IPP WG
Internet-Draft
Obsoletes: 2910,3382 (if approved)
Intended status: Standards Track
Expires: January 22, 2016

M. Sweet
Apple Inc.
I. McDonald
High North, Inc.

July 21, 2015

Internet Printing Protocol/1.1: Encoding and Transport
draft-sweet-rfc2910bis-05

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".

Editor's Note

This draft is being submitted as an AD-sponsored replacement of RFCs 2910 and 3382, with drafts being reviewed and edited by the IEEE-ISTO's Printer Working Group IPP WG. The initial goal is to have an clean version of IPP/1.1 as an IETF Proposed Standard. The long-term goal is to advance IPP/1.1 to IETF Internet Standard.

Status of This Memo

Copyright Notice

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 request and response. All IPP implementations MUST support HTTP/1.1, the relevant parts of which are described in the following RFCs:

- Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing [RFC7230]

- Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content [RFC7231]

- Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests [RFC7232]

- Hypertext Transfer Protocol (HTTP/1.1): Caching [RFC7234]

- Hypertext Transfer Protocol (HTTP/1.1): Authentication [RFC7235]

- The 'Basic' HTTP Authentication Scheme [HTTPAUTH-BASIC]

- HTTP Digest Access Authentication [HTTPAUTH-DIGEST]

IPP implementations MAY support HTTP/2 which is described in the following RFCs:

- Hypertext Transfer Protocol Version 2 (HTTP/2) [RFC7540]

- HPACK - Header Compression for HTTP/2 [RFC7541]

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" [RFC2911bis] defines the semantics of such a message body and the supported values. This document specifies the encoding of an IPP operation. The aforementioned document [RFC2911bis] is henceforth referred to as the "IPP model document" or simply "model document."

2. Conventions Used in This Document

2.1. Requirements Language

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

2.2. Printing Terminology

Client: Initiator of outgoing IPP session requests and sender of outgoing IPP operation requests (Hypertext Transfer Protocol -- HTTP/1.1 [RFC7230] User Agent).

Document: An object created and managed by a Printer that contains the description, processing, and status information. A Document object may have attached data and is bound to a single Job.

'ipp' URI: An IPP URI as defined in [RFC3510].

'ipps' URI: An IPP URI as defined in [RFC7472].

Job: An object created and managed by a Printer that contains description, processing, and status information. The Job also contains zero or more Document objects.

Logical Device: A print server, software service, or gateway that processes Jobs and either forwards or stores the processed Job or uses one or more Physical Devices to render output.

Output Device: A single Logical or Physical Device.

Physical Device: A hardware implementation of an endpoint device, e.g., a marking engine, a fax modem, etc.

Printer: Listener for incoming IPP session requests and receiver of incoming IPP operation requests (Hypertext Transfer Protocol -- HTTP/1.1 [RFC7230] Server) that represents one or more Physical Devices or a Logical Device.

2.3. Abbreviations

ABNF: Augmented Backus-Naur Form [RFC5234]

ASCII: American Standard Code for Information Interchange [ASCII]

HTTP: HyperText Transfer Protocol [RFC7230]

HTTPS: HTTP over TLS [RFC2818]

IANA: Internet Assigned Numbers Authority

IEEE: Institute of Electrical and Electronics Engineers

IESG: Internet Engineering Steering Group

IPP: Internet Printing Protocol (this document and [PWG5100.12])

ISTO: IEEE Industry Standards and Technology Organization

LPD: Line Printer Daemon Protocol [RFC1179]

PWG: IEEE-ISTO Printer Working Group [1]

RFC: Request for Comments

TCP: Transmission Control Protocol [RFC793]

TLS: Transport Layer Security [RFC5246]

URI: Uniform Resource Identifier [RFC3986]

URL: Uniform Resource Locator [RFC3986]

UTF-8: Unicode Transformation Format - 8-bit [RFC3629]

3. Encoding of the Operation Layer

The operation layer is the message body part of the HTTP request or response and it MUST contain a single IPP operation request or IPP 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 value fields and the request-id.

The following two sections present the encoding of the operation layer in two ways:

- informally through pictures and description

- formally through Augmented Backus-Naur Form (ABNF), as specified by RFC 5234 [RFC5234]

An operation request or response MUST use the encoding described in these two sections.

3.1. Picture of the Encoding

3.1.1. Request and Response

An operation request or response is encoded as follows:

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

Figure 1: IPP Message Format

The first three fields in the above diagram contain the value of attributes described in Section 3.1.1 of the Model document.

The fourth field is the "attribute-group" field, and it occurs 0 or more times. Each "attribute-group" field represents a single group of attributes, such as an Operation Attributes group or a Job Attributes group (see the Model document). The IPP model document specifies the required attribute groups and their order for each operation request and response.

The "end-of-attributes-tag" field is always present, even when the "data" is not present. The Model document specifies for each

operation request and response whether the "data" field is present or absent.

