Internet Printing Protocol/1.1: Encoding and Transport

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2000). 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 Hypertext Transfer Protocol (HTTP) a message body whose Content-Type is "application/ipp". This document defines a new scheme named 'ipp' for identifying IPP printers and jobs.
The full set of IPP documents includes:

- Design Goals for an Internet Printing Protocol [RFC2567]
- Rationale for the Structure and Model and Protocol for the Internet Printing Protocol [RFC2568]
- Internet Printing Protocol/1.1: Model and Semantics [RFC2911]
- Internet Printing Protocol/1.1: Encoding and Transport (this document)
- Internet Printing Protocol/1.1: Implementer’s Guide [ipp-iig]
- Mapping between LPD and IPP Protocols [RFC2569]

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

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

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


The document "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 ................................... 6
      3.1.1 Request and Response ................................ 6
      3.1.2 Attribute Group ...................................... 6
      3.1.3 Attribute ........................................... 7
      3.1.4 Picture of the Encoding of an Attribute-with-one-value. 7
      3.1.5 Additional-value ...................................... 8
      3.1.6 Alternative Picture of the Encoding of a Request Or a Response ........................................... 9
   3.2 Syntax of Encoding ........................................... 9
   3.3 Attribute-group ............................................. 11
   3.4 Required Parameters ....................................... 12
      3.4.1 Version-number ...................................... 12
      3.4.2 Operation-id .......................................... 12
      3.4.3 Status-code .......................................... 12
      3.4.4 Request-id ............................................ 13
   3.5 Tags ...................................................... 13
      3.5.1 Delimiter Tags ........................................ 13
      3.5.2 Value Tags ............................................ 14
   3.6 Name-Length .............................................. 16
   3.7 (Attribute) Name ........................................... 16
   3.8 Value Length ............................................ 16
   3.9 (Attribute) Value ........................................ 17
   3.10 Data .................................................... 18
4. Encoding of Transport Layer .................................... 18
   4.1 Printer-uri and job-uri ................................... 19
5. IPP URL Scheme .................................................. 20
6. IANA Considerations ............................................. 22
7. Internationalization Considerations ........................... 23
8. Security Considerations ......................................... 23
   8.1 Security Conformance Requirements ........................ 23
      8.1.1 Digest Authentication ................................ 23
      8.1.2 Transport Layer Security (TLS) ....................... 24
   8.2 Using IPP with TLS ....................................... 25
9. Interoperability with IPP/1.0 Implementations .................. 25
   9.1 The "version-number" Parameter ............................ 25
   9.2 Security and URL Schemes .................................. 26
10. References .................................................... 27
11. Authors’ Addresses ............................................ 29
12. Other Participants: ........................................... 31
13. Appendix A: Protocol Examples ................................ 33
    13.1 Print-Job Request ....................................... 33
    13.2 Print-Job Response (successful) ........................ 34
    13.3 Print-Job Response (failure) ........................... 35
13.4 Print-Job Response (success with attributes ignored) .....36
13.5 Print-URI Request ........................................... 38
13.6 Create-Job Request ........................................ 39
13.7 Get-Jobs Request ........................................... 40
13.8 Get-Jobs Response ........................................... 41
14. Appendix B: Registration of MIME Media Type Information for
"application/ipp"..................................................42
15. Appendix C: Changes from IPP/1.0 ..............................44
16. Full Copyright Statement ........................................ 45

1. Introduction

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

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

The operation layer consists of a message body in an HTTP request or
response. The document "Internet Printing Protocol/1.1: Model and
Semantics" [RFC2911] defines the semantics of such a message body and
the supported values. This document specifies the encoding of an IPP
operation. The aforementioned document [RFC2911] is henceforth
referred to as the "IPP model document" or simply "model document".

Note: the version number of IPP (1.1) and HTTP (1.1) are not linked.
They both just happen to be 1.1.

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 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 2234 [RFC2234]

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                     |
|               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
```

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
```

The "begin-attribute-group-tag" field marks the beginning of an "attribute-group" field and its value identifies the type of attribute group, e.g. Operations Attributes group versus a Job Attributes group. The "begin-attribute-group-tag" field also marks the end of the previous attribute group except for the "begin-attribute-group-tag" field in the first "attribute-group" field of a request or response. The "begin-attribute-group-tag" field acts as an "attribute-group" terminator because an "attribute-group" field cannot nest inside another "attribute-group" field.

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
|-------------------------------------------|
```

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 Picture of the Encoding of an 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
|------------------------------------------|
```
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 |

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

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

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).

