

INTERNET-DRAFT

Donald E. Eastlake 3rd
CyberCash
29 January 1996

Expires: 28 July 1996

Application Level Internet Payment Syntax

Status of This Document

This draft, file name [draft-eastlake-internet-payment-01.txt](#), is intended to become one or more Proposed Standard RFCs. Distribution of this document is unlimited. Comments should be sent to the author <dee@cybercash.com> or the ietf-payments mailing list <ietf-pay@imc.org>.

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months. Internet-Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet-Drafts as reference material or to cite them other than as a ``working draft'' or ``work in progress.''

To learn the current status of any Internet-Draft, please check the `1id-abstracts.txt` listing contained in the Internet-Drafts Shadow Directories on `ds.internic.net` (East USA), `ftp.isi.edu` (West USA), `nic.nordu.net` (North Europe), `ftp.nis.garr.it` (South Europe), `munari.oz.au` (Pacific Rim), or `ftp.is.co.za` (Africa).

INTERNET-DRAFT

Internet Payment Framework

29 January 1996

Abstract

The Internet is becoming an increasingly commercial arena in which information is being bought and sold and payments are rendered for services and data delivered within the Internet and goods and services to be delivered outside of the Internet. Thus far, the protocols and format used for such payments have been ad hoc and proprietary.

This draft proposes a uniform application level syntax to support commerce in applications level data and services. Proposed specifications are given for how this syntax fits into and enables commerce in the data and services rendered by the World Wide Web, FTP, Telnet, SMTP, and DNS protocols.

Acknowledgments

The contributions of the following persons to this draft are gratefully acknowledged:

Brian Boesch <boesch@cybercash.com>
Phillip Hallam-Baker <hallam@w3.org>
Dave Kristol <dmk@alleggra.att.com>
David S. Raggett <dsg@w3.org>.

INTERNET-DRAFT

Internet Payment Framework

29 January 1996

Table of Contents

Status of This Document.....	1
Abstract.....	2
Acknowledgments.....	2
Table of Contents.....	3
1 . Introductions.....	4
1.1 Applications Level Applicability.....	4
1.2 Overview of this document.....	4
2 . Price Tags.....	6
2.1 Prices.....	6
2.2 Payment System Strings.....	6
2.3 Price Tags.....	7
3 . Payments, Receipts, and Errors.....	9
3.1 Payment Strings.....	9
3.2 Receipt Strings.....	9
3.3 Message Flow and Payer Private Information.....	10
3.4 Refunds.....	10
3.5 Errors.....	10
4 . Use in the World Wide Web.....	11
4.1 Web Browser User Interface.....	11
4.2 Anchor Embedded Costs.....	11
4.3 Page Header Price Tags.....	12
4.4 HTML Form Price Tags.....	13
4.5 Payments and Receipts Via Miscellaneous Headers.....	14

4.6	Payments and Receipts Via PEP.....	15
4.7	Web Proxies.....	16
5.	Use in File Transfer Protocol.....	18
6.	Use in Telnet.....	19
7.	Use in Simple Message Transfer Protocol.....	20
8.	The Domain Name System.....	22
9.	Protocols to Which Payment is not Applicable.....	24
9.1	The ECHO, DISCARD, and CHARGEN Services.....	24
9.2	The Finger Service.....	24
9.3	The Auth Service.....	25
10.	Security Considerations.....	26
	References.....	27
	Author's Address.....	27

[1.](#) Introductions

Commerce in applications level data and services on the Internet requires a means to (1) indicate prices and acceptable methods of payment, (2) tender payment, and (3) issue a receipt acknowledging payment or notification if payment fails.

This document specifies a character string syntax for these three items.

[1.1](#) Applications Level Applicability

Payment facilities could be applied at a number of levels. This specification is concerned only with applications level provision of data items and services. It does not concern itself with network level packets or quality of service nor does it concern itself directly with transport level connections or quantity or quality of service except as these transport level measures impact application services.

This proposed syntax is concerned with such matters as payment for

access to a web page, upload of a file via ftp, initiation of a telnet session, or conducting an extensive WAIS search. These are generally user visible and meaningful data objects or tasks.

Within most legal systems, the owners of such data objects and/or the owners of the facilities used to present such objects or perform such tasks are frequently entitled to require some recompense if they choose to require it. This document does not concern itself with the morality of such laws or requirements but merely provides a syntax whereby cooperating entities may speak at that level about prices and payments.

There is no requirement that the "currencies" used with this syntax be the usually recognized national or international currencies. For example, some transactions could be denominated in frequent flyer miles or other private unit.

[1.2](#) Overview of this document

Sections [2](#) and [3](#) below define a basic syntactic framework for price tags, payments, and receipts.