3.1.2. Attribute Group

Each "attribute-group" field is encoded as follows:

```
-----------------------------------------------
|           begin-attribute-group-tag         |  1 byte
----------------------------------------------------------
|                   attribute                 |  p bytes |- 0 or more
----------------------------------------------------------
```

Figure 2: Attribute Group Encoding

An "attribute-group" field contains zero or more "attribute" fields.

Note, the values of the "begin-attribute-group-tag" field and the "end-of-attributes-tag" field are called "delimiter-tags".

3.1.3. Attribute

An "attribute" field is encoded as follows:

```
-----------------------------------------------
|          attribute-with-one-value           |  q bytes
----------------------------------------------------------
|             additional-value                |  r bytes |- 0 or more
----------------------------------------------------------
```

Figure 3: Attribute Encoding

When an attribute is single valued (e.g. "copies" with value of 10) or multi-valued with one value (e.g. "sides-supported" with just the value 'one-sided') it is encoded with just an "attribute-with-one-value" field. When an attribute is multi-valued with n values (e.g. "sides-supported" with the values 'one-sided' and 'two-sided-long-edge'), it is encoded with an "attribute-with-one-value" field followed by n-1 "additional-value" fields.

3.1.4. Attribute-with-one-value

Each "attribute-with-one-value" field is encoded as follows:

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

Figure 4: Single Value Attribute Encoding

An "attribute-with-one-value" field is encoded with five subfields:

- The "value-tag" field specifies the attribute syntax, e.g. 0x44 for the attribute syntax 'keyword'.

- The "name-length" field specifies the length of the "name" field in bytes, e.g. u in the above diagram or 15 for the name "sides-supported".

- The "name" field contains the textual name of the attribute, e.g. "sides-supported".

- The "value-length" field specifies the length of the "value" field in bytes, e.g. v in the above diagram or 9 for the (keyword) value 'one-sided'.

- The "value" field contains the value of the attribute, e.g. the textual value 'one-sided'.

3.1.5. Additional-value

Each "additional-value" field is encoded as follows:

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

Figure 5: Additional Attribute Value Encoding

An "additional-value" is encoded with four subfields:

- o The "value-tag" field specifies the attribute syntax, e.g. 0x44 for the attribute syntax 'keyword'.

- o The "name-length" field has the value of 0 in order to signify that it is an "additional-value". The value of the "name-length" field distinguishes an "additional-value" field ("name-length" is 0) from an "attribute-with-one-value" field ("name-length" is not 0).

- o The "value-length" field specifies the length of the "value" field in bytes, e.g. w in the above diagram or 19 for the (keyword) value 'two-sided-long-edge'.

- o The "value" field contains the value of the attribute, e.g. the textual value 'two-sided-long-edge'.

3.1.6. Collection Attribute

Collection attributes create a named group containing related "member" attributes. The "attribute-with-one-value" field for a collection attribute is encoded as follows:

```
   -----------------------------------------------
   |          value-tag (value is 0x34)          |   1 byte
   -----------------------------------------------
   |          name-length (value is u)           |   2 bytes
   -----------------------------------------------
   |                     name                    |   u bytes
   -----------------------------------------------
   |        value-length (value is 0x0000)       |   2 bytes
   -----------------------------------------------------------
   |               member-attribute              |   q bytes |-0 or more
   -----------------------------------------------------------
   |        end-value-tag (value is 0x37)        |   1 byte
   -----------------------------------------------
   |      end-name-length (value is 0x0000)      |   2 bytes
   -----------------------------------------------
   |      end-value-length (value is 0x0000)     |   2 bytes
   -----------------------------------------------
```

Figure 6: Collection Attribute Encoding

Collection attribute is encoded with eight subfields:

- o The "value-tag" field specifies the start attribute syntax: 0x34 for the attribute syntax 'begCollection'.

- The "name-length" field specifies the length of the "name" field in bytes, e.g. u in the above diagram or 9 for the name "media-col". Additional collection attribute values use a name length of 0x0000.

- The "name" field contains the textual name of the attribute, e.g. "media-col".

- The "value-length" field specifies a length of 0x0000.

- The "member-attribute" field contains member attributes encoded as defined in Section 3.1.7.

- The "end-value-tag" field specifies the end attribute syntax: 0x37 for the attribute syntax 'endCollection'.

- The "end-name-length" field specifies a length of 0x0000.

- The "end-value-length" field specifies a length of 0x0000.

3.1.7. Member Attributes

Each "member-attribute" field is encoded as follows:

```
   -----------------------------------------------
   |          value-tag (value is 0x4A)          |   1 byte
   -----------------------------------------------
   |        name-length (value is 0x0000)        |   2 bytes
   -----------------------------------------------
   |          value-length (value is w)          |   2 bytes
   -----------------------------------------------
   |             value (member-name)             |   w bytes
   -----------------------------------------------
   |               member-value-tag              |   1 byte
   -----------------------------------------------
   |        name-length (value is 0x0000)        |   2 bytes
   -----------------------------------------------
   |      member-value-length (value is x)       |   2 bytes
   -----------------------------------------------
   |                member-value                 |   x bytes
   -----------------------------------------------
```

Figure 7: Member Attribute Encoding

A "member-attribute" is encoded with eight subfields:

- The "value-tag" field specifies 0x4A for the attribute syntax 'memberAttrName'.