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

3.1.6 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:

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>version-number</td>
</tr>
<tr>
<td>operation-id (request) or status-code (response)</td>
</tr>
<tr>
<td>request-id</td>
</tr>
<tr>
<td>tag (delimiter-tag or value-tag)</td>
</tr>
<tr>
<td>empty or rest of attribute</td>
</tr>
<tr>
<td>end-of-attributes-tag</td>
</tr>
<tr>
<td>data</td>
</tr>
</tbody>
</table>

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

- "begin-attribute-group-tag": expect zero or more "attribute" fields
- "value-tag": expect the remainder of an "attribute-with-one-value" or an "additional-value".
- "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 [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 *attribute-group end-of-attributes-tag data
ipp-response = version-number status-code request-id *attribute-group end-of-attributes-tag data
attribute-group = begin-attribute-group-tag *attribute

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

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 ; 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

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
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
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:

<table>
<thead>
<tr>
<th>Model Document Group</th>
<th>&quot;begin-attribute-group-tag&quot; field values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation Attributes</td>
<td>&quot;operations-attributes-tag&quot;</td>
</tr>
<tr>
<td>Job Template Attributes</td>
<td>&quot;job-attributes-tag&quot;</td>
</tr>
<tr>
<td>Job Object Attributes</td>
<td>&quot;job-attributes-tag&quot;</td>
</tr>
<tr>
<td>Unsupported Attributes</td>
<td>&quot;unsupported-attributes-tag&quot;</td>
</tr>
<tr>
<td>Requested Attributes (Get-Job-Attributes)</td>
<td>&quot;job-attributes-tag&quot;</td>
</tr>
<tr>
<td>Requested Attributes (Get-Printer-Attributes)</td>
<td>&quot;printer-attributes-tag&quot;</td>
</tr>
<tr>
<td>Document Content</td>
<td>in a special position as described above</td>
</tr>
</tbody>
</table>

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 [RFC2911]. 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 MUST consist of a major and minor version-number, each of which MUST be represented by a SIGNED-BYTE. The major version-number MUST be the first byte of the encoding and the minor version-number MUST be the second byte of the encoding. The protocol described in this document MUST have a major version-number of 1 (0x01) and a minor version-number of 1 (0x01). The ABNF for these two bytes MUST be %x01.01.

3.4.2 Operation-id

The "operation-id" field MUST contain an operation-id value defined in the model document. The value MUST be encoded as a SIGNED-SHORT and it MUST be in the third and fourth bytes of the encoding of an operation request.

3.4.3 Status-code

The "status-code" field MUST contain a status-code value defined in the model document. The value MUST be encoded as a SIGNED-SHORT and it MUST be in the third and fourth bytes of the encoding of an operation response.
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.

3.4.4 Request-id

The "request-id" field MUST contain a request-id value as defined in the model document. The value MUST be encoded as a SIGNED-INTEGER and it MUST be 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

3.5.1 Delimiter Tags

The following table specifies the values for the delimiter tags:

<table>
<thead>
<tr>
<th>Tag Value (Hex)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>reserved for definition in a future IETF standards track document</td>
</tr>
<tr>
<td>0x01</td>
<td>&quot;operation-attributes-tag&quot;</td>
</tr>
<tr>
<td>0x02</td>
<td>&quot;job-attributes-tag&quot;</td>
</tr>
<tr>
<td>0x03</td>
<td>&quot;end-of-attributes-tag&quot;</td>
</tr>
<tr>
<td>0x04</td>
<td>&quot;printer-attributes-tag&quot;</td>
</tr>
<tr>
<td>0x05</td>
<td>&quot;unsupported-attributes-tag&quot;</td>
</tr>
<tr>
<td>0x06-0x0f</td>
<td>reserved for future delimiters in IETF standards track documents</td>
</tr>
</tbody>
</table>

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

<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 ‘default’ for definition in a future IETF standards track document</td>
</tr>
<tr>
<td>0x12</td>
<td>unknown</td>
</tr>
<tr>
<td>0x13</td>
<td>no-value</td>
</tr>
<tr>
<td>0x14-0x1F</td>
<td>reserved for &quot;out-of-band&quot; values in future IETF standards track documents</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Tag Value (Hex)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20</td>
<td>reserved for definition in a future IETF standards track document</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 integer types for definition in future IETF standards track documents</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" field:

<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 definition in a future IETF</td>
</tr>
<tr>
<td></td>
<td>standards track document</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 octetString type definitions in</td>
</tr>
<tr>
<td></td>
<td>future IETF standards track documents</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Tag Value (Hex)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x40</td>
<td>reserved for definition in a future IETF</td>
</tr>
<tr>
<td></td>
<td>standards track document</td>
</tr>
<tr>
<td>0x41</td>
<td>textWithoutLanguage</td>
</tr>
<tr>
<td>0x42</td>
<td>nameWithoutLanguage</td>
</tr>
<tr>
<td>0x43</td>
<td>reserved for definition in a future IETF</td>
</tr>
<tr>
<td></td>
<td>standards track document</td>
</tr>
<tr>
<td>0x44</td>
<td>keyword</td>
</tr>
<tr>
<td>0x45</td>
<td>uri</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>
<tr>
<td>0x49</td>
<td>mimeMediaType</td>
</tr>
<tr>
<td>0x4A-0x5F</td>
<td>reserved for character string type definitions in future IETF standards track documents</td>
</tr>
</tbody>
</table>

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 type definitions in IETF 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 0x00 to 0x37777777 are reserved for definition in future IETF standard track documents. The values 0x40000000 to 0x7FFFFFFF are reserved for vendor extensions.

