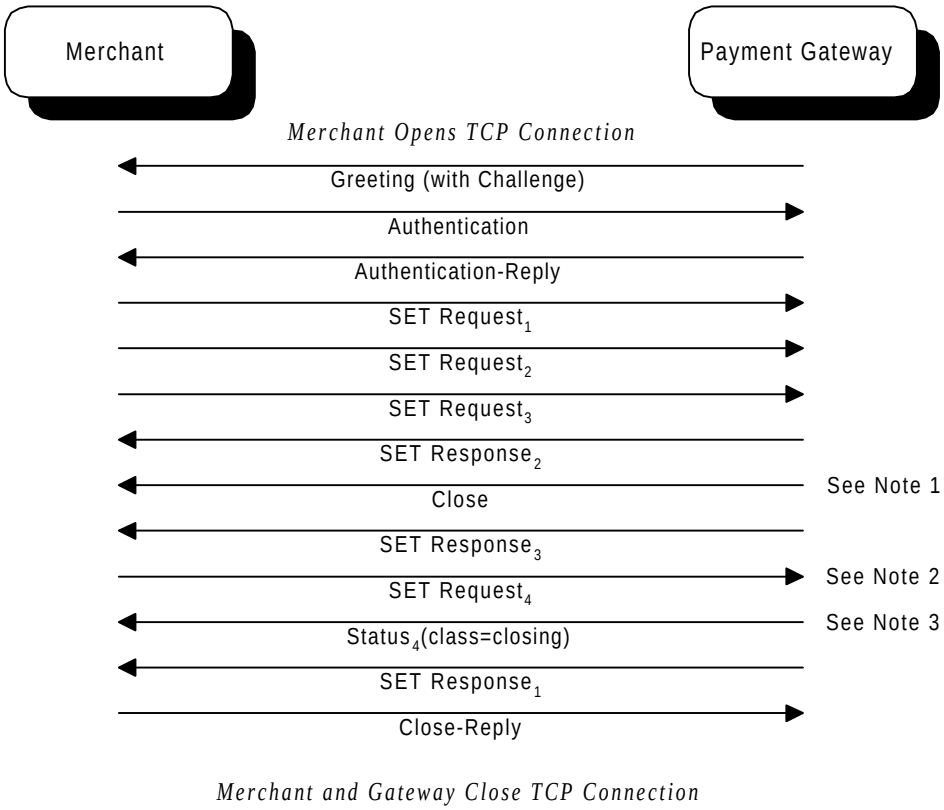

Figure 20: TCP - Payment Gateway Sends Close Request

Notes:

1. Payment Gateway sends a Close request.
2. Merchant sends another SET request. This may be unintentional due to IP stack or transmission delays at either end of the connection.
3. Payment Gateway sends a Status with a `class=Closing`, indicating that the message should not be resent in this connection. Merchant may resend the request on another connection.

Continued on next page

Example Communication, continued

Transport errors

Figure 21 demonstrates processing during which transport errors occur.

