INTERNET-DRAFT   Robert Herriot (editor)
<draft-ietf-ipp-protocol-v11-03.txt>   Xerox Corporation
                                                         Sylvan Butler
                                                        Hewlett-Packard
                                                             Paul Moore
                                                              Microsoft
                                                          Randy Turner
                                                             2wire.com
                                                             John Wenn
                                                       Xerox Corporation
                                                          June 23, 1999

Internet Printing Protocol/1.1: Encoding and Transport

Status of this Memo

This document is an Internet-Draft and is in full conformance with all
provisions of Section 10 of [RFC2026]. 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".

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed as
http://www.ietf.org/shadow.html.

Abstract

This document is one of a set of documents, which together describe all
aspects of a new Internet Printing Protocol (IPP). IPP is an application
level protocol that can be used for distributed printing using Internet
tools and technologies. This document defines the rules for encoding IPP
operations and IPP attributes into a new Internet mime media type called
"application/ipp". This document also defines the rules for
transporting over HTTP a message body whose Content-Type is
"application/ipp". This document defines a new scheme named 'ipp' for
identifying IPP printers and jobs.

The full set of IPP documents includes:

    Design Goals for an Internet Printing Protocol [RFC2567]
    Rationale for the Structure and Model and Protocol for the Internet
    Printing Protocol [RFC2568]
    Internet Printing Protocol/1.1: Model and Semantics [ipp-mod]
    Internet Printing Protocol/1.1: Encoding and Transport (this
    document)
    Internet Printing Protocol/1.1: Implementer's Guide [ipp-iig]
    Mapping between LPD and IPP Protocols [RFC2569]

The document, "Design Goals for an Internet Printing Protocol", takes a
broad look at distributed printing functionality, and it enumerates
real-life scenarios that help to clarify the features that need to be
included in a printing protocol for the Internet. It identifies
requirements for three types of users: end users, operators, and
administrators. It calls out a subset of end user requirements that are
satisfied in IPP/1.1. A few OPTIONAL operator operations have been added
to IPP/1.1.

The document, "Rationale for the Structure and Model and Protocol for
the Internet Printing Protocol", describes IPP from a high level view,
defines a roadmap for the various documents that form the suite of IPP
specification documents, and gives background and rationale for the IETF
working group's major decisions.

The document, "Internet Printing Protocol/1.1: Model and Semantics",
describes a simplified model with abstract objects, their attributes,
and their operations that are independent of encoding and transport. It
introduces a Printer and a Job object. The Job object optionally
supports multiple documents per Job. It also addresses security,
internationalization, and directory issues.

The document "Internet Printing Protocol/1.1: Implementer's Guide",
gives advice to implementers of IPP clients and IPP objects.

The document "Mapping between LPD and IPP Protocols" gives some advice
to implementers of gateways between IPP and LPD (Line Printer Daemon)
implementations.

Table of Contents

1. Introduction

This document contains the rules for encoding IPP operations and describes two layers: the transport layer and the operation layer.

The transport layer consists of an HTTP/1.1 request or response. RFC 2616 [RFC2616] describes HTTP/1.1. This document specifies the HTTP headers that an IPP implementation supports.

The operation layer consists of a message body in an HTTP request or response. The document "Internet Printing Protocol/1.1: Model and Semantics" [ipp-mod] defines the semantics of such a message body and the supported values. This document specifies the encoding of an IPP operation. The aforementioned document [ipp-mod] is henceforth referred to as the "IPP model document"

2. Conformance Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Encoding of the Operation Layer

The operation layer MUST contain a single operation request or operation response. Each request or response consists of a sequence of values and attribute groups. Attribute groups consist of a sequence of attributes each of which is a name and value. Names and values are ultimately sequences of octets

The encoding consists of octets as the most primitive type. There are several types built from octets, but three important types are integers, character strings and octet strings, on which most other data types are built. Every character string in this encoding MUST be a sequence of characters where the characters are associated with some charset and some natural language. A character string MUST be in "reading order" with the first character in the value (according to reading order) being the first character in the encoding. A character string whose associated charset is US-ASCII whose associated natural language is US English is henceforth called a US-ASCII-STRING. A character string whose associated charset and natural language are specified in a request or response as described in the model document is henceforth called a LOCALIZED-STRING. An octet string MUST be in "IPP model document order" with the first octet in the value (according to the IPP model document order) being the first octet in the encoding

Every integer in this encoding MUST be encoded as a signed integer using
two's-complement binary encoding with big-endian format (also known as
"network order" and "most significant byte first"). The number of octets
for an integer MUST be 1, 2 or 4, depending on usage in the protocol.
Such one-octet integers, henceforth called SIGNED-BYTE, are used for the

version-number and tag fields. Such two-byte integers, henceforth called
SIGNED-SHORT are used for the operation-id, status-code and length
fields. Four byte integers, henceforth called SIGNED-INTEGER, are used
for values fields and the sequence number.

The following two sections present the operation layer in two ways

   - informally through pictures and description

   - formally through Augmented Backus-Naur Form (ABNF), as specified by
      RFC 2234 [RFC2234]

3.1 **Picture of the Encoding**

The encoding for an operation request or response consists of:

```
    -----------------------------------------------
    |                version-number               |   2 bytes  - required
    -----------------------------------------------
    |             operation-id (request)          |
    |                     or                       |   2 bytes  - required
    |             status-code (response)          |
    -----------------------------------------------
    |                  request-id                 |   4 bytes  - required
    ---------------------------------------------------------
    |             xxx-attributes-tag              |   1 byte  |
    -----------------------------------------------          |-0 or more
    |            xxx-attribute-sequence           |   n bytes |
    ---------------------------------------------------------
    |            end-of-attributes-tag            |   1 byte   - required
    -----------------------------------------------
    |                    data                     |   q bytes - optional
    -----------------------------------------------
```

The xxx-attributes-tag and xxx-attribute-sequence represents four
different values of "xxx", namely, operation, job, printer and
unsupported. The xxx-attributes-tag and an xxx-attribute-sequence
represent attribute groups in the model document. The xxx-attributes-tag
identifies the attribute group and the xxx-attribute-sequence contains
the attributes.

The expected sequence of xxx-attributes-tag and xxx-attribute-sequence
is specified in the IPP model document for each operation request and
operation response.

A request or response SHOULD contain each xxx-attributes-tag defined for

that request or response even if there are no attributes except for the
unsupported-attributes-tag which SHOULD be present only if the
unsupported-attribute-sequence is non-empty. A receiver of a request
MUST be able to process as equivalent empty attribute groups:

  a) an xxx-attributes-tag with an empty xxx-attribute-sequence,

  b) an expected but missing xxx-attributes-tag.

The data is omitted from some operations, but the end-of-attributes-tag
is present even when the data is omitted. Note, the xxx-attributes-tags
and end-of-attributes-tag are called 'delimiter-tags'. Note: the xxx-
attribute-sequence, shown above may consist of 0 bytes, according to the
rule below.

An xxx-attributes-sequence consists of zero or more compound-attributes.

```
    -------------------------------------------------
    |               compound-attribute              |   s bytes - 0 or more
    -------------------------------------------------
```

A compound-attribute consists of an attribute with a single value
followed by zero or more additional values.

Note: a 'compound-attribute' represents a single attribute in the model
document. The 'additional value' syntax is for attributes with 2 or
more values.

Each attribute consists of:

```
    -------------------------------------------------
    |                  value-tag                    |   1 byte
    -------------------------------------------------
    |            name-length  (value is u)          |   2 bytes
    -------------------------------------------------
    |                    name                       |   u bytes
    -------------------------------------------------
    |           value-length  (value is v)          |   2 bytes
    -------------------------------------------------
    |                   value                       |   v bytes
    -------------------------------------------------
```

An additional value consists of:

```
    -------------------------------------------------------------
    |                   value-tag                   |   1 byte  |
    -------------------------------------------------           |
    |          name-length  (value is 0x0000)       |   2 bytes |
    -------------------------------------------------           |-0 or more
    |            value-length (value is w)          |   2 bytes |
    -------------------------------------------------           |
    |                   value                       |   w bytes |
    -------------------------------------------------------------
```

Note: an additional value is like an attribute whose name-length is 0.

