Label Syntax and Communication Protocols

Status of this Memo

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 and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "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 ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited.

Comments on this draft should be sent to "pics-spec-comments@w3.org".

1. Introduction

This document has been prepared for the technical subcommittee of PICS (Platform for Internet Content Selection). It defines a general format for labels that permits them to be embedded in RFC-822-style headers. It defines three methods by which PICS labels may be transmitted:

In a document
: One or more labels may be embedded in a document. We specify the format and note in particular how to use a META tag to embed labels in HTML documents.

With a document
: An HTTP client can request that labels be sent along with a document. An HTTP server can satisfy the request, by sending the labels in RFC-822-style headers.

Separately
: A client can request labels from a "label bureau" that runs the HTTP protocol. The labels may refer to items available through protocols other than HTTP, such as ftp, gopher, or netnews. The simplest implementation of a label bureau is an off-the-shelf

HTTP server running a special CGI script.

2. General Format

A label consists of a _service identifier_, _label options_, and a _rating_. The service identifier is the URL chosen by the rating service (see [1], "Rating Services and Rating Systems") as its unique identifier. Label options give additional properties of the document being rated as well as the rating itself, such as the time the document was rated. The rating itself is a set of attribute-value pairs that describe a document along one or more dimensions. One or more labels may be distributed together as a list. The general form for a label list (formatted for presentation, and not showing error status codes) is:

```
(PICS-1.0
   <service url> [option...]
   labels [option...] ratings (<category> <value> ...)
          [option...] ratings (<category> <value> ...)
          ...
   <service url> [option...]
   labels [option...] ratings (<category> <value> ...)
          [option...] ratings (<category> <value> ...)
          ...
   ...)
```

Label options are as follows (some options can be abbreviated, as shown):

at _quoted-ISO-date_
: The last modification date of the item to which this rating applies, at the time the rating was assigned. This can serve as a less expensive, but less reliable, alternative to the message integrity check (MIC) options.

by _quotedname_
: An identifier for the person or entity within the rating service who is responsible for this particular label.

comment _quotedname_
: Information for humans who may see the label; no associated semantics.

complete-label _quotedURL_

full _quotedURL_
: Dereferencing this URL returns a complete label that can be used in place of the current one. The complete label has values for as many attributes as possible. This is used when a short label is transmitted for performance purposes but additional information is also available. When the URL is dereferenced it returns an item of type application/pics-labels that contains a labellist with exactly the one label.

extension (optional _quotedURL_ _data_*)

extension (mandatory _quotedURL_ _data_*)
   Future extension mechanism. To avoid duplication of extension
   names, each extension is identified by a _quotedURL_. The URL
   can be dereferenced to get a human-readable description of the
   extension. If the extension is *optional* then software which
   does not understand the extension can simply ignore it; if the
   extension is *mandatory* then software which does not understand
   the extension should act as though no label had been supplied.
   Each item of _data_ must be one of a fixed set of simple-to-parse
   data types as specified in the detailed syntax below.
for _quotedURL_
   The URL of the item to which this rating applies.
generic _boolean_
gen _boolean_
   This label can be applied to any URL starting with the prefix
   given in the *for* option. This is used to supply ratings for
   entire sites or directories.
MIC-md5 "_Base64-string_"
md5 "_Base64-string_"
   A message integrity check (MIC) of the item being rated. The MD5
   Message Digest Algorithm is used to compute the MIC. See [2],
   "RFC 1321".
on _quoted-ISO-date_
   The date on which this rating was issued.
signature-PKCS "_Base64-string_"
   An RSA digital signature encompassing the label as transmitted,
   signed by the rating service that issued the label. See section
   14, "MICs and Digital Signatures".
until _quoted-ISO-date_
exp _quoted-ISO-date_
   The date on which this rating expires.