Sections [4](#) through [8](#) specify a standard for inclusion of these items in transactions for the World Wide Web (HTTP/HTML), File Transfer Protocol (FTP), Telnet, Simple Message Transfer Protocol (SMTP), and

the Domain Name System (DNS) protocols.

[Section 9](#) lists some protocols to which application level payment systems should not be applied.

[Section 10](#) discusses security considerations.

[Appendix A](#) is an initial list of payment systems that are or are planned to be usable via this syntax.

[Appendix B](#) gives a semi-formal BNF-like description of the syntax.

[2.](#) Price Tags

A uniform price tag format is needed to indicate when payment is due, how much, and what payment methods are acceptable by the seller. Such a price tag must include the specification of one or more acceptable payment systems (with a provision for payment system specific information) and will commonly include one or more prices.

Sections [2.1](#) and [2.2](#) below describe prices and payment system strings and [Section 2.3](#) assembles these for complete price tags.

[2.1](#) Prices

Prices are encoded as character strings consisting of a number followed by a currency code.

These currency codes are the three letter ISO 4217 codes, Internet Assigned Number Authority (IANA) registered four to eight letter currency codes, or any valid domain name containing at least two labels. (ISO 4217 codes normally consist of the two letter country code followed by a letter mnemonic for the major unit of currency.) Currency codes are case insensitive. Codes of one, two, or more than eight letters appearing in the place of a currency code are reserved for future definition.

The number preceding the currency designation is the quantity of major units of that currency. It may optionally have a decimal point and additional decimal fraction digits and may optionally have a "+" or "-" immediately followed by an integer scaling exponent. Any number of digits of precision and any size exponent may be provided but payment systems may define how many digits and what range they utilize.

Some examples:

2.34gbp	2 pounds and 34 pence sterling
79+0ALL	seventy nine Albanian Leks
123456-5cad	one dollar 23 and 456 thousandths cents Canadian
0.125usd	one eighth of a US dollar

[2.2](#) Payment System Strings

Payment system strings consist of the payment system name, an equal sign, and any payment system specific information (such as what account within that payment system the payment should be made payable to).

Payment system names are either case insensitive four to twelve letter names registered with IANA or a URL [[RFC 1738](#)] and are terminated by an equal sign. Codes occurring in the place of payment system names consisting of one to three letter or more than twelve letters are reserved for future definition.

Payment system specific information must be encoded so that it contains no internal spaces or unusual characters as described in [Appendix B](#). It is up to the named payment system to encode and decode any information it requires so as to fit within this syntax. Use of the base64 encoding defined in [RFC 1521](#) is recommended. The payment system specific information, if any, appears immediately after the payment system name and equal sign and is terminated by white space or the end of the price tag character string.

A registry of payment system names is maintained by IANA. Initial payment system names are listed in [Appendix A](#). For experimental or private use, any URL may be used.

[2.3](#) Price Tags

A complete price tag consists of a string of white space separated prices and payment system strings. There must be at least one payment system string present.

Normally there will also be at least one price. However, there are circumstances under which the cost of a service is highly unpredictable and the seller is, in effect, requesting an arbitrary donation or a payment system and account to which they can attempt to charge indefinite amounts or payment for a service which will vary with the amount of the payment. Where meaningful, it is recommended that a price be listed that is a reasonable ceiling such that if costs exceed it, the seller will have to present another price tag; however, it is permitted to omit the price and list only a payment system in a price tag.

Payment systems SHOULD provide a means for a limited amount of arbitrary seller information to be included in the payment system specific part of a price tag and be returned to the seller within a payment message based on that price tag.

A price appearing after a payment system string applies only to that system. Putting a price before the first payment system specific information makes that price a default for every payment system specified. The default can be overridden by specifying a different amount for that currency after a particular payment system.

For example:

INTERNET-DRAFT

Internet Payment Framework

29 January 1996

33.45all foocash=xxxx 22eTb barsys=yyyy 9.999ghC

indicates that payment of twenty-two Ethiopian Birrs via the foocash payment system or 9.999 Ghanaian Cedis via the barsys system or 33.45 Albania Leks via either system is acceptable.

In cases where the cost of the service is not known in advance, the price can be an estimate, deposit request, or the like, with any overpayment refunded. Underpayment can be fixed by collecting an additional payment from the client. In the absence of trust between the parties, frequent small payments may be required.

INTERNET-DRAFT

Internet Payment Framework

29 January 1996

[3.](#) Payments, Receipts, and Errors

The sections below describe encoding of payments and receipts and the inclusion of error messages in the payment or receipt context.

A null payment or receipt string is explicitly permitted in most contexts as a way for an entity to indicate merely that it is payment syntax aware.