>From the standpoint of a parsing loop, the encoding consists of:

```
     -------------------------------------------------
     |                 version-number               |   2 bytes  - required
     -------------------------------------------------
     |              operation-id (request)        |
     |                      or                    |   2 bytes  - required
     |              status-code (response)        |
     -------------------------------------------------
     |                   request-id               |   4 bytes  - required
     ----------------------------------------------------
     |       tag (delimiter-tag or value-tag)     |   1 byte  |
     -------------------------------------------------       |-0 or more
     |          empty or rest of attribute        |   x bytes |
     ----------------------------------------------------
     |              end-of-attributes-tag          |   2 bytes  - required
     -------------------------------------------------
     |                     data                    |   y bytes  - optional
     -------------------------------------------------
```

The value of the tag determines whether the bytes following the tag are:

  - attributes

  - data

  - the remainder of a single attribute where the tag specifies the
     type of the value.

3.2 Syntax of Encoding

The syntax below is ABNF [RFC2234] except 'strings of literals' MUST be
case sensitive. For example 'a' means lower case 'a' and not upper case
'A'.   In addition, SIGNED-BYTE and SIGNED-SHORT fields are represented
as '%x' values which show their range of values.


```
  ipp-message = ipp-request / ipp-response
  ipp-request = version-number operation-id request-id
          *(xxx-attributes-tag  xxx-attribute-sequence) end-of-
  attributes-tag data
  ipp-response = version-number status-code request-id
          *(xxx-attributes-tag xxx-attribute-sequence)  end-of-
  attributes-tag  data
  xxx-attribute-sequence = *compound-attribute

  xxx-attributes-tag = operation-attributes-tag / job-attributes-tag /
        printer-attributes-tag / unsupported-attributes-tag
```

```
version-number = major-version-number minor-version-number
major-version-number = SIGNED-BYTE  ; initially %d1
minor-version-number = SIGNED-BYTE  ; initially %d0

operation-id = SIGNED-SHORT    ; mapping from model defined below
status-code = SIGNED-SHORT  ; mapping from model defined below
```

```
request-id = SIGNED-INTEGER ; whose value is > 0

compound-attribute = attribute *additional-values

attribute = value-tag name-length name value-length value
additional-values = value-tag zero-name-length value-length value

name-length = SIGNED-SHORT    ; number of octets of 'name'
name = LALPHA *( LALPHA / DIGIT / "-" / "_" / "." )
value-length = SIGNED-SHORT  ; number of octets of 'value'
value = OCTET-STRING

data = OCTET-STRING

zero-name-length = %x00.00          ; name-length of 0
operation-attributes-tag =  %x01          ; tag of 1
job-attributes-tag   =  %x02             ; tag of 2
printer-attributes-tag =  %x04           ; tag of 4
unsupported- attributes-tag =  %x05      ; tag of 5
end-of-attributes-tag = %x03             ; tag of 3
value-tag = %x10-FF

SIGNED-BYTE = BYTE
SIGNED-SHORT = 2BYTE
SIGNED-INTEGER = 4BYTE
DIGIT = %x30-39    ;  "0" to "9"
LALPHA = %x61-7A   ;  "a" to "z"
BYTE = %x00-FF
OCTET-STRING = *BYTE
```

The syntax allows an xxx-attributes-tag to be present when the xxx-attribute-sequence that follows is empty. The syntax is defined this way to allow for the response of Get-Jobs where no attributes are returned for some job-objects. Although it is RECOMMENDED that the sender not send an xxx-attributes-tag if there are no attributes (except in the Get-Jobs response just mentioned), the receiver MUST be able to decode such syntax.

3.3 Version-number

The version-number MUST consist of a major and minor version-number, each of which MUST be represented by a SIGNED-BYTE. The protocol described in this document MUST have a major version-number of 1 (0x01) and a minor version-number of 1 (0x01). The ABNF for these two bytes MUST be %x01.01.

[3.4](#) Operation-id

Operation-ids are defined as enums in the model document. An operation-ids enum value MUST be encoded as a SIGNED-SHORT.

Note: the values 0x4000 to 0xFFFF are reserved for private extensions.

3.5 **Status-code**

Status-codes are defined as enums in the model document. A status-code
enum value MUST be encoded as a SIGNED-SHORT.

The status-code is an operation attribute in the model document. In the
protocol, the status-code is in a special position, outside of the
operation attributes.

If an IPP status-code is returned, then the HTTP Status-Code MUST be 200
(successful-ok). With any other HTTP Status-Code value, the HTTP
response MUST NOT contain an IPP message-body, and thus no IPP status-
code is returned.

3.6 **Request-id**

The request-id allows a client to match a response with a request. This
mechanism is unnecessary in HTTP, but may be useful when application/ipp
entity bodies are used in another context.

The request-id in a response MUST be the value of the request-id
received in the corresponding request. A client can set the request-id
in each request to a unique value or a constant value, such as 1,
depending on what the client does with the request-id returned in the
response. The value of the request-id MUST be greater than zero.

3.7 **Tags**

There are two kinds of tags:

  - delimiter tags: delimit major sections of the protocol, namely
     attributes and data

  - value tags: specify the type of each attribute value

3.7.1 **Delimiter Tags**

The following table specifies the values for the delimiter tags:

    Tag Value (Hex)   Delimiter

  0x00   reserved
  0x01   operation-attributes-tag
  0x02   job-attributes-tag
  0x03   end-of-attributes-tag
  0x04   printer-attributes-tag
  0x05   unsupported-attributes-tag
  0x06-0x0e   reserved for future delimiters
  0x0F   reserved for future chunking-end-of-attributes-
                      tag

When an xxx-attributes-tag occurs in the protocol, it MUST mean that
zero or more following attributes up to the next delimiter tag are
attributes belonging to group xxx as defined in the model document,
where xxx is operation, job, printer, unsupported.

Doing substitution for xxx in the above paragraph, this means the
following. When an operation-attributes-tag occurs in the protocol, it
MUST mean that the zero or more following attributes up to the next
delimiter tag are operation attributes as defined in the model document.
When an job-attributes-tag occurs in the protocol, it MUST mean that the
zero or more following attributes up to the next delimiter tag are job
attributes or job template attributes as defined in the model document.
When a printer-attributes-tag occurs in the protocol, it MUST mean that
the zero or more following attributes up to the next delimiter tag are
printer attributes as defined in the model document. When an
unsupported-attributes-tag occurs in the protocol, it MUST mean that the
zero or more following attributes up to the next delimiter tag are
unsupported attributes as defined in the model document.

The operation-attributes-tag and end-of-attributes-tag MUST each occur
exactly once in an operation. The operation-attributes-tag MUST be the
first tag delimiter, and the end-of-attributes-tag MUST be the last tag
delimiter. If the operation has a document-content group, the document
data in that group MUST follow the end-of-attributes-tag.

Each of the other three xxx-attributes-tags defined above is OPTIONAL
in an operation and each MUST occur at most once in an operation, except
for job-attributes-tag in a Get-Jobs response which may occur zero or
more times.

The order and presence of delimiter tags for each operation request and
each operation response MUST be that defined in the model document. For
further details, see section 3.9 "(Attribute) Name" and 11 "Appendix A:
Protocol Examples".

A Printer MUST treat the reserved delimiter tags differently from

reserved value tags so that the Printer knows that there is an entire
attribute group that it doesn't understand as opposed to a single value
that it doesn't understand.

[3.7.2](#) **Value Tags**

The remaining tables show values for the value-tag, which is the first octet of an attribute. The value-tag specifies the type of the value of the attribute. The following table specifies the "out-of-band" values for the value-tag.