3.6 Name-Length

The "name-length" field MUST consist of a SIGNED-SHORT. This field MUST specify 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 MUST be empty, and the following value MUST be 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 malformed (see [RFC2911] section 3.1.3). The zero-length name is the only mechanism for multi-valued attributes.

3.7 (Attribute) Name

The "name" field MUST contain the name of an attribute. The model document [RFC2911] specifies such names.

3.8 Value Length

The "value-length" field MUST consist of a SIGNED-SHORT. This field MUST specify 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 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, "Encoding of the Operation Layer". The 5 types are US-ASCII-STRING, LOCALIZED-STRING, SIGNED-INTEGER, SIGNED-SHORT, SIGNED-BYTE, and OCTET-STRING.

<table>
<thead>
<tr>
<th>Syntax of Attribute Value</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>textWithoutLanguage,</td>
<td>LOCALIZED-STRING.</td>
</tr>
<tr>
<td>nameWithoutLanguage</td>
<td></td>
</tr>
<tr>
<td>textWithLanguage</td>
<td>OCTET-STRING consisting of 4 fields:</td>
</tr>
<tr>
<td></td>
<td>a. a SIGNED-SHORT which is the number of octets in the following field</td>
</tr>
<tr>
<td></td>
<td>b. a value of type natural-language,</td>
</tr>
<tr>
<td></td>
<td>c. a SIGNED-SHORT which is the number of octets in the following field,</td>
</tr>
<tr>
<td></td>
<td>d. a value of type textWithoutLanguage.</td>
</tr>
<tr>
<td>nameWithLanguage</td>
<td>OCTET-STRING consisting of 4 fields:</td>
</tr>
<tr>
<td></td>
<td>a. a SIGNED-SHORT which is the number of octets in the following field</td>
</tr>
<tr>
<td></td>
<td>b. a value of type natural-language,</td>
</tr>
<tr>
<td></td>
<td>c. a SIGNED-SHORT which is the number of octets in the following field</td>
</tr>
<tr>
<td>charset,</td>
<td>US-ASCII-STRING.</td>
</tr>
<tr>
<td>naturalLanguage,</td>
<td></td>
</tr>
<tr>
<td>mimeMediaType,</td>
<td></td>
</tr>
<tr>
<td>keyword, uri,</td>
<td></td>
</tr>
<tr>
<td>uriScheme</td>
<td></td>
</tr>
</tbody>
</table>
Syntax of Attribute       Encoding
Value

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

rangeOfInteger           Eight octets consisting of 2 SIGNED-INTERGERS.
                         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 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 [RFC2616] is the transport layer for this protocol.

The operation layer has been designed with the assumption that the
transport layer contains the following information:
- the URI of the target job or printer operation
- the total length of the data in the operation layer, either as a single length or as a sequence of chunks each with a length.

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

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

An IPP server sends a response for each request that it receives. If an IPP server detects an error, it MAY send a response before it has read the entire request. If the HTTP layer of the IPP server completes processing the HTTP headers successfully, it MAY send an intermediate response, such as "100 Continue", with no IPP data before sending the IPP response. A client MUST expect such a variety of responses from an IPP server. For further information on HTTP/1.1, consult the HTTP documents [RFC2616].

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

4.1 Printer-uri and job-uri

All Printer and Job objects are identified by a Uniform Resource Identifier (URI) [RFC2396] so that they can be persistently and unambiguously referenced. Since every URL is a specialized form of a URI, even though the more generic term URI is used throughout the rest of this document, its usage is intended to cover the more specific notion of URL as well.
Some operation elements are encoded twice, once as the request-URI on the HTTP Request-Line and a second time as a REQUIRED operation attribute in the application/ipp entity. These attributes are the target URI for the operation and are called printer-uri and job-uri. Note: The target URI is included twice in an operation referencing the same IPP object, but the two URIs NEED NOT be literally identical. One can be a relative URI and the other can be an absolute URI. HTTP/1.1 allows clients to generate and send a relative URI rather than an absolute URI. A relative URI identifies a resource with the scope of the HTTP server, but does not include scheme, host or port. The following statements characterize how URLs should be used in the mapping of IPP onto HTTP/1.1:

1. Although potentially redundant, a client MUST supply the target of the operation both as an operation attribute and as a URI at the HTTP layer. The rationale for this decision is to maintain a consistent set of rules for mapping application/ipp to possibly many communication layers, even where URLs are not used as the addressing mechanism in the transport layer.

2. Even though these two URLs might not be literally identical (one being relative and the other being absolute), they MUST both reference the same IPP object. However, a Printer NEED NOT verify that the two URLs reference the same IPP object, and NEED NOT take any action if it determines the two URLs to be different.

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.