3. Example

For example, a label that uses the example rating system from the document [1] "Rating Services and Rating Systems" might be as follows:

```
(PICS-1.0 "http://www.gcf.org"
   labels on "1994.11.05T08:15-0500"
      until "1995.12.31T23:59-0000"
      for "http://www.gcf.org/index.html"
      by "John Patrick"
      ratings (suds 0.5 density 0 color/hue 1))
```

The same label may be transmitted more compactly by converting all of the line breaks and subsequent indentation characters into a single space, and by replacing the word "labels" with "l", "ratings" with "r" and long option names with their abbreviations. It may be compressed for transmission purposes even further by removing all of the optional information to a separate document and referencing that

document by a URL:

```
   (PICS-1.0 "http://www.gcf.org" l
      full "http://www.gcf.org/labels/13242123"
      r (suds 0.5 density 0 color/hue 1))
```

Finally, the optional information may be omitted entirely, reducing the information content of the label but making the transmission even smaller. The resulting label would then be:

```
(PICS-1.0 "http://www.gcf.org" l r (suds 0.5 density 0 color/hue 1))
```

4. Detailed Syntax

The following grammar, in modified BNF, describes the syntax of labels. The methods by which labels are embedded in specific protocols are detailed below.

Notes:

1. Whitespace is ignored except in quoted strings.
2. The string in a _transmit-name_ is case insensitive. All other strings are case sensitive.
3. Option names ("on", "until", "at", etc.) are case insensitive.
4. This specification requires the use of US-ASCII. Note that the document [1] "Rating Services and Rating Systems" describes how a service can map the US-ASCII transmit-names to descriptive strings using other character sets.
5. An option that appears in the _service-info_ applies to all labels in that _service-info_ unless overridden by an option in a specific _label_. That is, a _label_ is effectively lexically nested within the enclosing _service-info_ for the purpose of understanding the applicable options. This is most likely to be useful in the case of the "at", "by", "generic", "until" and experimental or future options.
6. Numbers in PICS labels may be integers or fractions with no greater range or precision than that provided by IEEE single-precision floating point numbers.
7. The _multi-value_ syntax *must* be used when the value on a particular (multi-valued) scale has either zero or more than one value. It *may* be used for a single-valued or multi-valued field when there is exactly one value, but the more compact version may also be used in that case.
8. The only options that may occur more than once in a single label are "comment" and "extension"; if the "extension" option is supplied more than once, the _quotedURL_s defining the extensions must be distinct.

```
labellist :: '(' 'PICS-1.0' _service-info_+ ')'
service-info :: 'error' '(no-ratings' _explanation_* ')'
```

```
           | _serviceID_ _service-error_
           | _serviceID_ _option_* _labelword_ _label_*
serviceID :: _quotedURL_
labelword :: 'labels' | 'l'
label :: _label-error_ | _single-label_ | '(' _single-label_* ')'
single-label :: _option_* _ratingword_ '(' _rating_+  ')'
ratingword :: 'ratings' | 'r'
quotedURL :: '"' _URL_ '"' as described and extended in [1] "Rating
           Services and Rating Systems.
option :: 'at' _quoted-ISO-date_
     | 'by' _quotedname_
     | 'comment' _quotedname_
     | 'complete-label' _quotedURL_ | 'full' _quotedURL_
     | 'extension' '(' _mand/opt_ _quotedURL_ _data_* ')'
     | 'generic' _boolean_          | 'gen' _boolean_
     | 'for' _quotedURL_
     | 'MIC-md5' "_base64-string_"  | 'md5' "_base64-string_"
     | 'on' _quoted-ISO-date_
     | 'signature-PKCS' "_base64-string_"
     | 'until' _quoted-ISO-date_    | 'exp' _quoted-ISO-date_
mand/opt :: 'optional' | 'mandatory'
data :: _quoted-ISO-date_ | _quotedURL_ | _number_ | _quotedname_
     | '(' _data_* ')'
quoted-ISO-date :: '"'YYYY'.'MM'.'DD'T'hh':'mmStz'"'
  based on the ISO 8601:1988 date and time standard, restricted
  to the specific form described here:
  YYYY :: four-digit year
  MM :: two-digit month (01=January, etc.)
  DD :: two-digit day of month (01 through 31)
  hh :: two digits of hour (00 through 23) (am/pm NOT allowed)
  mm :: two digits of minute (00 through 59)
  S  :: sign of time zone offset from UTC ('+' or '-')
  tz :: four digit amount of offset from UTC
        (e.g., 1512 means 15 hours and 12 minutes)
  For example, "1994.11.05T08:15-0500" is a valid _quoted-ISO-date_
  denoting November 5, 1994, 8:15 am, US Eastern Standard Time.
  Note:  The ISO standard allows considerably greater flexibility
  than that described here.  PICS requires *precisely* the syntax
  described here -- neither the time nor the time zone may be
  omitted, none of the alternate formats are permitted, and the
  punctuation must be as specified here.
rating :: _transmit-name_ _number_ | _transmit-name_ '(' _multi-value_* ')'
multi-value :: _number_ | _number_ ':' _number_
transmit-name :: [1*n]_alphanumpm_ ['/' _transmit-name_]
number :: [_sign_]_unsignedint_['.' [_unsignedint_]]
sign :: '+' | '-'
unsignedint :: [1*n][0-9]
quotedname :: ' " ' [1*n]_extendedalphanum_ ' " '
alphanumpm :: 'A' | ... | 'Z' | 'a' | ... | 'z' | '+' | '-'
extendedalphanum :: _alphanumpm_ | '.' | ' ' | ',' | ';' | ':'
           | '&' | '=' | '?' | '!' | '*' | '~' | '@' | '#'
```

```
base64-string :: as defined in [3] "RFC 1521".
service-error :: 'error' '(' 'request-denied' _explanation_* ')'
           | 'error' 'service-unavailable'
label-error :: 'error' '(' request-denied' [_quotedURL_
           _explanation_*] ')'
           | 'error' '(' not-labeled' _quotedURL_* ')'
explanation :: _quotedname_
```