   Tag Value (Hex) Meaning

   0x10   unsupported
   0x11   reserved for future 'default'
   0x12   unknown
   0x13   no-value
   0x14-0x1F   reserved for future "out-of-band" values.

The "unsupported" value MUST be used in the attribute-sequence of an error response for those attributes which the printer does not support. The "default" value is reserved for future use of setting value back to their default value. The "unknown" value is used for the value of a supported attribute when its value is temporarily unknown. The "no-value" value is used for a supported attribute to which no value has been assigned, e.g. "job-k-octets-supported" has no value if an implementation supports this attribute, but an administrator has not configured the printer to have a limit.

The following table specifies the integer values for the value-tag:

   Tag Value (Hex) Meaning

   0x20   reserved
   0x21   integer
   0x22   boolean
   0x23   enum
   0x24-0x2F   reserved for future integer types

NOTE: 0x20 is reserved for "generic integer" if it should ever be needed.

The following table specifies the octetString values for the value-tag:

   Tag Value (Hex) Meaning

   0x30   octetString with an unspecified format
   0x31   dateTime
   0x32   resolution
   0x33   rangeOfInteger

```
0x34            reserved for collection (in the future)
0x35            textWithLanguage
0x36            nameWithLanguage
0x37-0x3F       reserved for future octetString types
```

The following table specifies the character-string values for the value-tag:

    Tag Value (Hex)   Meaning

    0x40   reserved
    0x41   textWithoutLanguage
    0x42   nameWithoutLanguage
    0x43   reserved
    0x44   keyword
    0x45   uri
    0x46   uriScheme
    0x47   charset
    0x48   naturalLanguage
    0x49   mimeMediaType
    0x4A-0x5F   reserved for future character string types

NOTE: 0x40 is reserved for "generic character-string" if it should ever
be needed.

NOTE: an attribute value always has a type, which is explicitly
specified by its tag; one such tag value is "nameWithoutLanguage".   An
attribute's name has an implicit type, which is keyword.

The values 0x60-0xFF are reserved for future types. There are no values
allocated for private extensions. A new type MUST be registered via the
type 2 registration process [ipp-mod].

The tag 0x7F is reserved for extending types beyond the 255 values
available with a single byte. A tag value of 0x7F MUST signify that the
first 4 bytes of the value field are interpreted as the tag value.
Note, this future extension doesn't affect parsers that are unaware of
this special tag. The tag is like any other unknown tag, and the value
length specifies the length of a value which contains a value that the
parser treats atomically. All these 4 byte tag values are currently
unallocated except that the values 0x40000000-0x7FFFFFFF are reserved
for experimental use.

3.8 **Name-Length**

The name-length field MUST consist of a SIGNED-SHORT. This field MUST
specify the number of octets in the name field which follows the name-
length field, excluding the two bytes of the name-length field.

If a name-length field has a value of zero, the following name field
MUST be empty, and the following value MUST be treated as an additional
value for the preceding attribute. Within an attribute-sequence, if two
attributes have the same name, the first occurrence MUST be ignored. The
zero-length name is the only mechanism for multi-valued attributes.

3.9 (Attribute) Name

Some operation elements are called parameters in the model document [ipp-mod]. They MUST be encoded in a special position and they MUST NOT appear as an operation attributes. These parameters are:

   - "version-number": The parameter named "version-number" in the IPP
     model document MUST become the "version-number" field in the
     operation layer request or response.

   - "operation-id": The parameter named "operation-id" in the IPP model
     document MUST become the "operation-id" field in the operation
     layer request.

   - "status-code": The parameter named "status-code" in the IPP model
     document MUST become the "status-code" field in the operation layer
     response.

   - "request-id": The parameter named "request-id" in the IPP model
      document MUST become the "request-id" field in the operation layer
      request or response.

All Printer and Job objects are identified by a Uniform Resource
Identifier (URI) [RFC2396] so that they can be persistently and
unambiguously referenced. The notion of a URI is a useful concept,
however, until the notion of URI is more stable (i.e., defined more
completely and deployed more widely), it is expected that the URIs used
for IPP objects will actually be URLs [RFC1738] [RFC1808]. Since every
URL is a specialized form of a URI, even though the more generic term
URI is used throughout the rest of this document, its usage is intended
to cover the more specific notion of URL as well.

Some operation elements are encoded twice, once as the request-URI on
the HTTP Request-Line and a second time as a REQUIRED operation
attribute in the application/ipp entity. These attributes are the
target URI for the operation and are called printer-uri and job-uri.
Note: The target URI is included twice in an operation referencing the
same IPP object, but the two URIs NEED NOT be literally identical. One
can be a relative URI and the other can be an absolute URI. HTTP/1.1
allows clients to generate and send a relative URI rather than an
absolute URI. A relative URI identifies a resource with the scope of
the HTTP server, but does not include scheme, host or port. The
following statements characterize how URLs should be used in the mapping
of IPP onto HTTP/1.1:

  1. Although potentially redundant, a client MUST supply the target of
     the operation both as an operation attribute and as a URI at the
     HTTP layer. The rationale for this decision is to maintain a
     consistent set of rules for mapping application/ipp to possibly
     many communication layers, even where URLs are not used as the
     addressing mechanism in the transport layer.
  2. Even though these two URLs might not be literally identical (one
     being relative and the other being absolute), they MUST both

reference the same IPP object.

   3. The URI in the HTTP layer is either relative or absolute and is
      used by the HTTP server to route the HTTP request to the correct
      resource relative to that HTTP server. The HTTP server need not be
      aware of the URI within the operation request.

4. Once the HTTP server resource begins to process the HTTP request,
    it might get the reference to the appropriate IPP Printer object
    from either the HTTP URI (using to the context of the HTTP server
    for relative URLs) or from the URI within the operation request;
    the choice is up to the implementation.
5. HTTP URIs can be relative or absolute, but the target URI in the
    operation MUST be an absolute URI.

The model document arranges the remaining attributes into groups for
each operation request and response. Each such group MUST be represented
in the protocol by an xxx-attribute-sequence preceded by the appropriate
xxx-attributes-tag (See the table below and section 11 "Appendix A:
Protocol Examples"). In addition, the order of these xxx-attributes-tags
and xxx-attribute-sequences in the protocol MUST be the same as in the
model document, but the order of attributes within each xxx-attribute-
sequence MUST be unspecified. The table below maps the model document
group name to xxx-attributes-sequence:

Model Document Group	xxx-attributes-sequence
Operation Attributes	operations-attributes-sequence
Job Template Attributes	job-attributes-sequence
Job Object Attributes	job-attributes-sequence
Unsupported Attributes	unsupported- attributes-sequence
Requested Attributes (Get-Job-Attributes)	job-attributes-sequence
Requested Attributes (Get-Printer-Attributes)	printer-attributes-sequence
Document Content	in a special position as described above

If an operation contains attributes from more than one job object (e.g.
Get-Jobs response), the attributes from each job object MUST be in a
separate job-attribute-sequence, such that the attributes from the ith
job object are in the ith job-attribute-sequence. See Section 11
"Appendix A: Protocol Examples" for table showing the application of the
rules above.

3.10 **Value Length**

Each attribute value MUST be preceded by a SIGNED-SHORT, which MUST
specify the number of octets in the value which follows this length,
exclusive of the two bytes specifying the length.

For any of the types represented by binary signed integers, the sender
MUST encode the value in exactly four octets.

For any of the types represented by character-strings, the sender MUST
encode the value with all the characters of the string and without any
padding characters.

If a value-tag contains an "out-of-band" value, such as "unsupported", the value-length MUST be 0 and the value empty . the value has no meaning when the value-tag has an "out-of-band" value.