5. IPP URL Scheme

The IPP/1.1 document defines a new scheme ‘ipp’ as the value of a URL that identifies either an IPP printer object or an IPP job object. The IPP attributes using the ‘ipp’ scheme are specified below. Because the HTTP layer does not support the ‘ipp’ scheme, a client MUST map ‘ipp’ URLs to ‘http’ URLs, and then follows the HTTP [RFC2616][RFC2617] rules for constructing a Request-Line and HTTP headers. The mapping is simple because the ‘ipp’ scheme implies all of the same protocol semantics as that of the ‘http’ scheme
In the remainder of this section the term ‘ipp-URL’ means a URL whose scheme is ‘ipp’ and whose implicit (default) port is 631. The term ‘http-URL’ means a URL whose scheme is ‘http’, and the term ‘https-URL’ means a URL whose scheme is ‘https’.

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

- job attributes:
  - job-uri
  - job-printer-uri

- printer attributes:
  - printer-uri-supported

- operation attributes:
  - job-uri
  - printer-uri

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

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

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

When a client sends a request, it MUST convert a target ipp-URL to a target http-URL for the HTTP layer according to the following rules:

1. change the ‘ipp’ scheme to ‘http’
2. add an explicit port 631 if the URL does not contain an explicit port. Note: port 631 is the IANA assigned Well Known Port for the ‘ipp’ scheme.

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

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

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

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

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

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

6. IANA Considerations

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

1. attribute syntaxes - see [RFC2911] section 6.3
2. attribute groups - see [RFC2911] section 6.5
3. out-of-band attribute values - see [RFC2911] section 6.7
These extensions follow the "type2" registration procedures defined in [RFC2911] section 6. Extensions registered for use with IPP/1.1 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 [RFC2911] Section 11 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 [RFC2911] Section 11. The IPP/1.1 Encoding and Transport document may also be extended by an appropriate RFC 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" [RFC2911] for information on internationalization. This document adds no additional issues.

8. Security Considerations

The IPP Model and Semantics document [RFC2911] 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 MUST support:

Digest Authentication [RFC2617].

MD5 and MD5-sess MUST be implemented and supported.

The Message Integrity feature NEED NOT be used.
IPP Printers SHOULD support:

Digest Authentication [RFC2617].

MD5 and MD5-sess MUST be implemented and supported.

The Message Integrity feature NEED NOT be used.

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

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

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

8.1.2 Transport Layer Security (TLS)

IPP Printers SHOULD support Transport Layer Security (TLS) [RFC2246] for Server Authentication and Operation Privacy. IPP Printers MAY also support TLS for Client Authentication. If an IPP Printer supports TLS, it MUST support the TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite as mandated by RFC 2246 [RFC2246]. All other cipher suites are OPTIONAL. An IPP Printer MAY support Basic Authentication (described in HTTP/1.1 [RFC2617]) for Client Authentication if the channel is secure. TLS with the above mandated cipher suite can provide such a secure channel.

If an IPP client supports TLS, it MUST support the TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite as mandated by RFC 2246 [RFC2246]. All other cipher suites are OPTIONAL.
The IPP Model and Semantics document defines two printer attributes ("uri-authentication-supported" and "uri-security-supported") that the client can use to discover the security policy of a printer. That document also outlines IPP-specific security considerations and should be the primary reference for security implications with regard to the IPP protocol itself. For backward compatibility with IPP version 1.0, IPP clients and printers may also support SSL3 [ssl]. This is in addition to the security required in this document.

8.2 Using IPP with TLS

IPP/1.1 uses the "Upgrading to TLS Within HTTP/1.1" mechanism [RFC2817]. An initial IPP request never uses TLS. The client requests a secure TLS connection by using the HTTP "Upgrade" header, while the server agrees in the HTTP response. The switch to TLS occurs either because the server grants the client’s request to upgrade to TLS, or a server asks to switch to TLS in its response. Secure communication begins with a server’s response to switch to TLS.

9. Interoperability with IPP/1.0 Implementations

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

- understand any valid request in the format of IPP/1.0, or 1.1;
- respond appropriately with a response containing the same "version-number" parameter value used by the client in the request.

And we would expect IPP/1.1 clients to:

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

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 a '1.1' value and SHOULD try supplying alternate version numbers if they receive a 'server-error-version-not-supported' error return in a response.
2. IPP objects MUST accept requests containing a "version-number" parameter with a '1.1' value (or reject the request for reasons other than 'server-error-version-not-supported').

3. It is recommended that IPP objects accept any request with the major version '1' (or reject the request for reasons other than 'server-error-version-not-supported'). See [RFC2911] "versions" sub-section.

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

9.2 Security and URL Schemes

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

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

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

3. It is recommended that if a server registers a non-secure ipp-URL with a directory service (see [RFC2911] "Generic Directory Schema" Appendix), then it also register an http-URL for interoperability with IPP/1.0 clients (see section 9).