5. Semantics of PICS Labels and Label Lists

A _labellist_ is used to transmit a set of PICS labels. The format specified here is intended to be registered with IANA as the MIME type "application/pics-labels." It allows for transmission of both labels and reasons why labels are not available, and is the format used when labels must be conveyed in a document, along with a document, or from a PICS label bureau. The _labellist_ will always be surrounded by parentheses and begin with the PICS version number (1.0 in this specification).

A label list either specifies that there are no labels available at all ("error (no-ratings ...)") or is separated into sections of labels, one section for each rating service. The URL of each service must be specified (the _serviceID_). This is either followed by an error message indicating why no labels are available from that service (_service-error_) or an overall set of optional information (_option_*) followed by the keyword "labels" (or "l") and the _label_s from the service. The optional information provided here applies to every label from the service, unless overridden in the specific label itself.

A _label_ encompasses three separate cases. The first is an error that applies to retrieving the label for a particular URL (_label-error_). The second, and most common, is a _single-label_ consisting of options (which override those specified with the service), the marker word "ratings" (or "r") and the ratings themselves (a list of category names and values). Finally, in the special case where the ratings for an entire tree of documents have been requested, any number of _single-label_s can be transmitted, enclosed in parentheses. This case is described in more detail in the section on "Requesting Labels Separately".

A label may apply to a specific URL, or it may be generic. A generic label implicitly rates every URL for which the specified one is a prefix. For example, a generic label for the URL "http://www.gcf.org" implicitly rates every document available at that site. A regular (non-generic) label for the same URL, "http://www.gcf.org", does not give any implicit ratings: it merely rates the organization's home page that is fetched by the command "GET / " sent by HTTP to the host "www.gcf.org". A generic label *must* include the "for" option specifying the URL to which it

applies.

When a _multi-value_ is provided, any combination of numbers and ranges of numbers may be specified, with the endpoints of a range separated by a ":". Thus, in the labellist

```
(PICS-1.0 "http://www.gcf.org" l
   r (suds 0.5 density 0 color/hue 1 subject (0.5:2.5 3)))
```

all subject values between 0.5 and 2.5 (including both endpoints) apply to the item, as does the subject value 3. Given the example service description in [1], Rating Services and Rating Systems", all three document subjects apply, "soap", "water", and "soapdish".

6. RFC 822 Headers

Many protocols, such as Internet electronic mail, the HyperText Transfer Protocol, and USENET News, use ASCII headers as described in RFC 822. For use in such protocols, we define a new header, PICS-Label, used to contain the labels described in this document. The syntax is:

```
PICS-Label: <labellist>
```

where _labellist_ is described according to the syntax above. Continuation lines beginning with whitespace may be used following the specification given in RFC 822.

7. Embedding Labels in HyperText Markup Language (HTML)

Labels may be embedded in HTML files as meta-information, using the META element defined in the HTML specification. This embedding uses the HTTP header equivalency mechanism:

```
<META http-equiv="PICS-Label" content='_labellist_'>
```

(Note that the content attribute uses single quotes, because the PICS label syntax uses double quotes. Any of the following characters appearing within the content must be escaped using SGML entities:

```
'       &#39;           /* single quote */
&       &amp;           /*  ampersand   */
>       &gt;            /* greater than */
```

See [4], the "HTML 2.0 Proposed Standard".

