Internet Printing Protocol/1.0: Encoding and Transport

Status of this Memo

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

To learn the current status of any Internet-Draft, please check the "Iid-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

Copyright Notice

Copyright (C) The Internet Society (1998). All Rights Reserved.

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".
The full set of IPP documents includes:

- Design Goals for an Internet Printing Protocol [ipp-req]
- Rationale for the Structure and Model and Protocol for the Internet Printing Protocol [ipp-rat]
- Internet Printing Protocol/1.0: Model and Semantics [ipp-mod]
- Internet Printing Protocol/1.0: Encoding and Transport (this document)
- Internet Printing Protocol/1.0: Implementer’s Guide [ipp-iig]
- Mapping between LPD and IPP Protocols [ipp-lpd]

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/1.0. Operator and administrator requirements are out of scope for version 1.0.

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 specifications, and gives background and rationale for the IETF working group’s major decisions.

The document, "Internet Printing Protocol/1.0: 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.

This document "Internet Printing Protocol/1.0: 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.......................................................4
2. Conformance Terminology..........................................4
3. Encoding of the Operation Layer....................................4
   3.1 Picture of the Encoding.......................................5
   3.2 Syntax of Encoding............................................7
   3.3 Version-number................................................8
   3.4 Operation-id..................................................8
   3.5 Status-code..................................................9
   3.6 Request-id..................................................9
   3.7 Tags..................................................................9
      3.7.1 Delimiter Tags............................................9
      3.7.2 Value Tags................................................10
   3.8 Name-Length....................................................12
   3.9 (Attribute) Name.............................................12
   3.10 Value Length..................................................14
   3.11 (Attribute) Value............................................15
   3.12 Data..........................................................16
4. Encoding of Transport Layer.......................................16
5. Security Considerations............................................17
   5.1 Using IPP with SSL3..........................................18
6. References..........................................................18
7. Author’s Address..................................................20
8. Other Participants................................................20
9. Appendix A: Protocol Examples....................................21
   9.1 Print-Job Request..............................................21
   9.2 Print-Job Response (successful)..............................22
   9.3 Print-Job Response (failure)..................................23
   9.4 Print-Job Response (success with attributes ignored)....24
   9.5 Print-URI Request..............................................25
   9.6 Create-Job Request............................................26
   9.7 Get-Jobs Request..............................................27
   9.8 Get-Jobs Response.............................................28
10. Appendix C: Registration of MIME Media Type Information for
    "application/ipp"....................................................29
11. Appendix D: Full Copyright Statement.............................31
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 2068 [rfc2068] 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/1.0: 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

Herriot, Butler,
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)        |   2 bytes  - required
|                      or                     |
|               status-code (response)        |   2 bytes  - required
-----------------------------------------------
|                   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:

<table>
<thead>
<tr>
<th>value-tag</th>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-length (value is u)</td>
<td>2 bytes</td>
</tr>
<tr>
<td>name</td>
<td>u bytes</td>
</tr>
</tbody>
</table>

| value-length (value is v) | 2 bytes |
| value | v bytes |

An additional value consists of:

<table>
<thead>
<tr>
<th>value-tag</th>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-length (value is 0x0000)</td>
<td>2 bytes</td>
</tr>
<tr>
<td>value-length (value is w)</td>
<td>2 bytes</td>
</tr>
<tr>
<td>value</td>
<td>w bytes</td>
</tr>
</tbody>
</table>

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

>From the standpoint of a parsing loop, the encoding consists of:
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 / 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 0 (0x00). The ABNF for these two bytes
MUST be %x01.00.

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:

<table>
<thead>
<tr>
<th>Tag Value (Hex)</th>
<th>Delimiter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>reserved</td>
</tr>
<tr>
<td>0x01</td>
<td>operation-attributes-tag</td>
</tr>
<tr>
<td>0x02</td>
<td>job-attributes-tag</td>
</tr>
<tr>
<td>0x03</td>
<td>end-of-attributes-tag</td>
</tr>
<tr>
<td>0x04</td>
<td>printer-attributes-tag</td>
</tr>
<tr>
<td>0x05</td>
<td>unsupported-attributes-tag</td>
</tr>
<tr>
<td>0x06-0x0e</td>
<td>reserved for future delimiters</td>
</tr>
<tr>
<td>0x0F</td>
<td>reserved for future chunking-end-of-attributes-tag</td>
</tr>
</tbody>
</table>
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 section 9 "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.