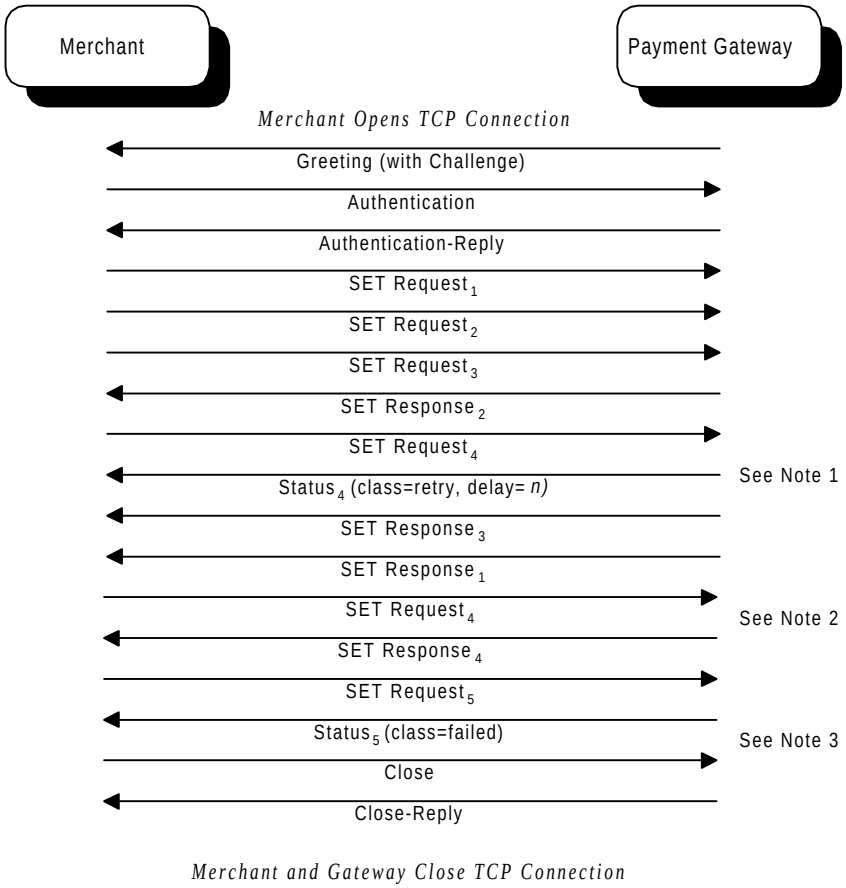

Figure 21: TCP - Transport Errors Occur During Processing

Notes:

1. The Payment Gateway received a request that could not be handled at this time. It sent the Merchant a Status message indicating that the request should be resent in *n* seconds. *N* is defined in the *delay* field of the MIME message header.
2. The Merchant waits *n* seconds before resending this request. It should not be assumed that no request may be sent until the delay expires as the Payment Gateway may be having a problem with specific requests.
3. A Status message with "`class=failed`" indicates that the previous message should not be resent on this or any other connection.
4. Neither party should ever send a Status message as a response to a received Status message.

Out-of-band Merchant/Acquirer Agreement

Items to agree

The Merchant and acquirer must agree on several networking attributes prior to TCP communications. This information must be exchanged out-of-band. A partial list of these attributes, with an explanation, is provided below:

Port number and one or more DNS names (or IP addresses) for Payment Gateway	Each participant requires sufficient information about the other participant to open a connection for communications. See page 73.
One or more DNS names (or IP addresses) for Merchant	The default payment gateway TCP port number is 257, as assigned by IANA. Gateways may provide a feature where, by mutual agreement with a merchant, they listen on a different port, but are not required to provide such a feature.
TransportID name	The identity of the message originator (for example, the name of the originating host). This field is used in the Authentication message. See page 77.
Maximum number of outstanding requests per TCP connection	The maximum number of pending request(s) in the participant's network queue to be processed. See "Interleaving" on page 69.
Minimum time-out on retry instance requests	The minimum time the requesting entity should wait before resubmitting SET messages when the TCP connection is prematurely broken or no response is received.
Maximum number of automatic retries per request	The maximum number of retries a participant can send per SET message when the TCP connection is prematurely broken or no response is received.
Maximum time-out for value t_1	The maximum time either party should wait while in the Greeting and Authenticating states. See pages 73 and 77. A reasonable value for t_1 is 30 seconds.
Maximum time-out for value t_2	The maximum time either party should wait while in the Closing state. See page 82. A reasonable value for t_2 is 120 seconds.
Maximum number of concurrent TCP connections	The maximum number of concurrent TCP connections that are available to each participant.
Authentication requirement	Whether authentication is necessary between the communicating parties. See page 70.
Shared secret value	The key that the HMAC function will use for generating the challenge/response. See page 77.

Table 23: TCP Networking Attributes Defined Out-of-band to SET

Appendix A
An Overview of MIME

Introduction

Purpose

This appendix specifies how the MIME (Multipurpose Internet Mail Extensions) standard is used in either of two cases:

- to wrap a SET message for those transport mechanisms that require MIME; or
- to create the SET initiation messages (as described on page 7).

The MIME standard is specified in RFCs 2045 through 2049. The reader is encouraged to supplement this overview with the information provided in these RFCs , and in RFC 822.

RFC 822

The MIME standard specifies a mechanism that extends RFC 822, *Standard For The Format Of ARPA Internet Text Messages*. RFC 822 "specifies a syntax for text messages that are sent among computer users." An RFC 822 message consists of "an envelope and contents. The envelope contains whatever information is needed to accomplish transmission and delivery. The contents compose the object to be delivered to the recipient." RFC 822 "applies only to the format and some of the semantics of message contents. It contains no specification of the information in the envelope."