8. Sending Labels With A Document

When an HTTP server sends a document to a client, it sends additional headers as well. We specify how the client can request that one or more labels be included in a header. HTTP servers should include PICS label headers only if requested to do so by the client, and should only include the labels from services requested by the client.

Example:

Client sends to HTTP server www.greatdocs.com:

```
GET foo.html HTTP/1.0
Accept-Protocol:
  {PICS-1.0 {params full
              {services "http://www.gcf.org/ratings"}}}
```

Server responds to client:

```
HTTP/1.0 200 OK
Date: Thursday, 30-Jun-95 17:51:47 GMT
MIME-version: 1.0
Last-modified: Thursday, 29-Jun-95 17:51:47 GMT
Protocol: {PICS-1.0 {headers PICS-Label}}
PICS-Label:
   (PICS-1.0 "http://www.gcf.org" labels
   on "1994.11.05T08:15-0500"
   exp "1995.12.31T23:59-0000"
   for "http://www.gcf.org/index.html"
   by "George Sanderson, Jr."
   ratings (suds 0.5 density 0 color/hue 1))
Content-type: text/html
...contents of foo.html...
```

Explanation of example:

The client requests the document foo.html. In addition, the client requests the full label of the document from the rating service "http://www.gcf.org/ratings". The server responds by sending back the label, in the PICS-Label header, as well as the document. The format of the PICS-Label header field (a _labellist_) allows the server to respond either with a label or an explanation of why the label is not available, since it would be inappropriate for the server to generate an HTTP error status if the document is available but (some of) the labels are not.

Following the usual HTTP distinction between HEAD and GET, a client that wishes to examine a rating before retrieving the full document can substitute the word HEAD for GET in the request. The server responds with exactly the headers shown above, but does not send back the document "foo.html".

9. Detailed Syntax of HTTP Requests for Labels With Document

The following grammar, in modified BNF, describes the syntax of the additional header line to be included in an HTTP request for a document and associated labels.

```
accept-header ::
   'Accept-Protocol: {PICS-1.0 {params ' [_completeness_]
      _extension_* _services_ '}}'
completeness :: 'minimal' | 'short' | 'full' | 'signed'
extension :: '{' _token-or-quoted-string_+ '}'
   where the first _token-or-quoted-string_ is not 'services'.
token-or-quoted-string :: _token_ | _quotedname_
token :: [1*n]_alphanumpm_
services :: '{' 'services' _quotedURL_+ '}'
```

A request for a *minimal* label asks that all options be omitted, unless a generic label is returned, in which case the "generic" and "for" options must also be included in the label. A *short* label includes everything that is included in a minimal label, plus additional options that the server deems appropriate. A request for a *full* label asks that as much information as possible should be sent back in the label, either directly or through the use of a "complete-label" (or "full") option, but no "signature-PKCS" option is needed.

A request for *signed* labels asks that all the information in a "full" label should be sent, along with a digital signature on the label itself. In a signed label the information must be transmitted directly as part of the label (and included in the computation of the signature); the "complete-label" (or "full") option may be sent, but it would be redundant. Details of signing labels are included in section 14, "MICs and Digital Signatures".

It is acceptable for a server to ignore the _completeness_, either by delivering more or fewer options than requested. If the _completeness_ is omitted, it should be treated as though "minimal" had been supplied. For future extensibility, any alphanumeric string may be used for a value of the _completeness_ option. Servers which receive a value of _completeness_ that they do not recognize must treat it as though "minimal" had been specified.

The _extension_s are for future extensions to the protocol; any extensions which are not understood by the server must be ignored by it. It is recommended that experimental extensions use a URL, which dereferences to a description of the extension, as the initial _token-or-quoted-string_.

Each _service_ specifies a rating service from which the client is requesting a label for the document. There may be as many repetitions of the _service_ part of the query as desired.