- The "name-length" field has the value of 0 in order to signify that it is a "member-attribute" contained in the collection.

- The "value-length" field specifies the length of the "value" field in bytes, e.g. w in the above diagram or 10 for the member attribute name 'media-type'. Additional member attribute values are specifies using a value length of 0.

- The "value" field contains the name of the member attribute, e.g. the textual value 'media-type'.

- The "member-value-tag" field specifies the attribute syntax for the member attribute, e.g. 0x44 for the attribute syntax 'keyword'.

- The second "name-length" field has the value of 0 in order to signify that it is a "member-attribute" contained in the collection.

- The "member-value-length" field specifies the length of the member attribute value, e.g. x in the above diagram or 10 for the value 'stationery'.

- The "member-value" field contains the value of the attribute, e.g. the textual value 'stationery'.

3.1.8. Alternative Picture of the Encoding of a Request Or a Response

From the standpoint of a parser that performs an action based on a "tag" value, 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          |   1 byte   - required
   -----------------------------------------------
   |                     data                    |   y bytes  - optional
   -----------------------------------------------
```

Figure 8: Encoding Based On Value Tags

The following show what fields the parser would expect after each type of "tag":

o "begin-attribute-group-tag": expect zero or more "attribute" fields

o "value-tag": expect the remainder of an "attribute-with-one-value" or an "additional-value".

o "end-of-attributes-tag": expect that "attribute" fields are complete and there is optional "data"

3.2. Syntax of Encoding

The syntax below is ABNF [RFC5234] 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
                  *attribute-group end-of-attributes-tag data
   ipp-response = version-number status-code request-id
                  *attribute-group end-of-attributes-tag  data

   version-number       = major-version-number minor-version-number
   major-version-number = SIGNED-BYTE
   minor-version-number = SIGNED-BYTE

   operation-id = SIGNED-SHORT     ; mapping from model
   status-code  = SIGNED-SHORT     ; mapping from model
   request-id   = SIGNED-INTEGER   ; whose value is > 0

   attribute-group          = begin-attribute-group-tag *attribute
   attribute                = attribute-with-one-value *additional-value
   attribute-with-one-value = value-tag name-length name
                              value-length value
   additional-value         = 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
   value-tag                 = %x10-FF           ; see section 3.7.2
   begin-attribute-group-tag = %x00-02 / %04-0F  ; see section 3.7.1
   end-of-attributes-tag     = %x03              ; tag of 3
                                                 ; see section 3.7.1

   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
```

Figure 9: ABNF of IPP Message Format

The syntax below defines additional terms that are referenced in this document. This syntax provides an alternate grouping of the delimiter tags.

```
   delimiter-tag = begin-attribute-group-tag  / ; see section 3.7.1
             end-of-attributes-tag
   delimiter-tag = %x00-0F                      ; see section 3.7.1
   begin-attribute-group-tag = %x00 / operation-attributes-tag /
      job-attributes-tag / printer-attributes-tag /
      unsupported-attributes-tag /  %x06-0F
   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
```

Figure 10: ABNF for Attribute Group Tags

3.3. Attribute-group

Each "attribute-group" field MUST be encoded with the "begin-attribute-group-tag" field followed by zero or more "attribute" sub-fields.

The table below maps the model document group name to value of the "begin-attribute-group-tag" field:

Model Document Group	"begin-attribute-group-tag" field values
Operation Attributes	"operations-attributes-tag"
Job Template Attributes	"job-attributes-tag"
Job Object Attributes	"job-attributes-tag"
Unsupported Attributes	"unsupported-attributes-tag"
Requested Attributes	(Get-Job-Attributes) "job-attributes-tag"
Requested Attributes	(Get-Printer-Attributes)"printer-attributes-tag"
Document Content	in a special position as described above

Table 1: Group Values

For each operation request and response, the model document prescribes the required and optional attribute groups, along with their order. Within each attribute group, the model document prescribes the required and optional attributes, along with their order.

When the Model document requires an attribute group in a request or response and the attribute group contains zero attributes, a request or response SHOULD encode the attribute group with the "begin-attribute-group-tag" field followed by zero "attribute" fields. For example, if the Client requests a single unsupported attribute with the Get-Printer-Attributes operation, the Printer MUST return no "attribute" fields, and it SHOULD return a "begin-attribute-group-tag" field for the Printer Attributes Group. The Unsupported Attributes group is not such an example. According to the model document, the Unsupported Attributes Group SHOULD be present only if the unsupported attributes group contains at least one attribute.