One or more equal sign terminated payment system names in isolation are permitted in a payment or receipt context but only as a way to indicate that a particular payment system is understood. Any actual payment or receipt must have a non-null payment specific information. Only one such full payment system string can occur in a payment or receipt.

The content and/or encoding of the payment system specific information would normally differ between the price tag, payment, and receipt contexts but this is a matter only of concern to the payment system.

[3.1](#) Payment Strings

After encountering a price tag, either initially, during a session, or in conjunction with a "payment required" error, an application needs some method of tendering payment. This is done with a payment system string with the same syntax as described in [Section 2.2](#) above. For example:

```
foocash=29Uso+0a/e92micHd4s3
```

The payment system used in the payment is selected from among those

in the price tag since those are known to be supported by the seller. Payment systems will commonly include in the payment system specific information some sort of serial or transaction number so that retransmission of a message containing the string will not result in duplicate payment.

[3.2](#) Receipt Strings

Normally the seller will provide a receipt for the amount of money actually collected or a message indicating payment failure or error. This will be via a receipt character string which is also simply in the form of a payment system string. For example:

```
barsys=8n7VtC2+uL341/==
```

Eastlake

[Page 9]

INTERNET-DRAFT

Internet Payment Framework

29 January 1996

Payment systems will commonly include, in the payment system specific information of a receipt, an indication of how much the receipt is for and some type of serial or transaction number so that retransmission of a message containing the receipt will not result in confusion.

[3.3](#) Message Flow and Payer Private Information

For some payment systems, the buyer, instead of sending a payment to the seller, actually completes payment based on the price tag and simply sends to the seller a receipt string, as described above, proving that they have paid.

For payment systems where the payment is sent to the seller, provision SHOULD be made for a small amount of arbitrary payer private information to be provided in the payment message by the buyer and returned to the buyer by the seller in the receipt.

[3.4](#) Refunds

Depending on payment system details, refunds may not be available or they can be implemented in two ways. It can be a payment message

from the seller to the buyer, normally leading to a receipt from the buyer to the seller. Or the seller may be able to directly refund to the buyer's account or the like and simply send the buyer a receipt. In some payment systems, both refund techniques might be available. In others, refunding may not be possible.

[3.5](#) Errors

Errors in formatting or the like that are internal to a payment or receipt should generally be handled by being logged and/or reported by an error message encoded into a receipt. Errors within a payment system in a price tag may be reported in a payment or receipt. Great care must be taken to be sure to avoid any situation that could result in an endless loop of receipts.

Errors outside of payment systems, such as receiving a payment via an unknown payment system or syntax errors that make it impossible to determine a payment system, should be reported via the normal error mechanism of the protocol within which the payments are embedded (see sections below).

[4.](#) Use in the World Wide Web

The World Wide Web is a rapidly evolving system for information interaction that is being increasingly used for commerce. It is particularly well suited for the inclusion of payment systems, especially any designed for efficient handling of small payments which might reasonably be incurred on a "per web page" basis or the like.

In the Web, a price is indicated by a COST parameter or by a payment required error response. As described below, a COST parameter can occur within an anchor, HTML document header, or several places in a FORM. Payment and receipts can be included with HTTP requests and payments using headers.

[4.1](#) Web Browser User Interface

[the user interface material could be moved to an appendix]

It is important that small payments be closely integrated into the browser user interface. An expected mode of operation will be one of many small payments, so the overhead associated with each must be small. It is unacceptable for the user to necessarily interact with a separate screen or window to approve each small payment although a user who wishes to do so should have that option.

The user should be able to establish some threshold (default perhaps around 0.15usd or equivalent) such that actions incurring that charge or less are semi-automatic. That is, no special approval action is required, although color coding or the like should be used to distinguish toll links from free links, an optional sound could be made when any money is sent, or other clues used to give the user a feel for what is going on.

To avoid spending an unexpectedly large amount in small pieces, possibly a bank graphic or the like should be displayed to show how much cash is still available to the browser before the user will have to take action. The act of refilling the bank would be a more heavyweight operation requiring user interaction or, to get a default amount, at least user approval.

[4.2](#) Anchor Embedded Costs

A cost can appear in an anchor. This is a very strong hint that payment of the indicated amount should accompany the GET or other operation that occurs when following that link. Note, however, that

it is ultimately up to the server being hit to determine if payment is adequate or to follow the course it chooses for different levels of payment.

The cost is given by a COST parameter in the anchor. For example:

```
<A COST="0.10usd 0.17cad paymentsystem=xxxxxx foocash=yyyyyy"
  HREF="http://mercenary.com/goodies">
Great stuff for one thin dime! </A>
```

It is recommended that toll links be shown in a different color or type style from toll-free links. Browsers may wish to go further and indicate different cost levels, particularly costs above or below any "automatic approval" level the user has. When the user has their pointer over the link, the browser may wish to display the payment particulars in a similar way to that in which it displays the URL. (Such a display could be filtered to the currency and/or payment system(s) actually available to the user.)