<table>
<thead>
<tr>
<th>Tag Value (Hex)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>unsupported</td>
</tr>
<tr>
<td>0x11</td>
<td>reserved for future ‘default’</td>
</tr>
</tbody>
</table>
Tag Value (Hex) Meaning

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:

<table>
<thead>
<tr>
<th>Tag Value (Hex)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20</td>
<td>reserved</td>
</tr>
<tr>
<td>0x21</td>
<td>integer</td>
</tr>
<tr>
<td>0x22</td>
<td>boolean</td>
</tr>
<tr>
<td>0x23</td>
<td>enum</td>
</tr>
<tr>
<td>0x24-0x2F</td>
<td>reserved for future integer types</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Tag Value (Hex)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x30</td>
<td>octetString with an unspecified format</td>
</tr>
<tr>
<td>0x31</td>
<td>dateTime</td>
</tr>
<tr>
<td>0x32</td>
<td>resolution</td>
</tr>
<tr>
<td>0x33</td>
<td>rangeOfInteger</td>
</tr>
<tr>
<td>0x34</td>
<td>reserved for collection (in the future)</td>
</tr>
<tr>
<td>0x35</td>
<td>textWithLanguage</td>
</tr>
<tr>
<td>0x36</td>
<td>nameWithLanguage</td>
</tr>
<tr>
<td>0x37-0x3F</td>
<td>reserved for future octetString types</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Tag Value (Hex)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x40</td>
<td>reserved</td>
</tr>
<tr>
<td>0x41</td>
<td>textWithoutLanguage</td>
</tr>
<tr>
<td>0x42</td>
<td>nameWithoutLanguage</td>
</tr>
<tr>
<td>0x43</td>
<td>reserved</td>
</tr>
<tr>
<td>0x44</td>
<td>keyword</td>
</tr>
<tr>
<td>0x45</td>
<td>uri</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>0x46</td>
<td>uriScheme</td>
</tr>
<tr>
<td>0x47</td>
<td>charset</td>
</tr>
<tr>
<td>0x48</td>
<td>naturalLanguage</td>
</tr>
</tbody>
</table>

Herriot, Butler,
Tag Value (Hex) | Meaning
--- | ---
0x49 | mimeMedia-type
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:

@ "printer-uri": When the target is a printer and the transport is HTTP or HTTPS (for SSL3 [ssl]), the target printer-uri defined in each operation in the IPP model document MUST be an operation attribute called "printer-uri" and it MUST also be specified outside of the operation layer as the request-URI on the Request-Line at the HTTP level.

@ "job-uri": When the target is a job and the transport is HTTP or HTTPS (for SSL3), the target job-uri of each operation in the IPP model document MUST be an operation attribute called "job-uri" and it MUST also be specified outside of the operation layer as the request-URI on the Request-Line at the HTTP level.

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-attributes-sequence preceded by the appropriate xxx-attributes-tag (See the table below and section 9 "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:

<table>
<thead>
<tr>
<th>Model Document Group</th>
<th>xxx-attributes-sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation Attributes</td>
<td>operations-attributes-sequence</td>
</tr>
<tr>
<td>Job Template Attributes</td>
<td>job-attributes-sequence</td>
</tr>
<tr>
<td>Job Object Attributes</td>
<td>job-attributes-sequence</td>
</tr>
<tr>
<td>Unsupported Attributes</td>
<td>unsupported-attributes-sequence</td>
</tr>
<tr>
<td>Requested Attributes (Get-Job-Attributes)</td>
<td>job-attributes-sequence</td>
</tr>
<tr>
<td>Requested Attributes (Get-Printer-Attributes)</td>
<td>printer-attributes-sequence</td>
</tr>
<tr>
<td>Document Content</td>
<td>in a special position as described above</td>
</tr>
</tbody>
</table>

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 9 "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. If a client receives a response with a nonzero value-length in this case, it MUST ignore the value field. If a printer receives a request with a nonzero value-length in this case, it MUST reject the request.