A receiver of a request MUST be able to process the following as equivalent empty attribute groups:

a. A "begin-attribute-group-tag" field with zero following "attribute" fields.

b. An expected but missing "begin-attribute-group-tag" field.

When the Model document requires a sequence of an unknown number of attribute groups, each of the same type, the encoding MUST contain one "begin-attribute-group-tag" field for each attribute group even when an "attribute-group" field contains zero "attribute" sub-fields. For example, for the Get-Jobs operation may return zero attributes for some jobs and not others. The "begin-attribute-group-tag" field followed by zero "attribute" fields tells the recipient that there is a Job in queue for which no information is available except that it is in the queue.

3.4. Required Parameters

Some operation elements are called parameters in the model document [RFC2911bis]. They MUST be encoded in a special position and they MUST NOT appear as operation attributes. These parameters are described in the subsections below.

3.4.1. Version-number

The "version-number" field consists of a major and minor version-number, each of which is represented by a SIGNED-BYTE. The major version-number is the first byte of the encoding and the minor version-number is the second byte of the encoding. The protocol described in this document has a major version-number of 1 (0x01) and a minor version-number of 1 (0x01). The ABNF for these two bytes is %x01.01.

Note: See Section 9 for more information on the "version-number" field and IPP version numbers.

3.4.2. Operation-id

The "operation-id" field contains an operation-id value as defined in the model document. The value is encoded as a SIGNED-SHORT and is located in the third and fourth bytes of the encoding of an operation request.

3.4.3. Status-code

The "status-code" field contains a status-code value as defined in the model document. The value is encoded as a SIGNED-SHORT and is located in the third and fourth bytes of the encoding of an operation response.

If an IPP status-code is returned, then the HTTP Status-Code MUST be 200 (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.

3.4.4. Request-id

The "request-id" field contains the request-id value as defined in the model document. The value is encoded as a SIGNED-INTEGER and is located in the fifth through eighth bytes of the encoding.

3.5. 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

Tags are part of the IANA IPP registry [IANA-IPP]

3.5.1. Delimiter Tags

The following table specifies the values for the delimiter tags defined in this document:

Tag Value (Hex)	Meaning
0x00	reserved for definition in a future standards track document
0x01	"operation-attributes-tag"
0x02	"job-attributes-tag"
0x03	"end-of-attributes-tag"
0x04	"printer-attributes-tag"
0x05	"unsupported-attributes-tag"
0x06-0x0f	reserved for future delimiters in standards track documents

Table 2: Delimiter Tags

When a "begin-attribute-group-tag" field occurs in the protocol, it means that zero or more following attributes up to the next delimiter tag are attributes belonging to the attribute group specified by the value of the "begin-attribute-group-tag". For example, if the value

of "begin-attribute-group-tag" is 0x01, the following attributes are members of the Operations Attributes group.

The "end-of-attributes-tag" (value 0x03) MUST occur exactly once in an operation and MUST be the last "delimiter-tag". If the operation has a document-content group, the document data in that group follows the "end-of-attributes-tag".

The order and presence of "attribute-group" fields (whose beginning is marked by the "begin-attribute-group-tag" subfield) for each operation request and each operation response MUST be that defined in the model document. For further details, see Section 3.7 "(Attribute) Name" and 13 "Appendix A: Protocol Examples".

A Printer MUST treat a "delimiter-tag" (values from 0x00 through 0x0F) differently from a "value-tag" (values from 0x10 through 0xFF) 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.

3.5.2. Value Tags

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

The following table specifies the "out-of-band" values for the "value-tag" field defined in this document:

Tag Value (Hex)	Meaning
0x10	unsupported
0x11	reserved for 'default' for definition in a future standards track document
0x12	unknown
0x13	no-value
0x14-0x1F	reserved for "out-of-band" values in future standards track documents.

Table 3: Out of Band Values

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

Tag Value (Hex)	Meaning
0x20	reserved for definition in a future standards track document
0x21	integer
0x22	boolean
0x23	enum
0x24-0x2F	reserved for integer types for definition in future standards track documents

Table 4: Integer Tags

Note: 0x20 is reserved for "generic integer" if it ever is needed.

The following table specifies the octetString values for the "value-tag" field defined in this document:

Tag Value (Hex)	Meaning
0x30	octetString with an unspecified format
0x31	dateTime
0x32	resolution
0x33	rangeOfInteger
0x34	begCollection
0x35	textWithLanguage
0x36	nameWithLanguage
0x37	endCollection
0x38-0x3F	reserved for octetString type definitions in future standards track documents

Table 5: octetString Tags

The following table specifies the character-string values for the "value-tag" field defined in this document:

Tag Value (Hex)	Meaning
0x40	reserved for definition in a future standards track document
0x41	textWithoutLanguage
0x42	nameWithoutLanguage
0x43	reserved for definition in a future standards track document
0x44	keyword
0x45	uri
0x46	uriScheme
0x47	charset
0x48	naturalLanguage
0x49	mimeMediaType
0x4A	memberAttrName
0x4B-0x5F	reserved for character string type definitions in future standards track documents

Table 6: String Tags

Note: 0x40 is reserved for "generic character-string" if it ever is 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 type definitions in standards track documents.