Extending RFC 822

RFC 822 defines a message representation protocol specifying considerable detail about US-ASCII message headers, and leaves the message content, or message body, as flat US-ASCII text. MIME extends the RFC 822 format of messages to allow for:

- textual message bodies in character sets other than US-ASCII,
- an extensible set of different formats for non-textual message bodies,
- multipart message bodies, and
- textual header information in character sets other than US-ASCII.

Continued on next page

Introduction, continued

Character sets

MIME allows multiple character sets to be used in the MIME header and body. To use a character set other than US-ASCII in a field, the application must use the *encoded-word* mechanism specified in RFC 2047.

Message format

An RFC 822/MIME message consists of:

header	Contains information about the body
blank line	This line separates the header from the body; it is composed of the US-ASCII characters CR (0x0D) immediately followed by LF (0x0A). This blank line need not be present if the body is empty.
body	The actual message, which may be empty.

Table 24: RFC 822 / MIME Message Format

Message Header

Header fields

The message header consists of a series of fields. Certain fields are required for MIME compliance and other fields are application defined or RFC 822 defined. The general format for a field is:

```
field-name: field-value↵
```

field-name	identifies the field, and is composed of the printable US-ASCII characters (0x21 - 0x7E) excluding colon (":").
field-value	contains the value of the field, and is composed of any US-ASCII character with the exception of CR (0x0D) or LF (0x0A). The RFC 2047 *encoded-word* mechanism can be employed to utilize other character sets.

Long lines

RFC 822 allows long lines to be "folded" into the next line. If a line begins with either US-ASCII Space (0x20) or Tab (0x09), it is considered part of the previous line. Refer to RFC 822 for details about folding and unfolding.

Case sensitivity

Field-names are not case sensitive. For example, "MIME", "Mime", and "mime" would all be treated as the same field-name.

The case sensitivity for the field-value is application-dependent. In SET, all field-bodies are not case sensitive unless otherwise specified.

Comments

RFC 822 allows comments in the header fields. Comments are not allowed within SET-defined fields; that is, any parentheses and the data contained between them are considered part of the value of the field.

MIME-Version

The **MIME-Version** header field specifies which version of MIME is being used. At the time of this writing, only version 1.0 of MIME has been defined.

In SET, each MIME header must contain the following header field:

```
MIME-Version: 1.0↵
```

Continued on next page

Message Header, continued

Content-*

A header that begins with "**Content-**" is considered a "MIME" header. It must describe an attribute of the message body.

Content-Type

This field specifies the general category (or type) of the data contained in the message body (for example, text, audio, image, video, or application). A sub-type can also be given, separated from the main type by a "/". Typically, the sub-types specify format of the data. In the case of the *application* type, the sub-type commonly specifies a particular application for the described data.

In SET, each MIME header must contain the following header field:

```
Content-Type: type/sub-type↵
```

Continued on next page

Message Header, continued

Content-Transfer-Encoding

This field specifies the encoding used to prepare the data for transmission so that it may be decoded upon arrival. Transfer encoding is particularly important for transmitting raw binary data via electronic mail. Many older mail gateways handle only plain text data properly, garbling any binary data that passes through them. The transfer encoding field allows binary data to be encoded for transmission through even the most limited mail gateways.

In SET messages, each MIME header must contain the following header field:

```
Content-Transfer-Encoding: encoding↵
```

where “encoding” is “binary”.

Content-Length

This field specifies the length in bytes (octets) of the body of the MIME message.

In SET, each MIME header may contain the following header field:

```
Content-Length: len↵
```

where “len” is the length of the body, counted from the first byte after the CRLF that marks the blank line ending the header.

MIME Body

Body

The MIME body carries the actual message being transported.

The MIME body is separated from the MIME header by a blank line. This line consists of the US-ASCII CRLF (0x0D 0x0A) pair. This blank line is required only if the message has a non-empty body.

Appendix B
World Wide Web Operation

Overview

Introduction

World Wide Web (WWW) service is delivered via the HTTP protocol. HTTP is intended for rapid, interactive applications. It also allows client browsers to request information from and post data to a network-based server. During an HTTP transfer, MIME headers are transmitted along with other headers following the initial request/response. This happens before any requested data is transferred. HTTP always supports the transfer of binary data.