Notice that the cost, if any, indicated by the anchor text ("Great stuff..." above) could be different from the actual "COST=" parameter which controls the payment sent with the request. In turn, the "COST=" amount could be different from what the server really wants. Or the server may provide different data or services for different payment amount. Such variable payment schemes may be better handled with a FORM as described below.

[4.3](#) Page Header Price Tags

The cost for accessing an HTML page and the default cost for accessing any anchors or forms within the page can be included in the header. For example:

```
<HTML>
<head><title>Mating Habits of the Red Breasted Geek</title>

<meta http-equiv="www-cost" content="0.05usd xxxsys=on93h5M+p1l=">

<cost> 0.75usd 0.99cad xxxsys=A8jne8W2/sw== </cost></head>

<body> ... </body></HTML>
```

An attempt to get such a document from a payment aware server without payment of at least 5 cents via the xxxsys payment system should fail. (See Payment Required Error section below.) A second attempt with payment will be required. This could be done in a manner similar to an access restriction failure followed by a second attempt with access authorization information.

The <cost>...</cost> item in the head indicates that all anchors and forms on the page should be considered to have a price of 75 cents US

or 99 cents Canadian via the xxxsys payment system unless otherwise indicated.

4.4 HTML Form Price Tags

A cost can be associated with a form and with multiple choice items within the form. For example:

```
<form COST="xxxsys=A8jne8W2/sw== 0.75usd 0.99cad" ACTION=POST>
```

Miscellaneous text, etc.

```
<input type="radio" name="extras" value="omit">plain vanilla  
<input type="radio" name="extras" value="include"  
    COST="0.25usd 0.40cad">chocolate fudge
```

```
<p>Your quality of service: <select name="quality">  
<option value="bronze"> Low<p>  
<option value="silver" cost="0.10usd 0.17cad"> Medium<p>  
<option value="gold" cost="0.20usd 0.34cad">  
High<p> </select>  
</form>
```

The COST associated with the form is a base price (which may be defaulted from a <cost> item in the document header), to which any multiple choice item costs are added. The form level COST may be omitted and COSTs can still appear with multiple choice items. The COST associated with a "select" is a default that applies only if no item is selected. When an item is selected, it overrides the selection level cost and become the price component added into the total form price for that selection.

The normally required payment system string can be omitted from some of the form COST parameters, in which case any prices add to the amount for all payment systems, or the COST parameters can contain payment system name(s) without payment specific information to cause price information to add to the amount for the designated payment system(s). But one or more payment systems and their payment system specific parameters must be determinable if any payment is to be sent. The payment system specific information associated with the last encountered payment system field used in calculating the payment for the form is used. If no payment system field is encountered, then no payment will be sent with the request even though "COST=" parameters are present.

As with anchor costs, it is desirable to indicate the cost of

multiple choice items by color coding and the cost of activating the form by color coding the submit button. Note that the submit button could change from free to toll or the like as choices are made in the form.

[4.5](#) Payments and Receipts Via Miscellaneous Headers

[This section defines payments and receipts using existing HTTP header fields and one new such field. As an alternative, [section 4.6](#) defines then using PEP.]

Payment can accompany an HTTP request by including a payment line in the message header. This consists of the "ChargeTo:" header label followed by a payment system string; however, "ChargeTo:" can appear with one or more bare payment system names for the purpose of indicating that the browser understands those systems without conveying any actual payment. Examples:

```
ChargeTo: xxxcash=A8jne8W2/sw==
```

```
ChargeTo: foocash= barsys=
```

The first example is in the form of a payment via the xxxcash system. The second example is an indication by the sender that it understands the foocash and barsys payment systems.

The browser should keep track of such actual payments it has sent and re-send the identical payment if the request needs to be retried with access authorization information or due to a transient error, rather than sending additional funds.

The collection of payment or the specifics of the failure of a tendered payment are indicated back to the customer by a receipt line in the response header. This consists of the "receipt:" header label followed by a payment system string. For example:

```
Receipt: xxxcash=b93njexW2/swq4==
```

There are cases where a larger payment is collected initially and the unused portion refunded or where adjustments are required after a purchase. Because of this, ChargeTo and Receipt headers are both allowed in both HTTP requests and responses.

If an HTTP request arrives without sufficient payment (or with none

at all) and payment is required by the server, the server can simply provide a web page with limited or no actual information and possibly one or more links with COST parameters embedded in them. Alternatively, a "402 payment required" error can be returned, in

which case there must be a "www-cost:" response header field analogous to the "www-authenticate:" header field for a "401 unauthorized" response. The value of the www-cost field is the same as for the COST parameter described above.