3.11 (Attribute) Value

The syntax types and most of the details of their representation are defined in the IPP model document. The table below augments the information in the model document, and defines the syntax types from the model document in terms of the 5 basic types defined in section 3 "Encoding of the Operation Layer". The 5 types are US-ASCII-STRING, LOCALIZED-STRING, SIGNED-INTEGER, SIGNED-SHORT, SIGNED-BYTE, and OCTET-STRING.

Syntax of Attribute Encoding
Value

textWithoutLanguage, LOCALIZED-STRING.
nameWithoutLanguage

textWithLanguage   OCTET_STRING consisting of 4 fields:
                        a) a SIGNED-SHORT which is the number of octets
                           in the following field
                        b) a value of type natural-language,
                        c) a SIGNED-SHORT which is the number of octets
                           in the following field,
                        d) a value of type textWithoutLanguage.

                       The length of a textWithLanguage value MUST be 4
                       + the value of field a + the value of field c.

nameWithLanguage   OCTET_STRING consisting of 4 fields:
                        a) a SIGNED-SHORT which is the number of octets
                           in the following field
                        b) a value of type natural-language,
                        c) a SIGNED-SHORT which is the number of octets
                           in the following field
                        d) a value of type nameWithoutLanguage.

                       The length of a nameWithLanguage value MUST be 4
                       + the value of field a + the value of field c.

charset,   US-ASCII-STRING.
naturalLanguage,
mimeMediaType,
keyword, uri, and
uriScheme

boolean   SIGNED-BYTE where 0x00 is 'false' and 0x01 is
                       'true'.

integer and enum   a SIGNED-INTEGER.

dateTime   OCTET-STRING consisting of eleven octets whose
                       contents are defined by "DateAndTime" in RFC
                       1903 [RFC1903].

resolution   OCTET_STRING consisting of nine octets of 2
                       SIGNED-INTEGERs followed by a SIGNED-BYTE. The
                       first SIGNED-INTEGER contains the value of cross
                       feed direction resolution. The second SIGNED-
                       INTEGER contains the value of feed direction

resolution. The SIGNED-BYTE contains the units
value.

rangeOfInteger   Eight octets consisting of 2 SIGNED-INTEGERs.
                      The first SIGNED-INTEGER contains the lower
                      bound and the second SIGNED-INTEGER contains the

Syntax of Attribute Encoding
Value

                    upper bound.

1setOf X   Encoding according to the rules for an attribute
                     with more than 1 value. Each value X is encoded
                     according to the rules for encoding its type.

octetString   OCTET-STRING

The type of the value in the model document determines the encoding in
the value and the value of the value-tag.

3.12 Data

The data part MUST include any data required by the operation

4. Encoding of Transport Layer

HTTP/1.1 [RFC2616] is the transport layer for this protocol.

The operation layer has been designed with the assumption that the
transport layer contains the following information:

  - the URI of the target job or printer operation

  - the total length of the data in the operation layer, either as a
     single length or as a sequence of chunks each with a length.

It is REQUIRED that a printer implementation support HTTP over the IANA
assigned Well Known Port 631 (the IPP default port), though a printer
implementation may support HTTP over some other port as well.

Each HTTP operation MUST use the POST method where the request-URI is
the object target of the operation, and where the "Content-Type" of the
message-body in each request and response MUST be "application/ipp". The
message-body MUST contain the operation layer and MUST have the syntax
described in section 3.2 "Syntax of Encoding". A client implementation
MUST adhere to the rules for a client described for HTTP1.1 [RFC2616] .
A printer (server) implementation MUST adhere the rules for an origin
server described for HTTP1.1 [RFC2616].

An IPP server sends a response for each request that it receives. If an

IPP server detects an error, it MAY send a response before it has read
the entire request. If the HTTP layer of the IPP server completes
processing the HTTP headers successfully, it MAY send an intermediate
response, such as "100 Continue", with no IPP data before sending the
IPP response. A client MUST expect such a variety of responses from an

IPP server. For further information on HTTP/1.1, consult the HTTP
documents [RFC2616].

An HTTP server MUST support chunking for IPP requests, and an IPP client
MUST support chunking for IPP responses according to HTTP/1.1[RFC2616].
Note: this rule causes a conflict with non-compliant implementations of
HTTP/1.1 that don't support chunking for POST methods, and this rule may
cause a conflict with non-compliant implementations of HTTP/1.1 that
don't support chunking for CGI scripts

5. IPP URL Scheme

The IPP/1.1 document defines a new scheme 'ipp' as the value of a URL
that identifies either an IPP printer object or an IPP job object. The
IPP attributes using the 'ipp' scheme are specified below. Because the
HTTP layer does not support the 'ipp' scheme, a client MUST map 'ipp'
URLs to 'http' URLs, and then follows the HTTP [RFC2616][RFC2617] rules
for constructing a Request-Line and HTTP headers. The mapping is simple
because the 'ipp' scheme implies all of the same protocol semantics as
that of the 'http' scheme [RFC2616], except that it represents a print
service and the implicit (default) port number that clients use to
connect to a server is port 631.

In the remainder of this section the term 'ipp-URL' means a URL whose
scheme is 'ipp' and whose implicit (default) port is 631. The term
'http-URL' means a URL whose scheme is 'http', and the term 'https-URL'
means a URL whose scheme is 'https',

A client and an IPP object (i.e. the server) MUST support the ipp-URL
value in the following IPP attributes.
    job attributes:
        job-uri
        job-printer-uri
    printer attributes:
        printer-uri-supported
    operation attributes:
        job-uri
        printer-uri

Each of the above attributes identifies a printer or job object. The
ipp-URL is intended as the value of the attributes in this list, and for
no other attributes. All of these attributes have a syntax type of
'uri', but there are attributes with a syntax type of 'uri' that do not
use the 'ipp' scheme, e.g. 'job-more-info'.

If a printer registers its URL with a directory service, the printer
MUST register an ipp-URL.

User interfaces are beyond the scope of this document. But if software
exposes the ipp-URL values of any of the above five attributes to a
human user, it is REQUIRED that the human see the ipp-URL as is.

When a client sends a request, it MUST convert a target ipp-URL to a target http-URL for the HTTP layer according to the following rules:
  1. change the 'ipp' scheme to 'http'
  2. add an explicit port 631 if the URL does not contain an explicit port. Note: port 631 is the IANA assigned Well Known Port for the 'ipp' scheme.

The client MUST use the target http-URL in both the HTTP Request-Line and HTTP headers, as specified by HTTP[RFC2616][RFC2617] . However, the client MUST use the target ipp-URL for the value of the "printer-uri" or "job-uri" operation attribute within the application/ipp body of the request. The server MUST use the ipp-URL for the value of the "printer-uri", "job-uri" or "printer-uri-supported" attributes within the application/ipp body of the response.

For example, when an IPP client sends a request directly (i.e. no proxy) to an ipp-URL   "ipp://myhost.com/myprinter/myqueue", it opens a TCP connection to port 631 (the ipp implicit port) on the host "myhost.com" and sends the following data:

POST /myprinter/myqueue HTTP/1.1
Host: myhost.com:631
Content-type: application/ipp
Transfer-Encoding: chunked
...
"printer-uri" "ipp://myhost.com/myprinter/myqueue"
          (encoded in application/ipp message body)
...

As another example, when an IPP client sends the same request as above via a proxy "myproxy.com", it opens a TCP connection to the proxy port **8080 on the proxy host "myproxy.com" and sends the following data:**

POST http://myhost.com:631/myprinter/myqueue   HTTP/1.1
Host: myhost.com:631
Content-type: application/ipp
Transfer-Encoding: chunked
...
"printer-uri" "ipp://myhost.com/myprinter/myqueue"
          (encoded in application/ipp message body)
...

The proxy then connects to the IPP origin server with headers that are the same as the "no-proxy" example above.