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. Values from 0x00000000 to 0x3FFFFFFF are reserved for definition in future standards track documents. The values 0x40000000 to 0x7FFFFFFF are reserved for vendor extensions.

3.6. Name-Length

The "name-length" field consists of a SIGNED-SHORT and specifies the number of octets in the immediately following "name" field. The value of this field excludes the two bytes of the "name-length" field. For example, if the "name" field contains "sides", the value of this field is 5.

If a "name-length" field has a value of zero, the following "name" field is empty and the following value is treated as an additional value for the attribute encoded in the nearest preceding "attribute-with-one-value" field. Within an attribute group, if two or more attributes have the same name, the attribute group is mal-formed (see Section ?? of [RFC2911bis]). The zero-length name is the only mechanism for multi-valued attributes.

3.7. (Attribute) Name

The "name" field contains the name of an attribute. The model document [RFC2911bis] specifies such names.

3.8. Value Length

The "value-length" field consists of a SIGNED-SHORT which specifies the number of octets in the immediately following "value" field. The value of this field excludes the two bytes of the "value-length" field. For example, if the "value" field contains the keyword (text) value 'one-sided', the value of this field is 9.

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 binary signed bytes, the sender MUST encode the value in exactly one octet.

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.

For "out-of-band" "value-tag" fields defined in this document, such as "unsupported", the "value-length" MUST be 0 and the "value" empty; the "value" has no meaning when the "value-tag" has one of these "out-of-band" values. For future "out-of-band" "value-tag" fields, the same rule holds unless the definition explicitly states that the "value-length" MAY be non-zero and the "value" non-empty

3.9. (Attribute) Value

The syntax types (specified by the "value-tag" field) and most of the details of the representation of attribute values 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. The 5 types are US-ASCII-STRING, LOCALIZED-STRING, SIGNED-INTEGER, SIGNED-SHORT, SIGNED-BYTE, and OCTET-STRING.

Syntax of Attribute Value	Encoding
textWithoutLanguage, nameWithoutLanguage	LOCALIZED-STRING
textWithLanguage	OCTET-STRING consisting of 4 fields: a SIGNED-SHORT which is the number of octets in the following field, a value of type natural-language, a SIGNED-SHORT which is the number of octets in the following field, and 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 SIGNED-SHORT which is the number of octets in the following field, a value of type natural-language, a SIGNED-SHORT which is the number of octets in the following field, and 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, naturalLanguage, mimeMediaType, keyword, uri, and uriScheme	US-ASCII-STRING
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 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
collection	Encoding as defined in Section 3.1.6.

Table 7: Attribute Value Encoding

The attribute syntax type of the value determines its encoding and the value of its "value-tag".

3.10. Data

The "data" field MUST include any data required by the operation.

4. Encoding of Transport Layer

HTTP/1.1 [RFC7230] is the REQUIRED transport layer for this protocol. HTTP/2 [RFC7540] is an OPTIONAL 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 [RFC7230]. A Printer (server) implementation MUST adhere the rules for an origin server described for HTTP1.1 [RFC7230].

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, consult the HTTP documents [RFC7230].

An HTTP/1.1 server MUST support chunking for IPP requests, and an IPP Client MUST support chunking for IPP responses according to HTTP/1.1 [RFC7230]. 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

4.1. Printer-uri and job-uri

All Printer and Job objects are identified by a Uniform Resource Identifier (URI) [RFC3986] so that they can be persistently and unambiguously referenced.

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 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 URIs are used in the mapping of IPP onto HTTP:

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 URIs are not used as the addressing mechanism in the transport layer.

2. Even though these two URIs 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.

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 URIs) 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.

5. IPP URI Schemes

The IPP URI schemes are 'ipp' [RFC3510] and 'ipps' [RFC7472]. A Client and an IPP object (i.e. the server) MUST support the ipp-URI 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-URI 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 URI with a directory service, the Printer MUST register an ipp-URI.

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

When a Client sends a request, it MUST convert a target ipp-URI to a target http-URL (or ipps-URI to a target https-URI) for the HTTP layer according to the following rules:

1. change the 'ipp' scheme to 'http' or 'ipps' scheme to 'https'

2. add an explicit port 631 if the http-URL or https-URL does not contain an explicit port. Note: port 631 is the IANA assigned Well Known Port for the 'ipp' and 'ipps' schemes.