4. In any case, security MUST NOT be compromised when a client supplies an 'http' or other non-secure URL scheme in the target "printer-uri" and "job-uri" operation attributes in a request.
10. References


[iana] IANA Registry of Coded Character Sets:


11. Authors’ Addresses

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

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

Sylvan Butler
Hewlett-Packard
11311 Chinden Blvd.
Boise, ID 83714

Phone: 208-396-6000
Fax: 208-396-3457
EMail: sbutler@boi.hp.com

Paul Moore
Peerless Systems Networking
10900 NE 8th St #900
Bellevue, WA 98004

Phone: 425-462-5852
EMail: pmoore@peerless.com


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

To subscribe to the ipp mailing list, send the following email:
1) send it to majordomo@pwg.org
2) leave the subject line blank
3) put the following two lines in the message body:
   subscribe ipp
   end
12. Other Participants:

Chuck Adams - Tektronix                      Shivaun Albright - HP
Stefan Andersson - Axis                      Jeff Barnett - IBM
Ron Bergman - Hitachi Koki Imaging Systems  Dennis Carney - IBM
Keith Carter - IBM                           Angelo Caruso - Xerox
Rajesh Chawla - TR Computing Solutions       Nancy Chen - Okidata
Stefan Andersson - Axis                      Jeff Copeland - QMS
Ron Bergman - Hitachi Koki Imaging Systems  Roger deBry - IBM
Dennis Carney - IBM                          Babek Jahromi - Microsoft
Jeff Copeland - QMS                          Satoshi Fujitami - Ricoh
Angelo Caruso - Xerox
Nancy Chen - Okidata
Mabry Dozier - QMS
Maulik Desai - Auco                          David Killerman - Northlake
Steve Gubert - IBM                           Software
Charles Gordon - Osicom                      Charles Kong - Panasonic
Jerry Hadsell - IBM                          Dave Kuntz - Hewlett-Packard
Tom Hastings - Xerox                         Rick Landau - Digital
Stephen Holmstead                             Henrik Holst - I-data
Scott Isaacson - Novell                       Zhi-Hong Huang - Zenographics
Swen Johnson - Xerox                          Satoshi Fujitami - Ricoh
Robert Kline - TrueSpectra                    Babek Jahromi - Microsoft
Carl Kugler - IBM                             David Kellerman - Northlake
Takeki Kurono - Brother                       Peter Michalek - Shinesoft
Scott Lawrence - Agranot Systems             Mike Moldovan - G3 Nova
Dwight Lewis - Lexmark                        Yuichi Niwa - Ricoh
Tony Liao - Vivid Image                       Yuji Sasaki - Japan Computer
Pete Loya - HP                                Japanese Industry
Mike MacKay - Novell, Inc.                    Richard Schneider - Epson
Carl-Uno Marnos - Xerox                       Rob Rhoads - Intel
Stan McConnell - Xerox                        Richard Roberts - Ricoh
Sandra Marts - Hewlett Packard                Stuart Rowley - Kyocera
Ira McDonald - High North Inc.                Richard Schneider - Epson
Tetsuya Morita - Ricoh                        Industry
Pat Nagay - IBM                               Katsuaki Sekiguchi - Canon
Hugo Parra, Novell                            Information Systems
Patrick Powell - Astart                      Jeff Rackowitz - Intermec
Software
Eric Random - Peerless                        Rob Rhoads - Intel
Xavier Riley - Xerox                          Gary Roberts - Ricoh
David Roach - Unisys                         Stuart Rowley - Kyocera
Yuji Sasaki - Japan Computer Industry
Bob Setterbo - Adobe
Hideki Tanaka - Cannon Information Systems
Mike Timperman - Lexmark
Shigeru Ueda - Canon
William Wagner - NetSilicon/DPI
Chris Wellens - Interworking Labs
Craig Whittle - Sharp Labs
Jasper Wong - Xionics
Michael Wu - Heidelberg Digital
Michael Yeung - Canon Information Systems
Atsushi Yuki - Kyocera
William Zhang - Canon Information Systems
Steve Zilles - Adobe

Gail Songer - Peerless
Devon Taylor - Novell, Inc.
Atsushi Uchino - Epson
Bob Von Andel - Allegro Software
Jim Walker - DAZEL
Trevor Wells - Hewlett Packard
Rob Whittle - Novell, Inc.
Don Wright - Lexmark
Rick Yardumian - Xerox
Lloyd Young - Lexmark
Peter Zehler - Xerox
Frank Zhao - Panasonic
Rob Zirnstein - Canon Information Systems
13. Appendix A: Protocol Examples

13.1 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>0x0101</td>
<td>1.1</td>
<td>version-number</td>
</tr>
<tr>
<td>0x0002</td>
<td>Print-Job</td>
<td>operation-id</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</td>
</tr>
<tr>
<td>0x0008</td>
<td>us-ascii</td>
<td>value-length</td>
</tr>
<tr>
<td>0x48</td>
<td>natural-language type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x001B</td>
<td>attributes-natural-language</td>
<td>name</td>
</tr>
<tr>
<td>0x0005</td>
<td>en-us</td>
<td>value-length</td>
</tr>
<tr>
<td>0x45</td>
<td>uri type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x000B</td>
<td>printer-uri</td>
<td>name</td>
</tr>
<tr>
<td>0x0015</td>
<td>ipp://forest/pinetree</td>
<td>value-length</td>
</tr>
<tr>
<td>0x42</td>
<td>nameWithoutLanguage type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0008</td>
<td>job-name</td>
<td>name</td>
</tr>
<tr>
<td>0x0006</td>
<td>foobar</td>
<td>value</td>
</tr>
<tr>
<td>0x22</td>
<td>boolean type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0016</td>
<td>ipp-attribute-fidelity</td>
<td>name</td>
</tr>
<tr>
<td>0x0001</td>
<td>true</td>
<td>value-length</td>
</tr>
</tbody>
</table>

Herriot, et al. Standards Track [Page 33]
13.2 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'.