This is similar to an access restriction error in that the browser can just try again with payment included the way it can try again with access information. It may be possible to combine these by returning a 402 error with the HTML accompanying the error having a link with a COST parameter pointing to the originally sought item. This would combine automatic charging for browsers that have 402 error processing implemented with a convenient way for the user to re-request with payment for browsers that understand anchor COST parameters but do not automatically handle 402 errors.

[4.6](#) Payments and Receipts Via PEP

[This section is just a direct translation of [section 4.5](#) into PEP [[draft-khare-http-pep-00.txt](#)]. Probably needs more work to fully take advantage of PEP.]

Payment can accompany an HTTP request by including a payment line in the message header. This consists of a "Protocol: {payment ...}" header where "..." is a payment system string.

It is also possible to use the Accept-protocol header with one or more bare payment system names for the purpose of indicating that the browser understands those systems without conveying any actual payment. Examples:

```
Protocol: { payment xxxcash=A8jne8W2/sw== }
```

```
Accept-protocol: { payment foocash= barsys= }
```

The first example is in the form of a payment via the xxxcash system. The second example is an indication by the sender that it understands

the foocash and barsys payment systems.

The browser should keep track of such actual payments it has sent and re-send the identical payment if the request needs to be retried with access authorization information or due to a transient error, rather than sending additional funds.

The collection of payment or the specifics of the failure of a tendered payment are indicated back to the customer by a receipt line in the response header. This consists of a "Protocol: { receipt ...}" header where "..." is a payment system string. For example:

Eastlake

[Page 15]

INTERNET-DRAFT

Internet Payment Framework

29 January 1996

Protocol: { receipt xxxcash=b93njexW2/swq4== }

There are cases where a larger payment is collected initially and the unused portion refunded or where adjustments are required after a purchase. Because of this, payment and receipt protocol and accept-protocol headers are both allowed in both HTTP requests and responses.

If an HTTP request arrives without sufficient payment (or with none at all) and payment is required by the server, the server can simply provide a web page with limited or no actual information and possibly one or more links with COST parameters embedded in them.

Alternatively, a "402 payment required" error can be returned, in which case there must be a "Protocol: { cost xxx }" response header field analogous to the "www-authenticate:" header field for a "401 unauthorized" response. The xxx field inside the protocol:cost header is the same as for the COST parameter described above.

This is similar to an access restriction error in that the browser can just try again with payment included the way it can try again with access information. It may be possible to combine these by returning a 402 error with the HTML accompanying the error having a link with a COST parameter pointing to the originally sought item. This would combine automatic charging for browsers that have 402 error processing implemented with a convenient way for the user to re-request with payment for browsers that understand anchor COST parameters but do not automatically handle 402 errors.

4.7 Web Proxies

[This section needs more work and the scoping provision of PEP [\[draft-khare-http-pep-00.txt\]](#) would almost certainly prove helpful.]

When information that has an owner and price is being cached and served to multiple different users by a proxy, the payments should be requested by the proxy. The safest thing for the proxy to do is to send payment to the entity it retrieved the data from using an HTTP request with a payment header and the OPTIONS method.

If the proxy understands the payment system well enough and there are no firewall problems, the proxy may be able to collect the payment and directly transfer funds to the information owner.

It is not expected that proxy payment collection will be perfect. There will initially be dumb proxies that don't understand payment and there may be proxies that deliberately avoid collecting and forwarding payment. But any large scale avoidance of payment will be noticed and corrected. In any case, if the proxy can cache a copy,

so could the user, who could then give copies to all his friends. The ease of automatically making small payments for information through this syntax is hoped to produce a net reduction in free copying of information for which the owner wished to impose a charge.

[5.](#) Use in File Transfer Protocol

An FTP server may wish to charge for a file transfer (either way) or for an FTP session.

It may do so by requesting, via the 332 or 532 reply codes, that an ACCT command be sent. 332 is used to indicate that a received command is being held in abeyance pending receipt of an ACCT while 532 indicates that a received command has been abandoned due to lack of payment and an ACCT command needs to be sent before attempting the command again.

Price tags are indicated in the 332 or 532 text by a string at the beginning of the form

<COST="foocash=xxxxx 0.05usd">

in the 332 or 532 text, i.e., a literal "<COST=" followed by a string conforming to the definition herein of a price tag, followed by a ">". The word "cost" is case insensitive. Arbitrary additional text may be included after the price tag.

A payment can be send by simply including a payment string, as defined in [section 3](#), after the ACCT command.

