

Robert Herriot (editor)  
Sun Microsystems  
Sylvan Butler  
Hewlett-Packard  
Paul Moore  
Microsoft.  
Randy Turner  
Sharp Labs  
October 23, 1997

Internet Printing Protocol/1.0: Protocol Specification  
draft-ietf-ipp-protocol-02.txt

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43

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 "lid-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

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 technology. The protocol is heavily influenced by the printing model introduced in the Document Printing Application (ISO/IEC 10175 DPA) standard [dpa]. Although DPA specifies both end user and administrative features, IPP version 1.0 is focused only on end user functionality.

The full set of IPP documents includes:

- Internet Printing Protocol: Requirements
- Internet Printing Protocol/1.0: Model and Semantics
- Internet Printing Protocol/1.0: Protocol Specification

The requirements document 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. The requirements document calls out a subset of end user requirements that MUST be satisfied in the first version of IPP. Operator and administrator requirements are out of scope for v1.0. The model and semantics document describes a simplified model with abstract objects, their attributes, and their operations. The model introduces a Printer object and a Job object. The Job object supports multiple documents per job. The protocol specification is formal document which incorporates the ideas in all the other documents into a concrete mapping using clearly defined data representations and transport protocol mappings that real implementers can use to develop interoperable client and printer (server) side components.

This document is the "Internet Printing Protocol/1.0: Protocol Specification" document.

Notice

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

47 Copyright © The Internet Society (date). All Rights Reserved.

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

55 The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

56 This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND  
57 THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING  
58 BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE  
59 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR  
60 PURPOSE.

## 61 Table of Contents

62	1. Introduction .....	4
63	2. Conformance Terminology .....	4
64	3. Encoding of the Operation Layer.....	4
65	3.1 Picture of the Encoding .....	4
66	3.2 Syntax of Encoding .....	6
67	3.3 Version .....	7
68	3.4 Mapping of Operations .....	7
69	3.5 Mapping of Status-code .....	8
70	3.6 Tags .....	8
71	3.6.1 Delimiter Tags .....	8
72	3.6.2 Value Tags.....	9
73	3.7 Name-Lengths.....	10
74	3.8 Mapping of Attribute Names.....	10
75	3.9 Value Lengths .....	11
76	3.10 Mapping of Attribute Values.....	11
77	3.11 Data .....	13
78	4. Encoding of Transport Layer .....	13
79	4.1 General Headers .....	14
80	4.2 Request Headers.....	14
81	4.3 Response Headers .....	15
82	4.4 Entity Headers.....	15
83	5. Security Considerations .....	16
84	6. References .....	16
85	7. Author's Address.....	17
86	8. Other Participants: .....	18
87	9. Appendix A: Protocol Examples .....	18
88	9.1 Print-Job Request .....	18
89	9.2 Print-Job Response (successful).....	19
90	9.3 Print-Job Response (failure).....	20
91	9.4 Print-URI Request.....	21
92	9.5 Create-Job Request .....	21
93	9.6 Get-Jobs Request.....	22
94	9.7 Get-Jobs Response .....	22
95	10. Appendix B: Mapping of Each Operation in the Encoding .....	23
96	11. Appendix C: Hints to implementors using IPP with SSL3 .....	26
97		
98		
99		

## 100 **1. Introduction**

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

103 The transport layer consists of an HTTP/1.1 request or response. RFC 2068 [r2068] describes HTTP/1.1. This document  
104 specifies the HTTP headers that an IPP implementation supports.

105 The operation layer consists of a message body in an HTTP request or response. The document "Internet Printing Protocol/1.0:  
106 Model and Semantics" [ipp-m] defines the semantics of such a message body and the supported values. This document specifies  
107 the encoding of an IPP operation. The aforementioned document [ipp-m] is henceforth referred to as the "IPP model document"

## 108 **2. Conformance Terminology**

109 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT",  
110 "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [r2119].

## 111 **3. Encoding of the Operation Layer**

112 The operation layer SHALL contain a single operation request or operation response.

113 The encoding consists of octets as the most primitive type. There are several types built from octets, but two important types are  
114 integers and characters, on which most other data types are built. Every character in this encoding SHALL be a member of the  
115 UCS-2 coded character set and SHALL be encoded using UTF-8 which uses 1 to 3 octets per character. Every integer in this  
116 encoding SHALL be encoded as a signed integer using two's-complement binary encoding with big-endian format (also known as  
117 "network order" and "most significant byte first"). The number of octets for an integer SHALL be 1, 2 or 4, depending on usage  
118 in the protocol. Such one-octet integers, henceforth called SIGNED-BYTE, are used for the version and tag fields. Such two-byte  
119 integers, henceforth called SIGNED-SHORT are used for the operation, status-code and length fields. Four byte integers,  
120 henceforth called SIGNED-INTEGER, are used for values fields.

121 The following two sections present the operation layer in two ways

- 122 • informally through pictures and description
- 123 • formally through Augmented Backus-Naur Form (ABNF), as specified by draft-ietf-drums-abnf-02.txt [abnf]

### 124 **3.1 Picture of the Encoding**

125 The encoding for an operation request or response consists of:

126	-----		
127		version	2 bytes - required
128	-----		
129		operation (request) or status-code (response)	2 bytes - required
130	-----		
131		xxx-attributes-tag	1 byte
132	-----		-0 or more
133		xxx-attribute-sequence	n bytes
134	-----		
135		data-tag	1 byte - required
136	-----		
137		data	q bytes - optional
138	-----		

139 The xxx-attributes-tag and xxx-attribute-sequence represents four different values of “xxx”, namely, operation, job, printer and  
 140 unsupported-job. The xxx-attributes-tag and xxx-attribute-sequence may be omitted if the operation has no attributes or it may be  
 141 repeated with the same or different values of “xxx” in ways that are specific to each operation. The data is omitted from some  
 142 operations, but the data-tag is present even when the data is omitted. Note, the xxx-attributes-tags and data-tag are called  
 143 ‘delimiter-tags’.