10. Detailed Syntax For HTTP Response Headers for Labels With Document

Two additional headers are specified:

```
protocol-header :: 'Protocol: {PICS-1.0 {headers PICS-Label}}'
label-header :: 'PICS-Label: ' _labellist_
```

11. Requesting Labels Separately

PICS labels can also be retrieved separately from the documents to which they refer. To request labels in this way, a client contacts a *label bureau*. A label bureau is an HTTP server that understands a particular query syntax, defined below. It can provide labels for documents that reside on other servers, and, indeed, for documents available through protocols other than HTTP. It is anticipated that there will be "well-known" label bureaus which dispense (possibly for a fee) labels created by many rating services.

Rating services are also encouraged to act as label bureaus, providing on-line access to their own labels. By default, the URL that identifies a rating service also identifies its label bureau. If a client requests the URL that identifies a rating service, a human-readable description of the service is returned, as specified in [1], "Rating Services and Rating Systems". If, on the other hand, a client requests the same URL and includes query parameters as defined below, it should be interpreted as a request for labels. A rating service, however, is not required to act as a label bureau, and it may choose a different URL (perhaps even on a different HTTP server) to act as its label bureau.

Sample Query:

Imagine a rating service, identified by the URL "http://www.labels.org/Ratings", which decides to run a label bureau to dispense (at least) its own labels for documents. The following sample request, made to the HTTP server "www.labels.org", is illustrative (line breaks are inserted for presentation purposes only):

```
GET /Ratings?opt=generic&
   u="http%3A%2F%2Fwww.questionable.org%2Fimages"&
   s="http%3A%2F%2Fwww.gcf.org%2Fratings"&
   HTTP/1.0
```

The query asks the label bureau "http://www.labels.org/Ratings" to send a single label that applies to everything in the images directory at site "www.questionable.org". The desired label should have been created by the service "http://www.gcf.org/ratings". Notice the use of %3A to represent a ":" and %2F for "/". This is

required for encoding characters within a URL. See [5], "RFC 1738".

The label bureau responds by sending back a document of type "application/pics-labels." The labels should be as complete as possible, either by including as many options as possible or by supplying the "complete-label" (or "full") option.

12. Detailed Syntax and Semantics of HTTP Query for Labels Separate From Documents

The following grammar, in modified BNF, describes the syntax of the GET request to a label bureau:

```
get :: 'get' _url-fragment_ '?' [_opt_] [_format_] _extension_*
   _url_+ _service_+
url-fragment :: the part of the original URL after the host name, as
   specified in HTTP 1.0.
opt :: 'opt=' _option_
option :: 'generic' | 'normal' | 'tree' | 'generic+tree'
format :: [and] 'format=' _form_
form :: 'minimal' | 'short' | 'full' | 'signed'
extension :: _token_ '=' _token-or-quoted-string_
   where the _token_ is not one of "opt", "format", "u", or "s"; and
   _token-or-quoted-string_ follows the quoting conventions
   specified in [5], "RFC 1738".
token-or-quoted-string :: _token_ | _quotedname_
token :: [1*n]_alphanumpm_
url :: [and] 'u=' encodedURL
service :: [and] 's=' encodedURL
boolean :: 't' | 'f' | 'true' | 'false'
and :: '&' this must be included unless it immediately follows the ?
   in the query.
encodedURL :: a URL, with quotation as required for inclusion within
another URL. According to [5], "RFC 1738", quotation is done using
"%xx" notation. Alphabetic characters, digits, and the special
characters $_-.+!*'(), need not be quoted, but other characters must
be. This *does* imply that the colon (:) must be encoded as %3A and
slash (/) as %2F.
```

Notes:

1. "opt=generic" requests generic labels. For each requested URL, the desired response is a generic label that implicitly applies to all URLs matching it. This is useful for requesting a rating of a site or directory.
2. "opt=tree" requests a tree of labels. For each requested URL, the desired response is all labels for URLS that match it. This is a way to request all the labels for items in a directory or a site. In the response, everywhere a _label_ would normally be expected in the response, a set of

_simple-label_s will be returned, surrounded by parentheses.

3. "opt=generic+tree" requests all generic labels that apply to matching URLs. This is a way to request generic labels for all of the directories at a site. In the response, everywhere a _label_ would normally be expected in the response, a set of _simple-label_s will be returned, surrounded by parentheses.
4. "opt=normal", or omitting the "opt" completely, requests specific labels for the URLs specified.
5. It is permitted to include more than one URL in the request.
6. The "format=" specifies the optional information that should be transmitted with the labels. It is treated precisely as the similar keywords would be when sent to a document server as the "completeness" (see section 9), except that the default is "full" (rather than "minimal"). Servers which receive a value of "completeness" that they do not recognize must treat it as though the default, "full" had been specified.