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, naturalLanguage, mimeMediaType, keyword, uri, and uriScheme US-ASCII-STRING.
boolean SIGNED-BYTE where 0x00 is 'false' and 0x01 is 'true'.

Herriot, Butler, [Page 15]
Syntax of Attribute Encoding Value

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.
rangleOfInteger 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

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 [rfc2068] 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. In

Herriot, Butler,
addition, a printer may have to support another port for privacy (See Section 5 "Security Considerations").

Note: even though port 631 is the IPP default, port 80 remains the default for an HTTP URI. Thus a URI for a printer using port 631 MUST contain an explicit port, e.g. "http://forest:631/pinetree". An HTTP URI for IPP with no explicit port implicitly reference port 80, which is consistent with the rules for HTTP/1.1. 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. A printer (server) implementation MUST adhere the rules for an origin server described for HTTP1.1.

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.

5. Security Considerations

The IPP Model document defines an IPP implementation with "privacy" as one that implements Secure Socket Layer Version 3 (SSL3). Note: SSL3 is not an IETF standards track specification. SSL3 meets the requirements for IPP security with regards to features such as mutual authentication and privacy (via encryption). The IPP Model document also outlines IPP-specific security considerations and should be the primary reference for security implications with regards to the IPP protocol itself.

The IPP Model document defines an IPP implementation with "authentication" as one that implements the standard way for transporting IPP messages within HTTP 1.1. These include the security considerations outlined in the HTTP 1.1 standard document and Digest Access Authentication extension.

The current HTTP infrastructure supports HTTP over TCP port 80. IPP server implementations MUST offer IPP services using HTTP over the IANA assigned Well Known Port 631 (the IPP default port). IPP server implementations may support other ports, in addition to this port.

See further discussion of IPP security concepts in the model document.
5.1 Using IPP with SSL3

An assumption is that the URI for a secure IPP Printer object has been found by means outside the IPP printing protocol, via a directory service, web site or other means.

IPP provides a transparent connection to SSL by calling the corresponding URL (a https URI connects by default to port 443). However, the following functions can be provided to ease the integration of IPP with SSL during implementation:

connect (URI), returns a status

"connect" makes an https call and returns the immediate status of the connection as returned by SSL to the user. The status values are explained in section 5.4.2 of the SSL document [ssl].

A session-id may also be retained to later resume a session. The SSL handshake protocol may also require the cipher specifications supported by the client, key length of the ciphers, compression methods, certificates, etc. These should be sent to the server and hence should be available to the IPP client (although as part of administration features).

disconnect (session)

to disconnect a particular session.

The session-id available from the "connect" could be used.

resume (session)

to reconnect using a previous session-id.

The availability of this information as administration features are left for implementers, and need not be specified at this time.

6. References


November 1996 (Also BCP0013), RFC 2048.
7. Author’s Address

Robert Herriot (editor)                      Paul Moore
Sun Microsystems Inc.                          Microsoft
901 San Antonio Road, MPK-17                 1 Microsoft Way
Palo Alto, CA 94303                          Redmond, WA 98053

Phone: 650-786-8995                         Phone: 425-936-0908
Fax: 650-786-7077                            Fax: 425-93MS-FAX
Email: robert.herriot@eng.sun.com            Email: paulmo@microsoft.com

Sylvan Butler                                    Randy Turner
Hewlett-Packard                                Sharp Laboratories  
11311 Chinden Blvd.                           5750 NW Pacific Rim Blvd
Boise, ID 83714                                Camas, WA 98607

Phone: 208-396-6000                        Phone: 360-817-8456
Fax: 208-396-3457                               Fax: 360-817-8436
Email: rturner@sharplabs.com
Email: sbutler@boi.hp.com

IPP Mailing List:  ipp@pwg.org
IPP Mailing List Subscription:  ipp-request@pwg.org
IPP Web Page:  http://www.pwg.org/ipp/

8. Other Participants:

Chuck Adams - Tektronix  Harry Lewis - IBM
Ron Bergman - Dataproducts  Tony Liao - Vivid Image
Keith Carter - IBM  David Manchala - Xerox
Angelo Caruso - Xerox  Carl-Uno Manros - Xerox

Herriot, Butler,
9. Appendix A: Protocol Examples

9.1 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.