The Client MUST use the target http-URL or https-URL in both the HTTP Request-Line and HTTP headers, as specified by HTTP [RFC7230]. However, the Client MUST use the target ipp-URI or ipps-URI 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-URI or ipps-URI 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-URI "ipp://printer.example.com/ipp/print/myqueue", it opens a TCP connection to port 631 (the ipp implicit port) on the host "printer.example.com" and sends the following data:

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

Figure 11: Direct IPP Request

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

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

Figure 12: Proxied IPP Request

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

6. IANA Considerations

This section describes the procedures for allocating encoding for the following standards track extensions and vendor extensions to the IPP/1.1 Encoding and Transport document:

1. attribute syntaxes - see Section 6.3 of [RFC2911bis]

2. attribute groups - see Section 6.5 of [RFC2911bis]

3. out-of-band attribute values - see Section 6.7 of [RFC2911bis]

These extensions follow the "type2" registration procedures defined in Section ?? of [RFC2911bis]. Extensions registered for use with IPP are OPTIONAL for Client and IPP object conformance to the IPP/1.1 Encoding and Transport document.

These extension procedures are aligned with the guidelines as set forth by the IESG [IANA-CON]. The [RFC2911bis] describes how to propose new registrations for consideration. IANA will reject registration proposals that leave out required information or do not follow the appropriate format described in [RFC2911bis]. The IPP/1.1 Encoding and Transport document may also be extended by an appropriate standards track document that specifies any of the above extensions.

7. Internationalization Considerations

See the section on "Internationalization Considerations" in the document "Internet Printing Protocol/1.1: Model and Semantics" [RFC2911bis] for information on internationalization. This document adds no additional issues.

8. Security Considerations

The IPP Model and Semantics document [RFC2911bis] 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.

8.1. Security Conformance Requirements

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

8.1.1. Digest Authentication

IPP Clients and Printers SHOULD support Digest Authentication [HTTPAUTH-DIGEST]. If supported, MD5 and MD5-sess MUST be implemented and supported. Use of the Message Integrity feature (qop="auth-int") is OPTIONAL.

The reasons that IPP Clients and 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.

2. Print devices that have high-volume throughput and have available ROM space are more likely 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 occurs.

8.1.2. Transport Layer Security (TLS)

IPP Clients and Printers SHOULD support Transport Layer Security (TLS) [RFC5246] [RFC7525] for Server Authentication and Operation Privacy. IPP Printers MAY also support TLS for Client Authentication. IPP Clients and Printers MAY support Basic Authentication [HTTPAUTH-BASIC] for User Authentication if the channel is secure.

The IPP Model and Semantics document [RFC2911bis] 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 is the primary reference for security implications with regard to the IPP protocol itself.

8.2. Using IPP with TLS

IPP uses the "Upgrading to TLS Within HTTP/1.1" mechanism [RFC2817] for 'ipp' URIs. 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.

IPP uses the "HTTPS: HTTP over TLS" mechanism [RFC2818] for 'ipps' URIs. The Client and server negotiate a secure TLS connection immediately and unconditionally.

9. Interoperability with Other IPP Versions

It is beyond the scope of this specification to mandate conformance with versions of IPP other than 1.1. IPP was deliberately designed,

however, to make supporting other versions easy. IPP Printer implementations SHOULD:

- understand any valid request in the format of IPP/1.1, 2.0, 2.1, or 2.2;

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

IPP Clients SHOULD:

- understand any valid response in the format of IPP/1.1, 2.0, 2.1, or 2.2.

9.1. 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 the highest supported value, e.g., '1.1', '2.0', etc., 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. IPP objects SHOULD accept any request with the major version '1' or '2', or reject the request for reasons other than 'server-error-version-not-supported'. See [RFC2911bis] "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/2.0 conforming Printer object accepts version '1.1' requests and is configured to enforce Digest Authentication, it MUST do the same for a version '1.1' request.

9.2. Security and URI Schemes

The following are rules regarding security, the "version-number" parameter, and the URI 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', 'ipps', 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. In any case, the server MUST NOT compromise security when a Client supplies an 'ipp' or other non-secure URI scheme in the target "printer-uri" and "job-uri" operation attributes in a request.

10. Changes Since RFC 2910

The following changes have been made since the publication of RFC 2910:

- Added references to current IPP extension specifications.

- Added optional support for HTTP/2.

- Added collection attribute syntax from RFC 3382.

- Fixed typographical errors.

- Now reference TLS/1.2 and no longer mandate the TLS/1.0 MTI cipher suites.

- Updated all references.

- Updated document organization to follow current style.

- Updated example ipp: URIs to follow RFC 7472 guidelines.

- Updated version compatibility for all versions of IPP.

- Updated HTTP Digest authentication to optional for Clients.

- Removed references to (experimental) IPP/1.0 and usage of http:/https: URLs.

11. References

11.1. Normative References

[ASCII] ANSI, "Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)", June 2007.

[HTTPAUTH-BASIC] Reschke, J., "The 'Basic' HTTP Authentication Scheme", February 2015, <http://tools.ietf.org/html/draft-ietf-httpauth-basicauth-update-07>.

[HTTPAUTH-DIGEST] Shekh-Yusef, R., Ahrens, D., and S. Bremer, "HTTP Digest Access Authentication", April 2015, <http://tools.ietf.org/html/draft-ietf-httpauth-digest-19>.