13. Detailed Syntax and Semantics of Response to Query for Labels Separate From Documents

The label bureau responds by sending back a document of type "application/pics-labels". Unless the document indicates an overall error, there should be one _service-info_ for each rating service requested in the query. Each _service-info_ should have an error message or a label (or list of labels, in the case of a "tree" query) for each requested URL.

The query's ordering must be preserved in the response. That is, the information from the rating services must be presented in the same order the rating services appear in the query, and the labels from each service must be presented in the same order the URLs appear in the query. If a rating service or label is not provided, the error message should appear in the same position that the _service-info_ or label would appear. Because order is preserved, it is acceptable to omit from the labels the "for" option which indicates the URL being rated (*unless* the label is "generic" in which case, as always for generic labels, the "for" is required). The client should match the label positionally with the URL for which it requested a rating.

In response to a request for a generic label, only a generic label may be returned. In response to a request for a regular label, a generic label for a URL that is a prefix of the requested URL may be returned. For example, in response to a label request for URL "http://www.gcf.org/index.html" a generic label for the URL "http://www.gcf.org" may be returned. In this case, it is required that the "for" and "generic" options be included in the label, to specify exactly what rating is being returned.

For a tree request, all the labels sent in response to a particular URL are enclosed in parentheses, so the client can match them positionally with the single request URL. The "for" option must be included in such labels to specify exactly which URLs the labels apply to.

14. MICs and Digital Signatures

This section remains to be specified. There are three particular difficulties that must be addressed:

1. On what data is the MIC included in the _mic-md5_ (or _md5_) option computed? In particular, if the URL "ftp://www.somewhere.com/Pictures/Interesting/Look.gif" refers to a compressed GIF image, is the MIC computed on the compressed or uncompressed form? Does it depend on the content-transfer-encoding? The MIME type?
2. How is the label canonicalized before computing the digital signature? Because header lines can be folded by various transports, it is important that a canonical form be carefully defined. Clearly, it should not include the signature itself, but does it include all of the other optional fields? Does a signed label necessarily imply a full label (hence the distinction should be dropped)?
3. How are the public keys for rating services distributed? Can it be done using a variant on the same technique used for communicating with a label bureau or is a full certificate authority required? What authority should be used or can multiple be used? Is the service's URL a satisfactory distinguished name for use with a certificate authority?

15. Security Considerations

Security considerations will be addressed in future revisions of this draft.

16. Glossary

application/pics-service
: A new MIME data type used to describe a _rating service_, defined in [1], "Rating Services and Rating Systems".

application/pics-labels
: A new MIME data type used to transmit one or more _labels_, defined in this document.

BNF
: Backus-Naur Form (or Backus Normal Form). A notation for describing a formal syntax, used extensively in describing

programming languages and computer-readable data formats.

category
: The part of a rating system which describes a particular criterion used for rating. For example, a rating system might have three categories named "sexual material," "violence," and "vocabulary." Also called a _dimension_.

content label
: A data structure containing information about a given document's contents. Also called a _rating_ or _content rating_. The content label may accompany the document it is about or be available separately.

content rating
: See _content label_.

dimension
: See _category_.

HTML
: HyperText Markup Language. A means of representing _hypertext_ documents. Based on _SGML_. See [4], the "HTML 2.0 Proposed Standard".

HTTP
: HyperText Transfer Protocol. Used for retrieving document contents and/or descriptive header information.

hypertext
: Text, graphics, and other media connected through links.

label
: See _content label_.

MD5
: An algorithm, see [2], "RFC 1321", that can be used to compute a MIC. PICS specifies this particular algorithm for use in PICS labels.

MIC
: Message Integrity Check. Also known as a "cryptographic checksum." For PICS, the importance of a MIC is that a rating service can compute the MIC of a piece of information when the label is created and that MIC can be put into the label itself. A client can retrieve the label and the information to which it is supposed to be attached, recompute the MIC and compare it to the one in the label. If they match, for all practical purposes, it is a proof that the label really belongs to the information that has been retrieved. The particular algorithm specified by PICS to compute the MIC is MD5.

MIME
: Multimedia Internet Message Extension. A technique for sending arbitrary data through electronic mail on the Internet. See [3], "RFC 1521".