<table>
<thead>
<tr>
<th>Octets</th>
<th>Symbolic Value</th>
<th>Protocol field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00012</td>
<td>attributes-charset</td>
<td>attributes-charset</td>
</tr>
<tr>
<td></td>
<td>attributes-charset</td>
<td>name</td>
</tr>
<tr>
<td>0x0008</td>
<td>us-ascii</td>
<td>US-ASCII</td>
</tr>
<tr>
<td></td>
<td>value-length</td>
<td>value</td>
</tr>
<tr>
<td>0x48</td>
<td>natural-language type</td>
<td>value-length</td>
</tr>
<tr>
<td></td>
<td>value</td>
<td></td>
</tr>
<tr>
<td>0x001B</td>
<td>attributes-natural-language</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>name-length</td>
<td></td>
</tr>
<tr>
<td>0x0005</td>
<td>en-us</td>
<td>en-US</td>
</tr>
<tr>
<td></td>
<td>value-length</td>
<td>value</td>
</tr>
<tr>
<td>URI Type</td>
<td>Value-Tag</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------</td>
<td></td>
</tr>
<tr>
<td>0x45</td>
<td>Name-Length</td>
<td></td>
</tr>
<tr>
<td>0x000B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Herriot, Butler,
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’.
Herriot, Butler,
9.3 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

0x0100                                     1.0                                    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
0x0008                                    attributes-charset name
0x0008                                    US-ASCII value-length
us-ascii                                   value
Herriot, Butler,
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).

```
<table>
<thead>
<tr>
<th>Octets</th>
<th>Symbolic Value</th>
<th>Protocol field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0100</td>
<td>1.0</td>
<td>version-number</td>
</tr>
<tr>
<td>0x0001</td>
<td>successful-ok-ignored-or-substituted-attributes</td>
<td>status-code</td>
</tr>
<tr>
<td>0x00000001</td>
<td>1</td>
<td>request-id</td>
</tr>
<tr>
<td>0x01</td>
<td>start operation-attributes</td>
<td>operation-attributes-tag</td>
</tr>
<tr>
<td>0x47</td>
<td>charset type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0012</td>
<td>attributes-charset</td>
<td>name-length</td>
</tr>
<tr>
<td>0x0008</td>
<td></td>
<td>value-length</td>
</tr>
</tbody>
</table>
```
Herriot, Butler,
Octets                                                                 Symbolic Value                  Protocol field
0x001B attributes-                                              attributes-natural-              name-length
natural-language                                             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
0x05 start unsupported-attributes                             unsupported-attributes         tag
attributes                                                     attributes                      tag
0x21 integer type                          integer type                      value-tag
0x0006                                                        name-length
copies                                                      copies                         name
0x0004                                                        value-length
0x000000014                                                   value
0x10 unsupported (type)                                      unsupported (type)          value-tag
0x0005                                                        name-length
sides                                                       sides                         name
0x0000                                                        value-length
0x02 start job-attributes                                     job-attributes-tag          tag
0x21 integer                                              integer                        value-tag
0x0006                                                        name-length
job-id                                                      job-id                          name
0x0004                                                        value-length
147 147                                                       value
0x45 uri type                                             uri type                        value-tag
0x0007                                                        name-length
job-uri                                                    job-uri                          name
0x001E                                                        value-length
http://forest:63/1/pinetree/123 job 123 on pinetree        value
0x42 nameWithoutLanguage type                                 name-length
0x0009                                                        name-length
job-state                                                  job-state                         name
0x0004                                                        value-length
0x0003 pending                                             pending                         value
0x03 end-of-attributes                                      end-of-attributes-tag