[PWG5100.12] Sweet, M. and I. McDonald, "IPP/2.0, 2.1, and 2.2", April 2015, <http://ftp.pwg.org/pub/pwg/ipp/wd/wd-ipp20-20150417.pdf>.

[RFC1903] McCloghrie, K., Case, J., Rose, M., and S. Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC2911bis] Sweet, M. and I. McDonald, "Internet Printing Protocol/1.1: Model and Semantics", April 2015, <https://tools.ietf.org/html/draft-sweet-rfc2911bis-01>.

[RFC3510] Herriot, R. and I. McDonald, "Internet Printing Protocol/1.1: IPP URL Scheme", RFC 3510, April 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7232] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, June 2014.

[RFC7234] Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014.

[RFC7235] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, June 2014.

[RFC7472] McDonald, I. and M. Sweet, "Internet Printing Protocol (IPP) over HTTPS Transport Binding and the 'ipps' URI Scheme", RFC 7472, March 2015.

[RFC7540] Belshe, M., Peon, R., and M. Thompson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", May 2015, <https://tools.ietf.org/html/rfc7540>.

[RFC7541] Peon, R. and H. Ruellan, "HPACK - Header Compression for HTTP/2", May 2015, <https://tools.ietf.org/html/rfc7541>.

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

11.2. Informative References

[IANA-CON]
Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[IANA-IPP]
"IANA IPP Registry", <http://www.iana.org/assignments/ipp-registrations/>.

[PWG5100.3]
Ocke, K. and T. Hastings, "IPP Production Printing Attributes Set 1", February 2001, <http://ftp.pwg.org/pub/pwg/candidates/cs-ippprodprint10-20010212-5100.3.pdf>.

[RFC1179] McLaughlin, L., "Line printer daemon protocol", RFC 1179, August 1990.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, May 2015.

11.3. URIs

[1] http://www.pwg.org/

Appendix A. Protocol Examples

A.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
0x002C		value-length
ipp://printer.example.com/ipp/print/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-long-edge	two-sided-long-edge	value
0x03	end-of-attributes	end-of-attributes-tag
%!PDF...	<PDF Document>	data

A.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
0x0030		value-length
ipp://printer.example.com/ipp/print/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

A.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-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
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

A.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
0x0030		value-length
ipp://printer.example.com/ipp/print/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

A.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
0x002C		value-length
ipp://printer.example.com/ipp/print/pinetree	printer pinetree	value
0x45	uri type	value-tag
0x000C		name-length
document-uri	document-uri	name
0x0019		value-length
ftp://foo.example.com /foo	ftp://foo.example.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

A.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-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
0x002C		value-length
ipp://printer.example.com/ipp/print/pinetree	printer pinetree	value
0x03	end-of-attributes	end-of-attributes-tag

A.7. Create-Job Request with Collection Attributes

The following is an example of Create-Job request with the "media-col" collection attribute [PWG5100.3] with the value "media-size={x-dimension=21000 y-dimension=29700} media-type='stationery'":

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-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
0x002C		value-length
ipp://printer.example.com/ipp/print/pinetree	printer pinetree	value
0x34	begCollection	value-tag
0x0009	9	name-length
media-col	media-col	name
0x0000	0	value-length
0x4A	memberAttrName	value-tag
0x0000	0	name-length
0x000A	10	value-length
media-size	media-size	value (member-name)
0x34	begCollection	member-value-tag
0x0000	0	name-length
0x0000	0	member-value-length
0x4A	memberAttrName	value-tag
0x0000	0	name-length
0x000B	11	value-length
x-dimension	x-dimension	value (member-name)
0x21	integer	member-value-tag
0x0000	0	name-length
0x0004	4	member-value-length
0x00005208	21000	member-value
0x4A	memberAttrName	value-tag
0x0000	0	name-length
0x000B	11	value-length
y-dimension	y-dimension	value (member-name)
0x21	integer	member-value-tag
0x0000	0	name-length
0x0004	4	member-value-length

0x00007404	29700	member-value
0x37	endCollection	end-value-tag
0x0000	0	end-name-length
0x0000	0	end-value-length
0x4A	memberAttrName	value-tag
0x0000	0	name-length
0x000A	10	value-length
media-type	media-type	value (member-name)
0x44	keyword	member-value-tag
0x0000	0	name-length
0x000A	10	member-value-length
stationery	stationery	member-value
0x37	endCollection	end-value-tag
0x0000	0	end-name-length
0x0000	0	end-value-length
0x03	end-of-attributes	end-of-attributes-tag

A.8. 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
0x002C		value-length
ipp://printer.example.com/ipp/print/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

A.9. 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-attributes-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
0x0009		sub-value-length
isch guet	isch guet	name
0x03	end-of-attributes	end-of-attributes-tag

Appendix B. Registration of MIME Media Type Information for "application/ipp"

This section is strictly informative. The MIME media type listed in this section should not be re-registered by IANA when this document is published.