A successful receipt is rendered by returning a 233 reply with a receipt payment system string as the beginning of its text. A payment failure receipt is rendered by returning a 433 or 533 reply depending on whether the failure is transient or permanent. In either case, the receipt string can be terminated by white space and additional text human readable text placed after the receipt string in the reply.

(See [RFC 959](#).)

SUMMARY

Price Tags - in existing 332 and 532 replies.

Payments - in existing ACCT command.

Receipts - in new 233, 433, and 533, replies.

[6](#). Use in Telnet

A host may wish to charge for or during a Telnet session. Telnet option code <TBD> is used to initially negotiate agreement of the two parties to speak about payment. As with other Telnet options, either side can sent IAC WILL xxx, in response to which an IAC DO xxx

indicates agreement and an IAC DON'T xxx indicate refusal. Or a party can send IAC DO xxx to which IAC WILL xxx indicates agreement and an IAC WON'T xxx indicates refusal.

After agreement to speak about payment has been reached, Telnet subnegotiation strings can be exchanged, bracketed with IAC SB and IAC SE. The initial subnegotiation byte indicates the type of payment message following in the rest of the subnegotiation byte string as follows:

Byte	Meaning
----	-----
01	Price-tag
02	Payment
03	Receipt

(See RFCs 854, 855.)

7. Use in Simple Message Transfer Protocol

A host or user may wish to charge for the receipt of mail. This is accomplished via the new 332 reply code. This is an interim success code that indicates that further information is required to complete a pending command. Note that use of 332 after the SMTP RCPT command would be a simple way to implement any particular user requiring payment for mail to be delivered to them and its use after the MAIL command would be a simple way to implement a system requiring payment for mail from all or certain sources (although this information is easy to forge).

Payment is indicated by the new ACCT command. This is followed by a payment string as defined in [section 3](#) above.

Charging for mail may cut off a host or user from the normal flow of mail. It seems unlikely that most individuals or mailing lists would be willing to pay to send mail to such an addressee. However, it is easy to envision cases where a service for which it would be reasonable to charge is requested via email. Or there may be individuals who do want to substantially cut themselves off from most mail or mail from certain senders.

SMTP servers that speak ESMTP (see [RFC 1651](#)) may optionally give the new EHLO keyword ACCT. However, ESMTP is designed for servers to list features to be optionally invoked by clients. It is not really appropriate as a means for servers to indicate features that they will **require** of clients.

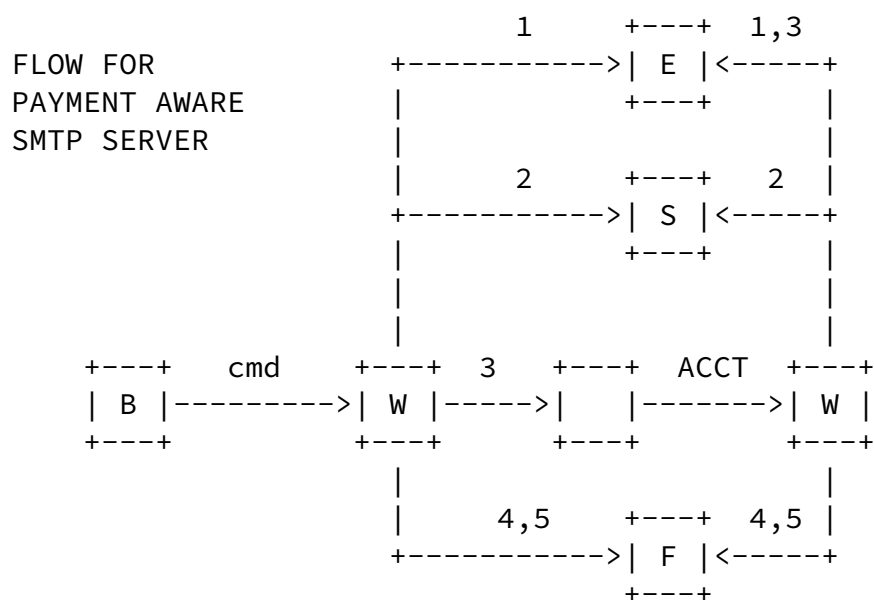
In any case, it is believed that no negotiation is necessary for an SMTP server to use the new 332 reply code. [RFC 821](#) is clear that the receipt of any 3xx reply code after a MAIL, RCPT, etc. command is to be considered an error. This is the appropriate understanding for an SMTP client that does not understand payment when an SMTP server requires payment.