<table>
<thead>
<tr>
<th>Octets</th>
<th>Symbolic Value</th>
<th>Protocol field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00101</td>
<td>1.1</td>
<td>version-number</td>
</tr>
<tr>
<td>0x0000</td>
<td>successful-ok</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</td>
</tr>
<tr>
<td>0x0008</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>us-ascii</td>
<td>US-ASCII</td>
<td>value</td>
</tr>
<tr>
<td>0x48</td>
<td>natural-language type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x001B</td>
<td>name-length</td>
<td></td>
</tr>
<tr>
<td>attributes-</td>
<td>attributes-natural-language</td>
<td></td>
</tr>
<tr>
<td>0x0005</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>en-us</td>
<td>en-US</td>
<td>value</td>
</tr>
<tr>
<td>0x41</td>
<td>textWithoutLanguage type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x000E</td>
<td>name-length</td>
<td></td>
</tr>
<tr>
<td>status-message</td>
<td>status-message</td>
<td>name</td>
</tr>
<tr>
<td>0x000D</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>Octets</td>
<td>Symbolic Value</td>
<td>Protocol field</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>0x02</td>
<td>start job-attributes</td>
<td>job-attributes-tag</td>
</tr>
<tr>
<td>0x21</td>
<td>integer</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0006</td>
<td>job-id</td>
<td>name-length</td>
</tr>
<tr>
<td>0x0004</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>147</td>
<td>value</td>
<td></td>
</tr>
<tr>
<td>0x45</td>
<td>uri type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0007</td>
<td>name-length</td>
<td></td>
</tr>
<tr>
<td>0x0019</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>ipp://</td>
<td>job 123 on</td>
<td>value</td>
</tr>
<tr>
<td></td>
<td>pinetree/123</td>
<td></td>
</tr>
<tr>
<td>0x23</td>
<td>enum type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0009</td>
<td>name-length</td>
<td></td>
</tr>
<tr>
<td>0x0004</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>0x0003</td>
<td>pending</td>
<td>value</td>
</tr>
<tr>
<td>0x03</td>
<td>end-of-attributes</td>
<td>end-of-attributes-tag</td>
</tr>
</tbody>
</table>

13.3 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).

<table>
<thead>
<tr>
<th>Octets</th>
<th>Symbolic Value</th>
<th>Protocol field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0101</td>
<td>1.1</td>
<td>version-number</td>
</tr>
<tr>
<td>0x040B</td>
<td>client-error-attributes-or-values-not-supported</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</td>
</tr>
<tr>
<td>0x0008</td>
<td>us-ascii</td>
<td>US-ASCII</td>
</tr>
</tbody>
</table>

value-length
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>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>attributes-charset</td>
<td>value-length</td>
</tr>
<tr>
<td>0x48</td>
<td>natural-language type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x001B</td>
<td>attributes-natural-language</td>
<td>name-length</td>
</tr>
<tr>
<td>0x0005</td>
<td>en-us</td>
<td>value-length</td>
</tr>
<tr>
<td>0x41</td>
<td>textWithoutLanguage type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x000E</td>
<td>status-message</td>
<td>name-length</td>
</tr>
<tr>
<td>0x002F</td>
<td>successful-ok-ignored-or-substituted-attributes</td>
<td>value</td>
</tr>
<tr>
<td>0x05</td>
<td>start unsupported-attributes</td>
<td>unsupported-attributes tag</td>
</tr>
<tr>
<td>0x21</td>
<td>integer type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0006</td>
<td>copies</td>
<td>name-length</td>
</tr>
<tr>
<td>0x0004</td>
<td>copies</td>
<td>name</td>
</tr>
<tr>
<td>0x00000014</td>
<td>20</td>
<td>value</td>
</tr>
<tr>
<td>0x10</td>
<td>unsupported (type)</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0005</td>
<td>sides</td>
<td>name-length</td>
</tr>
<tr>
<td>0x0000</td>
<td>sides</td>
<td>name</td>
</tr>
<tr>
<td>0x02</td>
<td>start job-attributes</td>
<td>job-attributes-tag</td>
</tr>
<tr>
<td>0x21</td>
<td>integer</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0006</td>
<td>job-id</td>
<td>name</td>
</tr>
<tr>
<td>0x0004</td>
<td>147</td>
<td>value-length</td>
</tr>
<tr>
<td>0x45</td>
<td>uri type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0007</td>
<td>job-uri</td>
<td>name</td>
</tr>
<tr>
<td>0x0019</td>
<td>job-uri</td>
<td>value-length</td>
</tr>
<tr>
<td>ipp://forest/pinetree/123</td>
<td>job 123 on pinetree</td>
<td>value</td>
</tr>
</tbody>
</table>
Octets | Symbolic Value | Protocol field
---|---|---
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