9.5 Print-URI Request

The following is an example of Print-URI request with copies and job-name parameters:
<table>
<thead>
<tr>
<th>Octets</th>
<th>Symbolic Value</th>
<th>Protocol field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0100</td>
<td>1.0</td>
<td>version-number</td>
</tr>
</tbody>
</table>

Herriot, Butler,
9.6 Create-Job Request

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

Octets  Symbolic Value  Protocol field
0x00003  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
0x001A  value-length
http://forest:631/pinetree  value
/fo
0x42  nameWithoutLanguage type  value-tag
0x0008  name-length
job-name  job-name  name
0x0006  value-length
fooar  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
The following is an example of Get-Jobs request with parameters but no attributes:

Octets          Symbolic Value            Protocol field
0x0100          1.0                      version-number
0x000A          Get-Jobs                 operation-id
0x00000123      0x123                    request-id
0x01            start operation-attributes operation-attributes-tag
0x47            charset type             value-tag
0x0012          attributes-charset      name-length
0x0008          value-length
us-ascii        US-ASCII                 value
0x48            natural-language type   value-tag
0x001B          name-length
0x0005          attributes-natural-language name
http://forest:6 printer pinetree       value
31/pinetree
0x03            end-of-attributes      end-of-attributes-tag

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

```plaintext
Octets Symbolic Value Protocol field
0x0100 1.0 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 attributes-charset name
0x000A ISO-8859-1 value
0x48 natural-language type value-tag
0x001B attributes-natural-language name
0x0005 en-us value-length
0x41 textWithoutLanguage type value-tag
0x000E status-message name
0x000D successful-ok value-length
0x0002 start job-attributes (1st object)
0x21 integer type value-tag
0x0006 job-id name-length
```
Herriot, Butler,
### 10. 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/1.0, 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:**

<table>
<thead>
<tr>
<th>Octets</th>
<th>Symbolic Value</th>
<th>Protocol field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x36</td>
<td>nameWithLanguage</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0008</td>
<td>job-name</td>
<td>name-length</td>
</tr>
<tr>
<td>0x000C</td>
<td>job-name</td>
<td>name</td>
</tr>
<tr>
<td>0x0005</td>
<td>fr-ca</td>
<td>value-length</td>
</tr>
<tr>
<td>0x0003</td>
<td>fr-CA</td>
<td>name</td>
</tr>
<tr>
<td>0x02</td>
<td>start job-attributes (2nd</td>
<td>job-attributes-tag</td>
</tr>
<tr>
<td></td>
<td>object)</td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>start job-attributes (3rd</td>
<td>job-attributes-tag</td>
</tr>
<tr>
<td></td>
<td>object)</td>
<td></td>
</tr>
<tr>
<td>0x21</td>
<td>integer type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0006</td>
<td>job-id</td>
<td>name-length</td>
</tr>
<tr>
<td>0x0004</td>
<td>job-id</td>
<td>name</td>
</tr>
<tr>
<td>148</td>
<td>148</td>
<td>value</td>
</tr>
<tr>
<td>0x36</td>
<td>nameWithLanguage</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0008</td>
<td>job-name</td>
<td>name-length</td>
</tr>
<tr>
<td>0x0012</td>
<td>job-name</td>
<td>name</td>
</tr>
<tr>
<td>0x0005</td>
<td>de-CH</td>
<td>value-length</td>
</tr>
<tr>
<td>0x0009</td>
<td>de-CH</td>
<td>name</td>
</tr>
<tr>
<td>0x03</td>
<td>end-of-attributes</td>
<td>end-of-attributes-tag</td>
</tr>
</tbody>
</table>
IPP/1.0 protocol requests/responses MAY contain long lines and ALWAYS contain binary data (for example attribute value lengths).

Security considerations:

IPP/1.0 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/1.0 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/1.0 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 specification:


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:

Scott A. Isaacson
Novell, Inc.
122 E 1700 S
Provo, UT 84606

Phone: 801-861-7366
Fax: 801-861-4025
Email: sisaacson@novell.com
or

Herriot, Butler,
11. Appendix D: Full Copyright Statement

Copyright (C)The Internet Society (1998). 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.