The rules and state diagrams in [RFC-821](#) are hereby amended and the state diagram for MAIL, RCPT, SEND, SOML, and SAML is modified to the following:

INTERNET-DRAFT

Internet Payment Framework

29 January 1996



A successful receipt is rendered by returning the new 233 reply with a receipt payment system string as the beginning of its text. A payment failure receipt is rendered by returning the new 433 or 533 replies depending on whether the failure is transient or permanent. In either case, the receipt string can be terminated by white space and additional human readable text placed after the receipt string in the reply.

The middle digit 3 in SMTP reply codes is reserved for accounting, corresponding to its existing use in FTP.

(See RFCs 821, 1651.)

SUMMARY

Price Tags - in new 332 reply.

Payments - in new ACCT command.

Receipts - in new 233, 433, and 533 replies.

INTERNET-DRAFT

Internet Payment Framework

29 January 1996

8. The Domain Name System

The Domain Name System (DNS) is currently used for such fundamental purposes as translating domain names (such as tam.cybercash.com) into IP addresses or specifying SMTP mail backup and routing servers. For such uses DNS data is public and charges SHOULD NOT be imposed for DNS queries.

However, new uses of DNS, including dynamic update ([draft-ietf-dnsind-dynDNS-*.txt](#)), and particularly burdensome operations such as zone transfers of a large zone to other than a secondary, may warrant charges.