13.5 Print-URI Request

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

Octets | Symbolic Value | Protocol field
---|---|---
0x0101 | 1.1 | version-number
0x0003 | Print-URI | operation-id
0x00000001 | 1 | request-id
0x01 | start operation-attributes | operation-attributes-tag
0x47 | charset type | value-tag
0x0012 | name-length | 
attributes-charset | attributes-charset | name
0x0008 | value-length | 
us-ascii | US-ASCII | value
0x48 | natural-language type | value-tag
0x001B | name-length | 
attributes-natural-language | attributes-natural-language | name
0x0005 | value-length | 
en-us | en-US | value
0x45 | uri type | value-tag
0x000B | name-length | 
printer-uri | printer-uri | name
0x0015 | value-length | 
ipp://forest/pinetree | printer-pinetree | value
0x45 | uri type | value-tag
0x000C | name-length | 
document-uri | document-uri | name
0x0011 | value-length | 
13.6 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   attributes-charset           name
0x0008   US-ASCII                     value-length
0x48     natural-language type        value-tag
0x001B   attributes-natural-language  name
0x0005   en-US                         value-length
0x45     uri type                     value-tag
0x000B   name-length                  value
0x00015  printer-pinetree             value-length
ipp://forest/  printer pinetree      value
### 13.7 Get-Jobs Request

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

<table>
<thead>
<tr>
<th>Octets</th>
<th>Symbolic Value</th>
<th>Protocol field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0101</td>
<td>1.1</td>
<td>version-number</td>
</tr>
<tr>
<td>0x000A</td>
<td>Get-Jobs</td>
<td>operation-id</td>
</tr>
<tr>
<td>0x00000123</td>
<td>0x123</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>attributes-charset</td>
<td></td>
<td>name</td>
</tr>
<tr>
<td>0x0008</td>
<td>us-ascii</td>
<td>value-length</td>
</tr>
<tr>
<td>0x48</td>
<td>natural-language type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x001B</td>
<td>attributes-natural-language</td>
<td>name-length</td>
</tr>
<tr>
<td>attributes-natural-language</td>
<td></td>
<td>name</td>
</tr>
<tr>
<td>0x0005</td>
<td>en-us</td>
<td>value-length</td>
</tr>
<tr>
<td>0x45</td>
<td>uri type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x000B</td>
<td>printer-uri</td>
<td>name-length</td>
</tr>
<tr>
<td>printer-uri</td>
<td>printer-uri</td>
<td>name</td>
</tr>
<tr>
<td>0x0015</td>
<td>ipp://forest/pinetree</td>
<td>value-length</td>
</tr>
<tr>
<td>0x21</td>
<td>integer type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0005</td>
<td>limit</td>
<td>name-length</td>
</tr>
<tr>
<td>limit</td>
<td></td>
<td>name</td>
</tr>
<tr>
<td>0x0004</td>
<td>50</td>
<td>value-length</td>
</tr>
<tr>
<td>0x44</td>
<td>keyword type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0014</td>
<td>requested-attributes</td>
<td>name-length</td>
</tr>
<tr>
<td>requested-attributes</td>
<td></td>
<td>name</td>
</tr>
<tr>
<td>0x0006</td>
<td></td>
<td>value-length</td>
</tr>
</tbody>
</table>
13.8 Get-Jobs Response

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

Octets          Symbolic Value                Protocol field

0x0101          1.1                           version-number
0x0000          successful-ok              status-code
0x00000123      0x123                       request-id (echoed back)
0x01           start operation-attributes   operation-attributes-tag
0x47            charset type                value-tag
0x0012          attributes-charset         name-length
attributes-charset
0x000A          value-length
ISO-8859-1      ISO-8859-1                  value
0x48            natural-language type      value-tag
0x001B          name-length
attributes-natural-language
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   job-attributes-tag