Purpose

This appendix discusses issues related to Web operation and provides a high-level overview of what happens when a user browses a merchant's Web site, decides to buy something, and wants to pay using SET.

For details about using HTTP to transport SET messages, see "HTTP-based Transport" on page 54.

Issues

Overview

Several issues must be addressed before implementing WWW support for SET transactions:

- Initiation
- Browser upstream support
- Browser manipulation
- Application state
- Session time-outs
- Retry limits
- Browser caching
- Diagnostic logging

Initiation and exchange of non-SET data

Because of the mechanics of HTTP, it is most convenient to start the SET helper application in response to data sent by the Merchant or CCA. By design, SET does not allow payment transactions to be initiated by the Merchant.

Also, the Merchant and the Cardholder need to agree on the OD and the purchase amount prior to the initial exchange. There is no straightforward method for supporting this exchange using SET exclusively.

It is expected that these issues will eventually be resolved through the development of shopping protocols. Meanwhile, a method for exchanging the OD and the amount of the purchase is described in “Payment-Initiation Message Header” starting on page 35. This mechanism also provides a work-around for the initiation problem.

Continued on next page

Issues, continued

Browser upstream support

Browser MIME support provides a channel from the Merchant, down through the browser, to the SET payment application. It does not provide a similar upstream channel; that is, there is not a pre-defined method for passing information back from the SET application to the Merchant via the user's browser.

It is expected that this issue will eventually be resolved through internal browser support for SET transactions.

Meanwhile, since there is no standardized channel for passing data upstream from helper applications through browsers, the upstream problem will have to be handled in a platform/browser dependent fashion. Methods may include operating system supported inter-process communication, manually driven file-based interaction, or browser-dependent client APIs.

If there is no channel for passing data upstream through the browser, the SET helper application may be required to initiate connections to the **SET-SET-URL** and **SET-Query-URL**. As HTTP is expected to be a common scheme for such URLs, the SET application may need to initiate HTTP POST requests for SET messages and receive the corresponding HTTP responses. As an HTTP originator, it may also need to deal with firewall-related issues such as proxy support.

Browser manipulation

The initiation mechanism described in this document provides for three URLs (**SET-Success-URL**, **SET-Failure-URL**, and **SET-Cancel-URL**), which the Cardholder software is expected to retrieve for a smooth transition to the appropriate merchant-controlled WWW page at the end of a payment transaction. For the smoothest transition, the contents of such URLs replace the contents of the merchant page that triggered the payment request. However, there is no standardized mechanism for a helper application either to identify the appropriate instance of a browser or to cause the browser to retrieve a particular page.

Therefore, the browser manipulation issue must be handled in a platform- and/or browser-dependent fashion. For example, Windows-based browsers may provide OLE automation interfaces that enable retrieval of URLs.

If there is no feasible mechanism for the SET "helper application" to cause the browser to retrieve URLs, the helper application may not be able to support the transition to the **SET-Success-URL**, **SET-Failure-URL**, or **SET-Cancel-URL**. The helper application is not expected to provide HTML support itself in order to retrieve the transition URLs and display the associated Web pages on its own.

Continued on next page

Issues, continued

Application state

In some environments, it will not be possible to keep the SET helper application live during the entire SET transaction "lifetime." Therefore, implementations may need to save state between invocations. It may be possible in some environments to request that the "host" application provide caching for this state. In other environments, the helper application will be required to implement all cache management. The solution to caching state will be implementation- and application-specific.

Due to the unspecified nature of the end of a transaction, deletion of cache state may require user intervention. For instance, on the client side, the purchase response (**PRes**) message may not signal the end of the transaction because it is still possible for the user to request a payment inquiry (**InqReq**).

Session time-outs

Under a number of circumstances, SET software operating over HTTP connections may not receive a timely response: The upstream system may be down; a server may be overloaded; an intervening network connection may be down.

For whatever reason, the upstream system in a SET transaction may fail to respond in a timely manner.

In the case of the Payment Gateway, the upstream system is the financial network. Over these connections, the default time-out value is thirty seconds.