PICS
: Platform for Internet Content Selection, the name for both the suite of specification documents of which this is a part, and for the organization writing the documents. For more information, see the PICS home page on the World Wide Web at: "http://www.w3.org/PICS".

rating
   See _content label_.
label bureau
   A computer system which supplies, via a computer network, ratings of documents. It may or may not provide the documents themselves.
rating server
   See _label bureau_.
rating service
   An individual or organization that assigns labels according to some rating system, and then distributes them, perhaps via a label bureau or via CD-ROM.
rating system
   A method for rating information. A rating system consists of one or more _categories_.
scale
   The range of permissible values for a category.
SGML
   Standard Generalized Markup Language. See ISO 8879.
transmission name
   (of a _category_) The short name intended for use over a network to refer to the category. This is distinct from the category name in as much as the transmission name must be language-independent, encoded in ASCII, and as short as reasonably possible. Within a single _rating system_ the transmission names of all categories must be distinct.
URL
   Uniform Resource Locator. Described in [5], "RFC 1738". A URL describes the location and means of retrieval for a single document. It consists of three components: the "scheme" (protocol used to retrieve a document, like "http" or "ftp"), a host name, and a hierarchical document name within that host. For example "http://www.w3.org/PICS" is the URL of the PICS home page. The scheme for retrieving it is "http," the host is "www.w3.org" and the name within that host is "PICS".

17. References

[1] PICS, "Rating Services and Rating Systems", Internet Draft, "draft-pics-services-00.txt", 11/21/95.

[2] R. Rivest, "The MD5 Message-Digest Algorithm", RFC 1321, 04/16/1992.

[3] N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, 09/23/1993.

[4] T. Berners-Lee, D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, 11/03/1995.

[5] T. Berners-Lee, L. Masinter, M. McCahill, "Uniform Resource Locators (URLs)", RFC 1738, 12/20/94.

18. Acknowledgments

Primary authors of this document:

Tim Krauskopf, Spyglass
Jim Miller, W3C
Paul Resnick, AT&T
G. Winfield Treese, OpenMarket

Additional contributors:

Brenda Baker, AT&T
Tim Berners-Lee, W3C
Roxana Bradescu, AT&T
Daniel W. Connolly, W3C
Roy Fielding, W3C
Jay Friedland, SurfWatch
Michael Gordon, Prodigy
Wayne Gramlich, Sun
Woodson Hobbs, NewView
Rohit Khare, W3C
Charlie Kim, Apple
John C. Klensin, MCI
Ann McCurdy, Microsoft
Rich Petke, CompuServe
Dave Raggett, W3C
Bob Schloss, IBM
David Singer, IBM
Michael Smith, Prodigy
Marcy Swenson, Providence Systems
Jason Thomas, MIT

19. Author's Address

PICS Technical Committee
World Wide Web Consortium
545 Technology Square
Cambridge, MA 02139
Phone: 617-253-3194
EMail: pics-spec-comments@w3.org

Temporary Appendix A: Why HTTP For Label Bureaus

This section is not expected to be contained in future versions of this document.

Instead of extending HTTP, we considered proposals for special-purpose label transport protocols. Before making a final decision, we constructed the following lists of pros and cons.

Advantages of Using HTTP

- An existing HTTP server can be used as a PICS label bureau. This is particularly useful in the short term. CGI scripts at the HTTP server can handle the special header fields of a request for labels.
- A label returned from a label bureau and a label returned along with a document from an HTTP server can use identical label formats.
- Client programs that already support HTTP will have much less new code to implement.
- Client programs that do not support HTTP will have to support a new protocol in any case. It may be easier to support HTTP than a newly defined label transport protocol, because of available software libraries.
- Several protocol elements are already fully specified by HTTP that would be required in any PICS protocol.

    - Date and time formats.
    - Content encoding types.
    - Character set and Internationalization issues.
    - Error/result conditions. Both result categories (extensible), as well as a sample set of messages are specified.
    - Handling of expiration dates for each URL queried.

- HTTP is quite stable, has not diverged, and is well accepted.
- Security and payment systems either exist or are being developed for HTTP. A binary format may also be developed for speed. PICS need not reinvent such systems.
- Firewalls tend to allow HTTP headers to be transmitted already. A new protocol would take much longer to be accepted.
- A reliable connection (initially TCP based), ASCII-based protocol seems desirable initially.
- Current extensibility already defines how extensions to PICS itself should be accomplished.