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) whose syntax is described in Section 3 "Encoding of the Operation Layer" of [RFC2910bis], and whose semantics are described in [RFC2911bis].

Required parameters: none

Optional parameters: none

Encoding considerations:

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

Security considerations:

IPP protocol requests/responses do not introduce any security risks not already inherent in the underlying transport protocols. Protocol mixed-version interworking rules in [RFC2911bis] as well as protocol encoding rules in [RFC2910bis] are complete and unambiguous.

Interoperability considerations:

IPP requests (generated by clients) and responses (generated by servers) MUST comply with all conformance requirements imposed by the normative specifications [RFC2911bis] and [RFC2910bis]. Protocol encoding rules specified in [RFC2910bis] 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 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:

[RFC2911bis] Sweet, M., McDonald, I., "Internet Printing Protocol/1.1: Model and Semantics" draft-sweet-rfc2911bis-01.txt, April 26, 2015.

[RFC2910bis] Sweet, M., McDonald, I., "Internet Printing Protocol/1.1: Encoding and Transport", draft-sweet-rfc2910bis-01.txt, April 27, 2015.

Applications which use this media type:

Internet Printing Protocol (IPP) print clients and print servers, communicating using HTTP (see [RFC2910bis]), 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.

Appendix C. Acknowledgements

The authors would like to acknowledge the following individuals for their contributions to the original IPP/1.1 specification:

Robert Herriot (original RFC 2910 editor), Paul Moore, Sylvan Butler, Randy Turner, and John Wenn

Appendix D. Change History

D.1. Changes In -05

The following changes are in draft-sweet-rfc2910bis-05:

- Submission type is now IETF (AD-sponsored), clarify goals.
- Abstract: This document does not define the 'ipp' URI scheme.
- Section 5: drop reference to RFC 2617
- Section 8.1.1: combine identical Client and Printer requirements
- Section 8.1.2: also applies to clients, HTTP Basic is User authentication, not Client authentication.
- References to RFC 2617 are updated to the updated drafts in the RFC editor's queue
- Global: Client, Printer, and Job are defined terms, capitalize

D.2. Changes In -04

The following changes are in draft-sweet-rfc2910bis-04:

- Removed more references to IPP/1.0.
- Section 5: Be explicit about ipp/s-URI and http/s-URL
- Section 9.1: Reword SHOULD recommendation (avoid passive voice)
- Section 9.2: Reword item 3: the server MUST NOT compromise security...
- Make sure to use URI for generic schemes and URL for HTTP/HTTPS.
- Fixed incorrect usage of lowercase conformance words.

D.3. Changes In -03

The following changes are in draft-sweet-rfc2910bis-03:

- New HTTP/2 RFCs: 7540, 7541
- Added informative reference to UTA BCP (RFC 7525)

- Culled list of people in acknowledgements to the original RFC 2910 editors, per IPP F2F.

D.4. Changes In -02

The following changes are in draft-sweet-rfc2910bis-02:

- Sections 3.1.x: Dropped "Picture of the encoding of" from titles

- Section 3.4.1: Added reference to IPP version interoperability section.

- Section 3.4.2: Removed paragraph on status-code as operation attribute (already covered in 3.4 intro) and updated HTTP status code 200 name (OK)

- Section 3.5: Added reference to IANA IPP registry for tags.

- Section 3.8: Mention SIGNED-BYTE is 1 octet.

- Section 4.1: Drop mention of URIs not being widely implemented, and that implementations will pass around URLs (not true). Also remove more "need not" text.

- Section 6: Fixed references.

- Section 8.1.1: Make Digest authentication a SHOULD for clients.

- Section 9: Reworked for generic IPP version compatibility.

- Section 9.1: Reworked for IPP 2.x compatibility.

- Section 9.2: Drop reference to IPP/1.0 and http/https URI schemes.

- Appendix A: Updated example URIs to follow IETF and IPP/IPPS URI examples

- Global: ipp-URL tp ipp-URI, URL to URI

- Global: Don't use conformance language for statements of fact.

- Global: Change "NOTE:" to "Note:" for consistency.

D.5. Changes In -01

The following changes are in draft-sweet-rfc2910bis-01:

- Errata ID 4100: Cleaned up TLS references and recommendations - no longer include cipher suites.

- Errata ID 4172: Fixed range of standards-track value tags (to 0x3ffffff not 0x37777777)

- Updated RFC references.

- Added HTTP/2 references, made it clear that only HTTP/1.1 is required and HTTP/2 is optional.

- Added collection attribute encoding from RFC 3382.

Authors' Addresses

Michael Sweet
Apple Inc.
1 Infinite Loop
MS 111-HOMC
Cupertino, CA 95014
US

Email: msweet@apple.com

Ira McDonald
High North, Inc.
PO Box 221
Grand Marais, MI 49839
US

Phone: +1 906-494-2434
Email: blueroofmusic@gmail.com