For connections other than Payment Gateway to financial network:

- Time-out response is less critical. Since all necessary SET messages are idempotent, SET software may simply resend request messages until a successful response is received. Duplicate requests are simply ignored.
- Time-out policy (that is, time until time-out and response in case of time-out) should be configurable on a per connection basis.

Continued on next page

Issues, continued

Retry limits

Unconstrained use of message retries can result in an excessive burden on SET servers. This may aggravate whatever problems necessitated message retry to begin with. In order to avoid this, it is recommended that SET software institute reasonable limitations on message retries on outbound messages.

Browser caching

Many proxy servers cache HTTP pages between sessions. Despite all best efforts, this cached data may occasionally be displayed to web users for which it was not intended. For more information, see “Browser caching” on page 56.

Diagnostic logging

It may be desirable to support a logging mechanism outside the normal SET channel to avoid error message loops. Unfortunately, firewalls, which often allow HTTP connections, do not usually allow other types of TCP connections – therefore error logging via HTTP may be necessary.

In the Cardholder–to–Merchant link, any SET **Error** messages should be sent by the Cardholder to the **SET-Diagnostic-URL**.

In the Merchant–to–Payment Gateway link, if logging is supported on HTTP connections, Merchant software should use a unique URL, negotiated out-of-band, for SET **Error** messages.

WWW Interaction

Typical interaction

A typical WWW payment transaction might proceed as follows:

Step	Action
1	Customer shops at merchant web site, selects goods, negotiates price, and then selects SET as the payment method.
2	Merchant's server sends to the customer's browser the SET payment initiation message with MIME **Content-Type** *application/set-payment-initiation*. This message contains the OD and amount of the payment.
3	The customer's browser starts a SET helper application.
4	The SET application presents the OD and amount to the user for verification.
5	If the user disapproves the OD or amount, the SET application causes the browser to transition to the WWW page given by the **SET-Cancel-URL**, then stops.
6	If the user approves, the SET application: • authenticates the customer, and • allows him or her to select a payment card account from the electronic wallet.
7	The SET application: • records the initial transaction data for later use, • creates a **PInitReq** message, and • passes the **PInitReq** message to the browser via a platform/browser-dependent channel.
8	The browser posts the message to the **SET-SET-URL** address with the MIME **Content-Type** *application/set-payment.*
9	The Merchant application: • receives and processes the posted message, • creates an appropriate **PInitRes**, and • passes the **PInitRes** back to the customer via the server as a response to the customer's post, using the MIME **Content-Type** *application/set-payment.*

Table 25: Typical SET Interaction via the WWW

Continued on next page

WWW Interaction, continued

Typical interaction (continued)

Step	Action
10	The browser passes the **PInitRes** message to the SET application.
11	The SET application extracts the merchant's name and address from the merchant's certificate, shows them to the user, and asks for approval.
12	If the user approves, the SET application: • uses the OD and amount and other information to form the **PReq** message, • updates its records, and • posts the **PReq** to an application at the merchant server via the customer's web browser.
13	The Merchant's software: • processes the **PReq**, • forms an appropriate **PRes**, and • returns it to the customer via the merchant's web server using the MIME **Content-type** *application/set-payment*.
14	The customer's browser passes the **PRes** to the SET application.
15	The SET application parses the messages and displays the results.
16	If an error occurred in the processing of the SET protocol, the SET application causes the browser to transition to the WWW page given by the **SET-Failure-URL**. Otherwise, it causes the browser to retrieve the page given by the **SET-Success-URL**.

Table 25: Typical SET Interaction via the WWW, continued

Initiation Retry

An alternative interaction may occur if the Merchant provides a **SET-Response-URL** in the *payment-initiation* message at step 2. The **SET-Response-URL** indicates that the Merchant is prepared to accept a *payment-initiation-response* message from the Cardholder, as described in "Initiation Response Message" on page 45. At any of steps 3 to 7, the Cardholder software may send a *payment-initiation-response* to the Merchant. One reason (of several) might be to solicit an alternate format for the OD.

If the Merchant receives a *payment-initiation-response* message for which another *payment-initiation* is warranted, the Merchant goes back to step 2 and sends another *payment-initiation* message based on the information provided in the *payment-initiation-response*. Another message may be warranted if, for example, the response indicated that an alternate character set should be used.