6. Security Considerations

The IPP Model and Semantics document [ipp-mod] discusses high level security requirements (Client Authentication, Server Authentication and

Operation Privacy). Client Authentication is the mechanism by which the
client proves its identity to the server in a secure manner. Server
Authentication is the mechanism by which the server proves its identity
to the client in a secure manner. Operation Privacy is defined as a
mechanism for protecting operations from eavesdropping.

[6.1](#) **Security Conformance Requirements**

This section defines the security requirements for IPP clients and IPP objects.

[6.1.1](#) **Digest Authentication**

IPP clients MUST support:

  Digest Authentication [[RFC2617](#)].

    MD5 and MD5-sess MUST be implemented and supported.

    The Message Integrity feature NEED NOT be used.

IPP Printers SHOULD support:

  Digest Authentication [[RFC2617](#)].

    MD5 and MD5-sess MUST be implemented and supported.

    The Message Integrity feature NEED NOT be used.

The reasons that IPP Printers SHOULD (rather than MUST) support Digest Authentication are:

1.While Client Authentication is important, there is a certain class of printer devices where it does not make sense. Specifically, a low-end device with limited ROM space and low paper throughput may not need Client Authentication. This class of device typically requires firmware designers to make trade-offs between protocols and functionality to arrive at the lowest-cost solution possible. Factored into the designer.s decisions is not just the size of the code, but also the testing, maintenance, usefulness, and time-to-market impact for each feature delivered to the customer. Forcing such low-end devices to provide security in order to claim IPP/1.1 conformance would not make business sense and could potentially stall the adoption of the standard.

2.Print devices that have high-volume throughput and have available ROM space have a compelling argument to provide support for Client Authentication that safeguards the device from unauthorized access. These devices are prone to a high loss of consumables and paper if unauthorized access should occur.

6.1.2 Transport Layer Security (TLS)

IPP Printers SHOULD support Transport Layer Security (TLS) [RFC2246] for
Server Authentication and Operation Privacy. IPP Printers MAY also
support TLS for Client Authentication. If an IPP Printer supports TLS,
it MUST support the TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite as

mandated by RFC 2246 [RFC2246]. All other cipher suites are OPTIONAL.
An IPP Printer MAY support Basic Authentication (described in HTTP/1.1
[RFC2617]) for Client Authentication if the channel is secure. TLS with
the above mandated cipher suite can provide such a secure channel.

If a IPP client supports TLS, it MUST support the
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite as mandated by RFC 2246
[RFC2246]. All other cipher suites are OPTIONAL.

The IPP Model and Semantics document defines two printer attributes
("uri-authentication-supported" and "uri-security-supported") that the
client can use to discover the security policy of a printer. That
document also outlines IPP-specific security considerations and should
be the primary reference for security implications with regard to the
IPP protocol itself. For backward compatibility with IPP version 1.0,
IPP clients and printers may also support SSL3. This is in addition to
the security required in this document.

6.2 Using IPP with TLS

IPP/1.1 uses the "Upgrading to TLS Within HTTP/1.1" mechanism [http-
tls]. An initial IPP request never uses TLS. The client requests a
secure TLS connection by using the HTTP .Upgrade. header, while the
server agrees in the HTTP response. The switch to TLS occurs either
because the server grants the client.s request to upgrade to TLS, or a
server asks to switch to TLS in its response. Secure communication
begins with a server.s response to switch to TLS.

7. Interoperability with IPP/1.0 Implementations

It is beyond the scope of this specification to mandate conformance with
previous versions. IPP/1.1 was deliberately designed, however, to make
supporting previous versions easy. It is worth noting that, at the time
of composing this specification (1999), we would expect IPP/1.1 Printer
implementations to:

     understand any valid request in the format of IPP/1.0, or 1.1;

     respond appropriately with a response containing the same "version-
     number" parameter value used by the client in the request.

And we would expect IPP/1.1 clients to:

     understand any valid response in the format of IPP/1.0, or 1.1.

7.1 The "version-number" Parameter

The following are rules regarding the "version-number" parameter (see
[section 3.3](#)):

1. Clients MUST send requests containing a "version-number" parameter
   with a '1.1' value and SHOULD try supplying alternate version
   numbers if they receive a 'server-error-version-not-supported'
   error return in a response.

2. IPP objects MUST accept requests containing a "version-number"
   parameter with a '1.1' value (or reject the request for reasons
   other than 'server-error-version-not-supported').

3. It is recommended that IPP objects accept any request with the
   major version '1' (or reject the request for reasons other than
   'server-error-version-not-supported'). See [ipp-mod] "versions"
   sub-section.

4. In any case, security MUST NOT be compromised when a client
   supplies a lower "version-number" parameter in a request. For
   example, if an IPP/1.1 conforming Printer object accepts version
   '1.0' requests and is configured to enforce Digest Authentication,
   it MUST do the same for a version '1.0' request.

7.2 Security and URL Schemes

The following are rules regarding security, the "version-number"
parameter, and the URL scheme supplied in target attributes and
responses:

1. When a client supplies a request, the "printer-uri" or "job-uri"
   target operation attribute MUST have the same scheme as that
   indicated in one of the values of the "printer-uri-supported"
   Printer attribute.

2. When the server returns the "job-printer-uri" or "job-uri" Job
   Description attributes, it SHOULD return the same scheme ('ipp',
   'https', 'http', etc.) that the client supplied in the "printer-
   uri" or "job-uri" target operation attributes in the Get-Job-
   Attributes or Get-Jobs request, rather than the scheme used when
   the job was created. However, when a client requests job
   attributes using the Get-Job-Attributes or Get-Jobs operations, the
   jobs and job attributes that the server returns depends on: (1) the
   security in effect when the job was created, (2) the security in
   effect in the query request, and (3) the security policy in force.

3. It is recommended that if a server registers a non-secure ipp-URL
   with a directory service (see [IPP-MOD] "Generic Directory Schema"
   Appendix), then it also register an http-URL for interoperability
   with IPP/1.0 clients (see section 7).

4. In any case, security MUST NOT be compromised when a client

supplies an 'http' or other non-secure URL scheme in the target
"printer-uri" and "job-uri" operation attributes in a request.

8. **References**

[dpa] ISO/IEC 10175 Document Printing Application (DPA), June 1996.

[http-tls]R. Khare, S. Lawrence, "Upgrading to TLS Within HTTP/1.1",
      <draft-ietf-tls-http-upgrade-02>, June 1999.

[iana]IANA Registry of Coded Character Sets: ftp://ftp.isi.edu/in-
      notes/iana/assignments/character-sets.

[ipp-iig] Hastings, Tom, et al., "Internet Printing Protocol/1.1:
      Implementer's Guide", work in progress.

[ipp-mod] R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell,
      "Internet Printing Protocol/1.0: Model and Semantics", <draft-
      ietf-ipp-model-v11-03.txt>, June, 1999.

[ipp-pro] Herriot, R., Butler, S., Moore, P., Turner, R., "Internet
      Printing Protocol/1.1: Encoding and Transport", draft-ietf-ipp-
      protocol-v11-02-.txt, June 1999.

[RFC822] Crocker, D., "Standard for the Format of ARPA Internet Text
      Messages", RFC 822, August 1982.

[RFC1123] Braden, S., "Requirements for Internet Hosts - Application
      and Support", RFC 1123, October, 1989.

[RFC1179] McLaughlin, L. III, (editor), "Line Printer Daemon Protocol"
      RFC 1179, August 1990.

[RFC1543] Postel, J., "Instructions to RFC Authors", RFC 1543, October
      1993.

[RFC1738] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform
      Resource Locators (URL)", RFC 1738, December, 1994.

[RFC1759] Smith, R., Wright, F., Hastings, T., Zilles, S., and
      Gyllenskog, J., "Printer MIB", RFC 1759, March 1995.

[RFC1766] H. Alvestrand, " Tags for the Identification of Languages",
      RFC 1766, March 1995.

[RFC1808] R. Fielding, "Relative Uniform Resource Locators", RFC1808,
      June 1995.