RFC 2910          IPP/1.1: Encoding and Transport       September 2000

Octets       Symbolic Value                  Protocol field

<table>
<thead>
<tr>
<th>Octets</th>
<th>Symbolic Value</th>
<th>Protocol field</th>
</tr>
</thead>
<tbody>
<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>job-id</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td>0x0004</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>147</td>
<td>value</td>
<td></td>
</tr>
<tr>
<td>0x36</td>
<td>nameWithLanguage</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0008</td>
<td>name-length</td>
<td></td>
</tr>
<tr>
<td>job-name</td>
<td>job-name</td>
<td></td>
</tr>
<tr>
<td>0x000C</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>0x0005</td>
<td>sub-value-length</td>
<td></td>
</tr>
<tr>
<td>fr-ca</td>
<td>fr-CA</td>
<td></td>
</tr>
<tr>
<td>0x0003</td>
<td>sub-value-length</td>
<td></td>
</tr>
<tr>
<td>fou</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>start job-attributes (2nd</td>
<td>job-attributes-tag</td>
</tr>
<tr>
<td>object)</td>
<td>value-tag</td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>start job-attributes (3rd</td>
<td>job-attributes-tag</td>
</tr>
<tr>
<td>object)</td>
<td>value-tag</td>
<td></td>
</tr>
<tr>
<td>0x21</td>
<td>integer type</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0006</td>
<td>name-length</td>
<td></td>
</tr>
<tr>
<td>job-id</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td>0x0004</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>value</td>
<td></td>
</tr>
<tr>
<td>0x36</td>
<td>nameWithLanguage</td>
<td>value-tag</td>
</tr>
<tr>
<td>0x0008</td>
<td>name-length</td>
<td></td>
</tr>
<tr>
<td>job-name</td>
<td>job-name</td>
<td></td>
</tr>
<tr>
<td>0x0012</td>
<td>value-length</td>
<td></td>
</tr>
<tr>
<td>0x0005</td>
<td>sub-value-length</td>
<td></td>
</tr>
<tr>
<td>de-CH</td>
<td>de-CH</td>
<td></td>
</tr>
<tr>
<td>0x0009</td>
<td>sub-value-length</td>
<td></td>
</tr>
<tr>
<td>isch guet</td>
<td>isch guet</td>
<td>name</td>
</tr>
<tr>
<td>0x03</td>
<td>end-of-attributes</td>
<td>end-of-attributes-tag</td>
</tr>
</tbody>
</table>

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

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

MIME type name: application

MIME subtype name: ipp
A Content-Type of "application/ipp" indicates an Internet Printing Protocol message body (request or response). Currently there is one version: IPP/1.1, whose syntax is described in Section 3 "Encoding of the Operation Layer" of [RFC2910], and whose semantics are described in [RFC2911].

Required parameters: none

Optional parameters: none

Encoding considerations:

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

Security considerations:

IPP/1.1 protocol requests/responses do not introduce any security risks not already inherent in the underlying transport protocols. Protocol mixed-version interworking rules in [RFC2911] as well as protocol encoding rules in [RFC2910] are complete and unambiguous.

Interoperability considerations:

IPP/1.1 requests (generated by clients) and responses (generated by servers) MUST comply with all conformance requirements imposed by the normative specifications [RFC2911] and [RFC2910]. Protocol encoding rules specified in [RFC2910] are comprehensive, so that interoperability between conforming implementations is guaranteed (although support for specific optional features is not ensured). Both the "charset" and "natural-language" of all IPP/1.1 attribute values which are a LOCALIZED-STRING are explicit within IPP protocol requests/responses (without recourse to any external information in HTTP, SMTP, or other message transport headers).

Published specifications:


Applications which use this media type:

Internet Printing Protocol (IPP) print clients and print servers, communicating using HTTP/1.1 (see [RFC2910]), SMTP/ESMTP, FTP, or other transport protocol. Messages of type "application/ipp" are self-contained and transport-independent, including "charset" and "natural-language" context for any LOCALIZED-STRING value.

Person & email address to contact for further information:

Tom Hastings
Xerox Corporation
737 Hawaii St. ESAE-231
El Segundo, CA
Phone: 310-333-6413
Fax: 310-333-5514
EMail: hastings@cp10.es.xerox.com

or

Robert Herriot
Xerox Corporation
3400 Hillview Ave., Bldg #1
Palo Alto, CA 94304
Phone: 650-813-7696
Fax: 650-813-6860
EMail: robert.herriot@pahv.xerox.com

Intended usage:

COMMON

15. Appendix C: Changes from IPP/1.0

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

1. Attributes values that identify a printer or job object use a new 'ipp' scheme. The 'http' and 'https' schemes are supported only for backward compatibility. See section 5.

2. Clients MUST support of Digest Authentication, IPP Printers SHOULD support Digest Authentication. See Section 8.1.1

3. TLS is recommended for channel security. In addition, SSL3 may be supported for backward compatibility. See Section 8.1.2
4. It is recommended that IPP/1.1 objects accept any request with major version number '1'. See section 9.1.

5. IPP objects SHOULD return the URL scheme requested for "job-printer-uri" and "job-uri" Job Attributes, rather than the URL scheme used to create the job. See section 9.2.

6. The IANA and Internationalization sections have been added. The terms "private use" and "experimental" have been changed to "vendor extension". The reserved allocations for attribute group tags, attribute syntax tags, and out-of-band attribute values have been clarified as to which are reserved to future IETF standards track documents and which are reserved to vendor extension. Both kinds of extensions use the type2 registration procedures as defined in [RFC2911].

7. Clarified that future "out-of-band" value definitions may use the value field if additional information is needed.
Full Copyright Statement

Copyright (C) The Internet Society (2000). 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.

Acknowledgement

Funding for the RFC Editor function is currently provided by the
Internet Society.