Advantages of Creating a New Protocol Instead of Using HTTP

- A new protocol would avoid any HTTP protocol wars.
- Label bureaus and clients would not need to be updated to accommodate HTTP changes.
- RFC 822 and other precedents could still be used in the design of a new protocol.
- A binary format could be considered initially for speed.
- UDP or other datagram lookups could be considered.

Temporary Appendix B: FAQ - Frequently Asked Questions

This section is not expected to be contained in future versions of

this document.

Why is there no ftp, gopher, or netnews protocol for requesting labels along with a document?

> Labels can be sent as additional headers in any protocol that employs RFC 822 style headers. We have not yet determined, however, convenient extensions to protocols other than HTTP to permit requests that ask for labels created by specific services. We may specify such extensions in the future.

How do you get labels for items on FTP, Gopher, or netnews servers? Are we forcing all FTP implementations to implement all of HTTP as well?

> FTP, Gopher, and netnews servers need not distribute PICS labels. Labels for items on such servers can be retrieved from an HTTP-based label bureau.
>
> The PICS premise is that all compliant clients will have to implement some new protocol. The subset of HTTP which would be required for obtaining a PICS label can be minimal. HTTP will be no more difficult to implement in an FTP (or other) client than a brand-new protocol that provides similar features.

Can existing HTTP servers be used as PICS label bureaus?

> Using CGI scripts, or with a small amount of added code in the HTTP server, an existing HTTP server can be configured to access a database of labels and return that information coded as additional HTTP Headers. Most of the work is in the lookup and formatting of the labels themselves, not the modifications to HTTP.

How do I design a really fast PICS label bureau? Won't the overhead be too much?

> HTTP already explicitly defines the minimum fields required and then what rules must be followed when additional information is useful to the transaction. For example, HTTP does not require that clients provide "Accept:" headers to indicate preferred MIME types for the content, but if they are provided, servers can match up available formats with the client's request. An HTTP server may be designed to optimize throughput or to optimize the appearance of the result, or to adjust to the client software's preference.
>
> If you minimize the server's response to one line, plus the label information, you are already dealing with the minimum amount of data transfer possible to obtain a label. In addition, most performance issues for PICS will probably be addressed with caching, not by reducing lookup time for a single label. Caching

optimization requires meta-data which can be easily encoded within HTTP headers.

How can we keep the PICS extensions from getting tied up in HTTP standardization?

The management of header extensions for HTTP has been an issue of discussion and work by the HTTP group for some time. The HTTP specification lays down specific rules for the handling of extensions which guarantee that those extensions will not be made invalid by any revisions of HTTP itself. In addition, the W3C is working on a system (PEP) for managing and negotiating HTTP extensions even more intelligently.

The worst risk seems to be that HTTP could be upgraded to a new revision level forcing some HTTP implementations to support multiple versions (1.0 and 2.0, for example) or forcing some PICS bureaus to update their protocol as well. Hopefully a major update in HTTP would bring enough benefits for PICS to make any update worthwhile.

What is PEP and Why is PICS Using It?

The Protocol Extension Proposal from the World Wide Web Consortium uses a trio of header fields (Protocol, Accept-Protocol, and Content-Encoding) to allow a HTTP client and server to do sophisticated negotiation about the set of header fields and their meanings. It is being proposed for use in HTTP 1.2 and HTTP-ng, and is currently under careful scrutiny by the W3C Security Editorial Board to make sure that it contains the features necessary to provide security for general document transmission as well as electronic payments.

PICS faces many of the same problems that face the security and electronic payment community. In PICS the issue revolves around the ability for the client to tell the server from which rating services it would like to have labels. This is a simple negotiation problem of the kind PEP was designed to solve. Rather than invent an orthogonal mechanism it seemed best to use one that is already being proposed and investigated.

What if PEP Does Not Catch On?

If the general extension mechanism specified by PEP does not become a generic feature of HTTP servers, PICS label bureaus will need to look for the specific header line beginning Accept-Protocol: PICS/1.0 and process it to determine the rating request. PICS clients will need to look for and process the specific header lines PICS-Label and PICS-Status. We will also have to hope that no other group tries to extend HTTP in a way that uses headers named PICS-Label or PICS-Status.

This Internet Draft Expires on May 21, 1996