[RFC1903] J. Case, et al. "Textual Conventions for Version 2 of the
      Simple Network Management Protocol (SNMPv2)", RFC 1903, January
      1996.

[RFC2046] N. Freed & N. Borenstein, Multipurpose Internet Mail
        Extensions (MIME) Part Two: Media Types. November 1996, RFC 2046.

[RFC2048] N. Freed, J. Klensin & J. Postel. Multipurpose Internet Mail
      Extension (MIME) Part Four: Registration Procedures. November
      1996 (Also BCP0013), RFC 2048.

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate
      Requirement Levels", RFC 2119 , March 1997.

[RFC2184] N. Freed, K. Moore, "MIME Parameter Value and Encoded Word
      Extensions: Character Sets, Languages, and Continuations", RFC
      2184, August 1997.

[RFC2234] D. Crocker et al., "Augmented BNF for Syntax Specifications:
      ABNF", RFC 2234. November 1997.

[RFC2246] T. Dierks et al., "The TLS Protocol", RFC 2246. January 1999.

[RFC2396] Berners-Lee, T., Fielding, R., Masinter, L., "Uniform
      Resource Identifiers (URI): Generic Syntax", RFC 2396, August
      1998.

[RFC2565] Herriot, R., Butler, S., Moore, P., Turner, R., "Internet
      Printing Protocol/1.0: Encoding and Transport", rfc 2565, April
      1999.

[RFC2566] R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell,
      "Internet Printing Protocol/1.0: Model and Semantics", rfc 2566,
      April, 1999.

[RFC2567] Wright, D., "Design Goals for an Internet Printing Protocol",
      RFC2567, April 1999.

[RFC2568] Zilles, S., "Rationale for the Structure and Model and
      Protocol for the Internet Printing Protocol", RC 2568,April 1999.

[RFC2569] Herriot, R., Hastings, T., Jacobs, N., Martin, J., "Mapping
      between LPD and IPP Protocols RFC 2569, April 1999.

[RFC2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P.
      Leach, T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1",
      RFC 2616, June 1999.

[RFC2069] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P.
      Leach, A. Luotonen, L. Stewart, "HTTP Authentication: Basic and
      Digest Access Authentication", RFC 2617, June 1999.

9. Author's Address

Robert Herriot (editor)   Paul Moore
Xerox Corporation   Microsoft
3400 **Hillview Ave., Bldg #1**   One Microsoft Way
Palo Alto, CA 94304   Redmond, WA 98053

Phone: 650-813-7696   Phone: 425-936-0908
Fax: 650-813-6860   Fax: 425-93MS-FAX
Email: Email: paulmo@microsoft.com
robert.herriot@pahv.xerox.com

Sylvan Butler   Randy Turner
Hewlett-Packard   2Wire, Inc.
11311 **Chinden Blvd.**   694 Tasman Dr.
Boise, ID 83714   Milpitas, CA 95035

Phone: 208-396-6000   Phone: 408-546-1273
Fax: 208-396-3457
Email: sbutler@boi.hp.com

                                   John Wenn
                                   Xerox Corporation
                                   737 Hawaii St
                                   El Segundo, CA 90245

IPP Mailing List: ipp@pwg.org   Phone: 310-333-5764
IPP Mailing List Subscription: Fax: 310-333-5514
ipp-request@pwg.org
IPP Web Page: Email: jwenn@cp10.es.xerox.com
http://www.pwg.org/ipp/

10. Other Participants:

Chuck Adams - Tektronix   Shivaun Albright - HP
Jeff Barnett - IBM   Ron Bergman - Dataproducts
Keith Carter - IBM   Angelo Caruso - Xerox
Rajesh Chawla - TR Computing   Josh Cohen - Microsoft
Solutions
Jeff Copeland - QMS   Andy Davidson - Tektronix
Roger deBry - IBM   Mabry Dozier - QMS
Lee Farrell - Canon   Steve Gebert - IBM
Sue Gleeson - Digital   Charles Gordon - Osicom
Brian Grimshaw - Apple   Jerry Hadsell - IBM
Richard Hart - Digital   Tom Hastings - Xerox
Stephen Holmstead   Zhi-Hong Huang - Zenographics
Scott Isaacson - Novell   Babek Jahromi - Microsoft
Swen Johnson - Xerox   David Kellerman - Northlake

```
                        Software
Robert Kline - TrueSpectra      Carl Kugler - IBM
Dave Kuntz - Hewlett-Packard    Takami Kurono - Brother
Rick Landau - Digital           Scott Lawrence - Agranot Systems
Greg LeClair - Epson            Harry Lewis - IBM
Tony Liao - Vivid Image         Roy Lomicka - Digital
Pete Loya - HP                  Ray Lutz - Cognisys
```

Mike MacKay - Novell, Inc.   David Manchala - Xerox
Carl-Uno Manros - Xerox   Jay Martin - Underscore
Larry Masinter - Xerox   Stan McConnell - Xerox
Ira McDonald - High North Inc.   Peter Michalek - Shinesoft
Tetsuya Morita - Ricoh   Yuichi Niwa - Ricoh
Pat Nogay - IBM   Ron Norton - Printronics
Bob Pentecost - Hewlett-Packard   Patrick Powell - Astart
                                  Technologies
Jeff Rackowitz - Intermec   Rob Rhoads - Intel
Xavier Riley - Xerox   Gary Roberts - Ricoh
David Roach - Unisys   Stuart Rowley - Kyocera
Richard Schneider - Epson   Kris Schoff - HP
Bob Setterbo - Adobe   Devon Taylor - Novell, Inc.
Mike Timperman - Lexmark   Shigern Ueda - Canon
Bob Von Andel - Allegro Software William Wagner - Osicom
Jim Walker - DAZEL   Chris Wellens - Interworking Labs
Rob Whittle - Novell, Inc.   Jasper Wong - Xionics
Don Wright - Lexmark   Rick Yardumian - Xerox
Lloyd Young - Lexmark   Atsushi Yuki - Kyocera
Peter Zehler - Xerox   Frank Zhao - Panasonic
Steve Zilles - Adobe

11. Appendix A: Protocol Examples

11.1 Print-Job Request

The following is an example of a Print-Job request with job-name,
copies, and sides specified. The "ipp-attribute-fidelity" attribute is
set to 'true' so that the print request will fail if the "copies" or the
"sides" attribute are not supported or their values are not supported.

Octets	Symbolic Value	Protocol field
0x0101	1.1	version-number
0x0002	Print-Job	operation-id
0x00000001	1	request-id
0x01	start operation-attributes	operation-attributes-tag
0x47	charset type	value-tag
0x0012		name-length
attributes-charset	attributes-charset	name
0x0008		value-length
us-ascii	US-ASCII	value
0x48	natural-language type	value-tag
0x001B		name-length
attributes-natural-language	attributes-natural-language	name
0x0005		value-length
en-us	en-US	value
0x45	uri type	value-tag
0x000B		name-length
printer-uri	printer-uri	name
0x0015		value-length
ipp://forest/pinetree	printer pinetree	value
0x42	nameWithoutLanguage type	value-tag
0x0008		name-length
job-name	job-name	name
0x0006		value-length
foobar	foobar	value
0x22	boolean type	value-tag
0x0016		name-length
ipp-attribute-fidelity	ipp-attribute-fidelity	name
0x0001		value-length
0x01	true	value
0x02	start job-attributes	job-attributes-tag
0x21	integer type	value-tag
0x0006		name-length
copies	copies	name
0x0004		value-length
0x00000014	20	value
0x44	keyword type	value-tag
0x0005		name-length
sides	sides	name
0x0013		value-length
two-sided-	two-sided-long-edge	value

```
long-edge
0x03            end-of-attributes         end-of-attributes-tag
%!PS...         <PostScript>              data
```

11.2 Print-Job Response (successful)

Here is an example of a successful Print-Job response to the previous
Print-Job request. The printer supported the "copies" and "sides"
attributes and their supplied values. The status code returned is
'successful-ok'.