Payment information is all communicated via the PAY RR. The type for the PAY RR is <TBD> and its structre is as follows:

```

                                1 1 1 1 1 1
                                0 1 2 3 4 5
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           subtype           |                               /
+---+---+---+---+---+---+---+---+                               /
/                               data                               /
/                               /                                   /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The subtype indicates what the data is as follows:

0 - reserved.

1 - price tag.

2 - payment string.

3 - receipt string.

4-254 - available for IANA allocation.

255 - reserved.

PAY RRs occur in a zone and master file only under the zone name as a price tag indicating the fee for a zone transfer to anyone other than a zone secondary server.

Payment can be tendered with any request by including an appropriate PAY RR in the additional information section. WARNING: do not do this unless you are sure the server you are communicating with understands payments. Most current servers will ignore any request with a non-empty additional information section.

A receipt or price tag can be rendered by including an appropriate PAY RR in the additional information section of a reply. Error code

<tb> indicates payment is required and MUST be accompanied by a price tag PAY RR.

(See RFCs 1034, 1035, draft-ietf-dnsind-dynDNS-*.txt)

9. Protocols to Which Payment is not Applicable

Some protocols are inappropriate for the addition of payment mechanisms. Either they are sufficiently basic to the operation of the network or provide sufficiently light-weight access to public information or are so simple there is no obvious way to add extensions. Some of these protocols, listed below, SHOULD NOT make use of this syntax or impose prices or payments. This does not imply that there are not many more protocols for which it would be inappropriate to define payment extensions.

9.1 The ECHO, DISCARD, and CHARGEN Services

These are light weight services intended for network maintenance. ECHO echoes the packet sent to it (see [RFC 862](#)), DISCARD throws away

packets sent to it but maintains the connection (see [RFC 863](#)), and CHARGEN generates an infinite number of random characters and sends them until the calling party disconnects (see [RFC 864](#)).

Hosts are free to decide which, if any, of these three services they wish to provide (although ECHO is Recommended), but SHOULD NOT impose any charges for them. In any case, there really aren't any protocol hooks in these services on which to attach payment.

[9.2](#) The Finger Service

Finger is an optional information service intended to permit remote users to learn a limited amount of information about a user or users on an Internet host. Information such as the time they last logged in or contents of their ".plan" file. There are serious security considerations involved in allowing finger access to a host and hosts are free to decide how much such access, if any, they will provide.

In some cases, finger servers have been set up to act as information retrieval or reporting mechanisms, but this was not the designed purpose of finger and, in most cases, there are better mechanisms to provide such access.

If finger access is provided because a site wishes to be open, charges SHOULD NOT be imposed. In any case, the information returned by finger is free form text making it difficult to flag a payment requirement.

(See [RFC 1288](#).)

[9.3](#) The Auth Service

This service, when implemented, allows a remote host to determine the user associated with a TCP connection. It is intended as a security and auditing tool although it is weak in the face of anyone with direct access to the TPC or IP level who is attempting to mislead it. Implementation is optional.

Those who chose to provide this service are doing so to cooperate in

such security or auditing at some sacrifice in the privacy of their users. Charging for this service makes little sense in this context.

(See [RFC 931](#).)

Getting authorization to construct payments may, depending on the payment system, require the user to enter a passphrase. For example, a passphrase might be required at the beginning of their session to unlock a private key. Thus the user could be vulnerable to Trojan horse web browsers, ftp clients, telnet clients, etc., as they are to many other types of Trojan horse applications. Use of "secure" application distribution with signed executables, checksums, virus detection, etc., should be encouraged.

An adversary may be able to observe or modify traffic to and from an application. Payment systems should be designed so that such observation results in minimal loss of privacy and such observation or modification can not result in hijacking a payment. Note that an adversary that has complete control over application communications can pretend to be a merchant just as it could by controlling an end node. However, such impersonation from an end node may be easier to trace and control than impersonation at an unknown point along the communications path. Message (MOSS) and connection (IPSEC, IPv6) security protocols are available to help protect the communications path.

On receipt of an advance payment, a server is capable of charging the user regardless of whether the server actually provides the data or services being charged for. A server could even send back an error message but keep and use the payment. Some means of automatically logging payments that result in a software or human detectable failure to deliver should be implemented so these can be examined for patterns or cross checked with payment system statements of account.

A merchant can withhold and fail to send back to the user a receipt. Applications should assume any payment sent will be collected regardless of whether they get a receipt back.

With payment systems, a monetary cost can sometimes be associated with downloaded data. Caching algorithms may wish to take this into account and cache costly data in preference to free data. Servers should accept the identical data request from the same net entity for a reasonable amount of time even if the payment being presented is a duplicate. Transient errors may have prevented use of the data previously downloaded for that request.

A bad client application could generate payments exceeding the funds or authorization available to it. Servers should verify payments promptly and be cautious of extending services or goods unless they can confirm that payment is good. Applications and payment systems should be designed to limit the amount of funds a rogue application could transfer.

INTERNET-DRAFT

Internet Payment Framework

29 January 1996

References

[ISO 4217] - Codes for the representation of currencies and funds

[RFC 821] - J. Postel, "Simple Mail Transfer Protocol", 08/01/1982.

[RFC 854] - J. Postel, J. Reynolds, "Telnet option specifications", 05/01/1983.

[RFC 855] - J. Postel, J. Reynolds, "Telnet Protocol specification", 05/01/1983.

[RFC 959] - J. Postel, J. Reynolds, "File Transfer Protocol", 10/01/1985.

Author's Address

Donald E. Eastlake, 3rd
CyberCash, Inc.
318 Acton Street
Carlisle, MA 01741 USA

Telephone: +1 508 287 4877
 +1 508 371 7148 (fax)
 +1 703-620-4200 (main office, Reston, Virginia, USA)
email: dee@cybercash.com

Expiration and File Name

This draft expires 28 July 1996.

Its file name is [draft-eastlake-internet-payment-01.txt](#).

INTERNET-DRAFT

Internet Payment Framework

29 January 1996

Appendix A: Initial Payment System Names

This is an alphabetic list of the initial registered payment system names that intend to be usable via this syntax.

[send email to author, dee@cybercash.com, if you would like to be added]

Company Name	Email Contact	Home Page
-----	-----	-----
Payment System Name	-	(brief description)

CyberCash, Inc.	info@cybercash.com	http://www.cybercash.com
cybercash	-	(credit card)
cash	-	(cash)

Appendix B: Simplified BNF

This is a BNF-like description of the Payment Protocol syntax syntax, using the conventions of [RFC822](#), except that "|" is used to designate alternatives, and brackets [] are used around optional or repeated elements. Briefly, literals are quoted with "", optional elements are enclosed in [brackets], and elements may be preceded with <n>* to designate n or more repetitions of the following element or <n>*<m> to indicate n or more but not more than m repetitions; n defaults to 0.

;prices

isocurrency	= alpha alpha alpha
ietfcurrency	= 4*8alpha
privatecurrency	= 2*127[dns-label "."]
currency	= isocurrency ietfcurrency privatecurrency
digits	= 1*digit
decimal	= "." ","
number	= digits digits decimal *digit
exponent	= "+" digits "-" digits

cost	= number currency number exponent currency
------	--

;payment system strings

```

ianapaysys      = 4*12alpha
privatepaysys   = url
paysysname      = ianapaysys | url

paysys          = paysysname "=" *uchar

```

```

;price tag

```

```

pricetag = *sp paysys *[ 1*sp cost | 1*sp paysys ] *sp |
          *sp *[ cost 1*sp | paysys 1*sp ] paysys *sp

```

```

;miscellaneous definitions

```

```

lowalpha      = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
                "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
                "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
                "y" | "z"
hialpha       = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
                "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" |
                "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
                "Y" | "Z"

```

```

alpha         = lowalpha | hialpha
digit         = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                "8" | "9"
other         = "$" | "-" | "_" | "." | "+" | "/" | "=" | "@"
hex           = digit | "A" | "B" | "C" | "D" | "E" | "F" |
                "a" | "b" | "c" | "d" | "e" | "f"
escape        = "%" hex hex
sp            = " "
uchar         = alpha | digit | other | extra | escape

```