Octets	Symbolic Value	Protocol field
0x0101	1.1	version-number
0x0000	successful-ok	status-code
0x00000001	1	request-id
0x01	start operation-attributes	operation-attributes-tag
0x47	charset type	value-tag
0x0012		name-length
attributes-charset	attributes-charset	name
0x0008		value-length
us-ascii	US-ASCII	value
0x48	natural-language type	value-tag
0x001B		name-length
attributes-natural-language	attributes-natural-language	name
0x0005		value-length
en-us	en-US	value
0x41	textWithoutLanguage type	value-tag
0x000E		name-length
status-message	status-message	name
0x000D		value-length
successful-ok	successful-ok	value
0x02	start job-attributes	job-attributes-tag
0x21	integer	value-tag
0x0006		name-length
job-id	job-id	name
0x0004		value-length
147	**147**	value
0x45	uri type	value-tag
0x0007		name-length
job-uri	job-uri	name
0x0019		value-length
ipp://forest/pinetree/123	job 123 on pinetree	value
0x23	enum type	value-tag
0x0009		name-length
job-state	job-state	name
0x0004		value-length
0x0003	pending	value
0x03	end-of-attributes	end-of-attributes-tag

11.3 **Print-Job Response (failure)**

Here is an example of an unsuccessful Print-Job response to the previous
Print-Job request. It fails because, in this case, the printer does not
support the "sides" attribute and because the value '20' for the
"copies" attribute is not supported. Therefore, no job is created, and
neither a "job-id" nor a "job-uri" operation attribute is returned. The

error code returned is 'client-error-attributes-or-values-not-supported'
(0x040B).

Octets	Symbolic Value	Protocol field
0x0101	1.1	version-number
0x040B	client-error-attributes-or-values-not-supported	status-code
0x00000001	1	request-id
0x01	start operation-attributes	operation-attribute tag
0x47	charset type	value-tag
0x0012		name-length
attributes-charset	attributes-charset	name
0x0008		value-length
us-ascii	US-ASCII	value
0x48	natural-language type	value-tag
0x001B		name-length
attributes-natural-language	attributes-natural-language	name
0x0005		value-length
en-us	en-US	value
0x41	textWithoutLanguage type	value-tag
0x000E		name-length
status-message	status-message	name
0x002F		value-length
client-error-attributes-or-values-not-supported	client-error-attributes-or-values-not-supported	value
0x05	start unsupported-attributes	unsupported-attributes tag
0x21	integer type	value-tag
0x0006		name-length
copies	copies	name
0x0004		value-length
0x00000014	20	value
0x10	unsupported (type)	value-tag
0x0005		name-length
sides	sides	name
0x0000		value-length
0x03	end-of-attributes	end-of-attributes-tag

11.4 **Print-Job Response (success with attributes ignored)**

Here is an example of a successful Print-Job response to a Print-Job

request like the previous Print-Job request, except that the value of
'ipp-attribute-fidelity' is false. The print request succeeds, even
though, in this case, the printer supports neither the "sides" attribute
nor the value '20' for the "copies" attribute. Therefore, a job is
created, and both a "job-id" and a "job-uri" operation attribute are
returned. The unsupported attributes are also returned in an Unsupported

Attributes Group. The error code returned is 'successful-ok-ignored-or-substituted-attributes' (0x0001).

Octets	Symbolic Value	Protocol field
0x0101	1.1	version-number
0x0001	successful-ok-ignored-or-substituted-attributes	status-code
0x00000001	1	request-id
0x01	start operation-attributes	operation-attributes-tag
0x47	charset type	value-tag
0x0012		name-length
attributes-charset	attributes-charset	name
0x0008		value-length
us-ascii	US-ASCII	value
0x48	natural-language type	value-tag
0x001B		name-length
attributes-natural-language	attributes-natural-language	name
0x0005		value-length
en-us	en-US	value
0x41	textWithoutLanguage type	value-tag
0x000E		name-length
status-message	status-message	name
0x002F		value-length
successful-ok-ignored-or-substituted-attributes	successful-ok-ignored-or-substituted-attributes	value
0x05	start unsupported-attributes	unsupported-attributes tag
0x21	integer type	value-tag
0x0006		name-length
copies	copies	name
0x0004		value-length
0x00000014	20	value
0x10	unsupported (type)	value-tag
0x0005		name-length
sides	sides	name
0x0000		value-length
0x02	start job-attributes	job-attributes-tag
0x21	integer	value-tag
0x0006		name-length
job-id	job-id	name
0x0004		value-length
147	**147**	value
0x45	uri type	value-tag
0x0007		name-length
job-uri	job-uri	name

```
0x0019                                    value-length
ipp://forest/pin   job 123 on pinetree    value
etree/123
0x23             enum   type              value-tag
0x0009                                    name-length
job-state        job-state               name
0x0004                                    value-length
```

Octets	Symbolic Value	Protocol field
0x0003	pending	value
0x03	end-of-attributes	end-of-attributes-tag

11.5 Print-URI Request

The following is an example of Print-URI request with copies and job-name parameters:

Octets	Symbolic Value	Protocol field
0x0101	1.1	version-number
0x0003	Print-URI	operation-id
0x00000001	1	request-id
0x01	start operation-attributes	operation-attributes-tag
0x47	charset type	value-tag
0x0012		name-length
attributes-charset	attributes-charset	name
0x0008		value-length
us-ascii	US-ASCII	value
0x48	natural-language type	value-tag
0x001B		name-length
attributes-natural-language	attributes-natural-language	name
0x0005		value-length
en-us	en-US	value
0x45	uri type	value-tag
0x000B		name-length
printer-uri	printer-uri	name
0x0015		value-length
ipp://forest/pinetree	printer pinetree	value
0x45	uri type	value-tag
0x000C		name-length
document-uri	document-uri	name
0x0011		value-length
ftp://foo.com/foo	ftp://foo.com/foo	value
0x42	nameWithoutLanguage type	value-tag
0x0008		name-length
job-name	job-name	name
0x0006		value-length
foobar	foobar	value
0x02	start job-attributes	job-attributes-tag
0x21	integer type	value-tag
0x0006		name-length
copies	copies	name
0x0004		value-length
0x00000001	1	value
0x03	end-of-attributes	end-of-attributes-tag

[11.6](#) Create-Job Request

The following is an example of Create-Job request with no parameters and
no attributes:

```
Octets          Symbolic Value            Protocol field
0x0101          1.1                       version-number
0x0005          Create-Job                operation-id
0x00000001      1                         request-id
0x01            start operation-attributes  operation-attributes-tag
0x47            charset type              value-tag
0x0012                                     name-length
attributes-     attributes-charset        name
charset
0x0008                                       value-length
us-ascii        US-ASCII                  value
0x48            natural-language type     value-tag
0x001B                                      name-length
attributes-     attributes-natural-language name
natural-
language
0x0005                                        value-length
en-us           en-US                     value
0x45            uri type                  value-tag
0x000B                                      name-length
printer-uri     printer-uri               name
0x0015                                       value-length
ipp://forest/p  printer pinetree          value
inetree
0x03            end-of-attributes         end-of-attributes-tag
```

11.7 Get-Jobs Request

The following is an example of Get-Jobs request with parameters but no
attributes:

Octets	Symbolic Value	Protocol field
0x0101	1.1	version-number
0x000A	Get-Jobs	operation-id
0x00000123	0x123	request-id
0x01	start operation-attributes	operation-attributes-tag
0x47	charset type	value-tag
0x0012		name-length
attributes-charset	attributes-charset	name
0x0008		value-length
us-ascii	US-ASCII	value
0x48	natural-language type	value-tag
0x001B		name-length
attributes-natural-language	attributes-natural-language	name
0x0005		value-length
en-us	en-US	value
0x45	uri type	value-tag
0x000B		name-length
printer-uri	printer-uri	name
0x0015		value-length
ipp://forest/pinetree	printer pinetree	value
0x21	integer type	value-tag
0x0005		name-length
limit	limit	name
0x0004		value-length
0x00000032	50	value
0x44	keyword type	value-tag
0x0014		name-length
requested-attributes	requested-attributes	name
0x0006		value-length
job-id	job-id	value
0x44	keyword type	value-tag
0x0000	additional value	name-length
0x0008		value-length
job-name	job-name	value
0x44	keyword type	value-tag
0x0000	additional value	name-length
0x000F		value-length
document-format	document-format	value
0x03	end-of-attributes	end-of-attributes-tag

11.8 Get-Jobs Response

The following is an of Get-Jobs response from previous request with 3
jobs. The Printer returns no information about the second job (because
of security reasons):

Octets	Symbolic Value	Protocol field
0x0101	1.1	version-number
0x0000	successful-ok	status-code
0x00000123	0x123	request-id (echoed back)
0x01	start operation-attributes	operation-attribute-tag
0x47	charset type	value-tag
0x0012		name-length
attributes-charset	attributes-charset	name
0x000A		value-length
ISO-8859-1	ISO-8859-1	value
0x48	natural-language type	value-tag
0x001B		name-length
attributes-natural-language	attributes-natural-language	name
0x0005		value-length
en-us	en-US	value
0x41	textWithoutLanguage type	value-tag
0x000E		name-length
status-message	status-message	name
0x000D		value-length
successful-ok	successful-ok	value
0x02	start job-attributes (1st object)	job-attributes-tag
0x21	integer type	value-tag
0x0006		name-length
job-id	job-id	name
0x0004		value-length
147	**147**	value
0x36	nameWithLanguage	value-tag
0x0008		name-length
job-name	job-name	name
0x000C		value-length
0x0005		sub-value-length
fr-ca	fr-CA	value
0x0003		sub-value-length
fou	fou	name
0x02	start job-attributes (2nd object)	job-attributes-tag
0x02	start job-attributes (3rd object)	job-attributes-tag
0x21	integer type	value-tag
0x0006		name-length
job-id	job-id	name
0x0004		value-length

```
148                 149                     value
0x36                nameWithLanguage        value-tag
0x0008                                      name-length
job-name            job-name                name
0x0012                                      value-length
0x0005                                      sub-value-length
de-CH               de-CH                   value
```

Octets	Symbolic Value	Protocol field
0x0009		sub-value-length
isch guet	isch guet	name
0x03	end-of-attributes	end-of-attributes-tag

[12]. [Appendix C]: Registration of MIME Media Type Information for "application/ipp"

This appendix contains the information that IANA requires for registering a MIME media type. The information following this paragraph will be forwarded to IANA to register application/ipp whose contents are defined in [Section 3] "Encoding of the Operation Layer" in this document:

MIME type name: application

MIME subtype name: ipp

A Content-Type of "application/ipp" indicates an Internet Printing Protocol message body (request or response). Currently there is one version: IPP/1.1, whose syntax is described in [Section 3] "Encoding of the Operation Layer" of [ipp-pro], and whose semantics are described in [ipp-mod].

Required parameters: none

Optional parameters: none

Encoding considerations:

IPP/1.1 protocol requests/responses MAY contain long lines and ALWAYS contain binary data (for example attribute value lengths).

Security considerations:

IPP/1.1 protocol requests/responses do not introduce any security risks not already inherent in the underlying transport protocols. Protocol mixed-version interworking rules in [ipp-mod] as well as protocol encoding rules in [ipp-pro] are complete and unambiguous.

Interoperability considerations:

IPP/1.1 requests (generated by clients) and responses (generated by servers) MUST comply with all conformance requirements imposed by the normative specifications [ipp-mod] and [ipp-pro]. Protocol encoding rules specified in [ipp-pro] are comprehensive, so that interoperability between conforming implementations is guaranteed (although support for specific optional features is not ensured). Both the "charset" and

"natural-language" of all IPP/1.1 attribute values which are a
LOCALIZED-STRING are explicit within IPP protocol requests/responses
(without recourse to any external information in HTTP, SMTP, or other
message transport headers).

Published specifications:

[ipp-mod]   Isaacson, S., deBry, R., Hastings, T., Herriot, R.,
       Powell, P., "Internet Printing Protocol/1.1: Model and
       Semantics" draft-ietf-ipp-model-v11-03.txt, June, 1999.

[ipp-pro]   Herriot, R., Butler, S., Moore, P., Turner, R.,
       "Internet Printing Protocol/1.1: Encoding and Transport", draft-
       ietf-ipp-protocol-v11-02.txt, June, 1999.

Applications which use this media type:

Internet Printing Protocol (IPP) print clients and print servers,
communicating using HTTP/1.1 (see [IPP-PRO]), SMTP/ESMTP, FTP, or other
transport protocol. Messages of type "application/ipp" are self-
contained and transport-independent, including "charset" and "natural-
language" context for any LOCALIZED-STRING value.

Person & email address to contact for further information:

Tom Hastings
Xerox Corporation
 737 Hawaii St. ESAE-231
El Segundo, CA

Phone: 310-333-6413
Fax: 310-333-5514
Email: thastings@cp10.es.xerox.com

or

Robert Herriot
Xerox Corporation
3400 Hillview Ave., Bldg #1
Palo Alto, CA 94304

Phone: 650-813-7696
Fax: 650-813-6860
Email: robert.herriot@pahv.xerox.com

Intended usage:

COMMON

13. Appendix D: Changes from IPP/1.0

IPP/1.1 is identical to IPP/1.0 [RFC2565] with the follow changes:

1.Attributes values that identify a printer or job object use a new
  'ipp' scheme. The 'http' and 'https' schemes are supported only for

backward compatibility. See [section 5](#).

2.Clients MUST support of Digest Authentication, IPP Printers SHOULD
   support Digest Authentication. See [Section 6.1.1](#)

3.TLS is recommended for channel security. In addition, SSL3 may be
   supported for backward compatibility. See Section 6.1.2

4.It is recommended that IPP/1.1 objects accept any request with major
   version number '1'. See section 7.1.

5.IPP objects SHOULD return the URL scheme requested for "job-printer-
   uri" and "job-uri" Job Attributes, rather than the URL scheme used to
   create the job.   See section 7.2

14. Full Copyright Statement

The IETF takes no position regarding the validity or scope of any
intellectual property or other rights that might be claimed to pertain
to the implementation or use of the technology described in this
document or the extent to which any license under such rights might or
might not be available; neither does it represent that it has made any
effort to identify any such rights. Information on the IETF's
procedures with respect to rights in standards-track and standards-
related documentation can be found in BCP-11[BCP-11]. Copies of claims
of rights made available for publication and any assurances of licenses
to be made available, or the result of an attempt made to obtain a
general license or permission for the use of such proprietary rights by
implementers or users of this specification can be obtained from the
IETF Secretariat.

The IETF invites any interested party to bring to its attention any
copyrights, patents or patent applications, or other proprietary rights
which may cover technology that may be required to practice this
standard. Please address the information to the IETF Executive
Director.

Copyright (C)The Internet Society (1999). All Rights Reserved

This document and translations of it may be copied and furnished to
others, and derivative works that comment on or otherwise explain it or
assist in its implementation may be prepared, copied, published and
distributed, in whole or in part, without restriction of any kind,
provided that the above copyright notice and this paragraph are included
on all such copies and derivative works. However, this document itself
may not be modified in any way, such as by removing the copyright notice
or references to the Internet Society or other Internet organizations,
except as needed for the purpose of developing Internet standards in
which case the procedures for copyrights defined in the Internet
Standards process must be followed, or as required to translate it into
languages other than English.

The limited permissions granted above are perpetual and will not be

revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS
IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK
FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT

LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR
FITNESS FOR A PARTICULAR PURPOSE.