144 Note: the xxx-attribute-sequence, shown above may consist of 0 bytes, according to the rule below.

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

146	-----		
147		compound-attribute	s bytes - 0 or more
148	-----		

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

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

152 Each attribute consists of:

153	-----		
154		value-tag	1 byte
155	-----		
156		name-length (value is u)	2 bytes
157	-----		
158		name	u bytes
159	-----		
160		value-length (value is v)	2 bytes
161	-----		
162		value	v bytes
163	-----		

164 An additional value consists of:

165	-----		
166		value-tag	1 byte
167	-----		
168		name-length (value is 0x0000)	2 bytes
169	-----		
170		value-length (value is w)	2 bytes
171	-----		
172		value	w bytes
173	-----		
174			-0 or more

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

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

177	-----		
178		version	2 bytes - required
179	-----		
180		operation (request) or status-code (response)	2 bytes - required
181	-----		
182		tag (delimiter-tag or value-tag)	1 byte
183	-----		
184		empty or rest of attribute	x bytes
185	-----		
186		data-tag	2 bytes - required
187	-----		
188		data	y bytes - optional
189	-----		
190			-0 or more

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

- 192 • attributes
- 193 • data
- 194 • the remainder of a single attribute where the tag specifies the type of the value.

### 195 3.2 Syntax of Encoding

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

```

199 ipp-message = ipp-request / ipp-response
200 ipp-request = version operation
201             *(xxx-attributes-tag xxx-attribute-sequence) data-tag data
202 ipp-response = version status-code
203             *(xxx-attributes-tag xxx-attribute-sequence) data-tag data
204 xxx-attribute-sequence = *compound-attribute
205             ; where "xxx" in the three rules above stands for any of the following
206             ; values: "operation", "job", "printer" or "unsupported-job".
207
208
209 version = major-version minor-version
210 major-version = SIGNED-BYTE ; initially %d1
211 minor-version = SIGNED-BYTE ; initially %d0

```

212  
 213 operation = SIGNED-SHORT ; mapping from model defined below  
 214 status-code = SIGNED-SHORT ; mapping from model defined below  
 215  
 216 compound-attribute = attribute \*additional-values  
 217  
 218 attribute = value-tag name-length name value-length value  
 219 additional-values = value-tag zero-name-length value-length value  
 220  
 221 name-length = SIGNED-SHORT ; number of octets of 'name'  
 222 name = LALPHA \*( LALPHA / DIGIT / "-" / "\_" / "." )  
 223 value-length = SIGNED-SHORT ; number of octets of 'value'  
 224 value = OCTET-STRING  
 225  
 226 data = OCTET-STRING  
 227  
 228 zero-name-length = %x00.00 ; name-length of 0  
 229 operation-attributes-tag = %x01 ; tag of 1  
 230 job-attributes-tag = %x02 ; tag of 2  
 231 printer-attributes-tag = %x043 ; tag of 43  
 232 unsupported-job-attributes-tag = %x054 ; tag of 54  
 233 data-tag = %x03 ; tag of 3  
 234 value-tag = %x10-FF  
 235  
 236 SIGNED-BYTE = BYTE  
 237 SIGNED-SHORT = 2BYTE  
 238 DIGIT = %x30-39 ; "0" to "9"  
 239 LALPHA = %x61-7A ; "a" to "z"  
 240 BYTE = %x00-FF  
 241 OCTET-STRING = \*BYTE  
 242

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

### 247 3.3 Version

248 The version SHALL consist of a major and minor version, each of which SHALL be represented by a SIGNED-BYTE. The  
 249 protocol described in this document SHALL have a major version of 1 (0x01) and a minor version of 0 (0x00). The ABNF for  
 250 these two bytes SHALL be %x01.00.

### 251 3.4 Mapping of Operations

252 Operations are defined as enums in the model document. An operations enum value SHALL be encoded as a SIGNED-SHORT

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

## 254 3.5 Mapping of Status-code

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

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

## 258 3.6 Tags

259 There are two kinds of tags:

- 260 • delimiter tags: delimit major sections of the protocol, namely attributes and data
- 261 • value tags: specify the type of each attribute value

### 262 3.6.1 Delimiter Tags

263 The following table specifies the values for the delimiter tags:

Tag Value (Hex)	Delimiter
0x00	reserved
0x01	operation-attributes-tag
0x02	job-attributes-tag
0x03	data-tag
0x04	printer-attributes-tag
0x05	unsupported-job-attributes-tag
0x06-0x0F	reserved for future delimiters

264

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

267 Doing substitution for xxx in the above paragraph, this means the following. When an operation-attributes-tag occurs in the  
268 protocol, it SHALL mean that the zero or more following attributes up to the next delimiter tag are operation attributes as defined  
269 in the model document. When an job-attributes-tag occurs in the protocol, it SHALL mean that the zero or more following  
270 attributes up to the next delimiter tag are job attributes as defined in the model document. When an printer-attributes-tag occurs in  
271 the protocol, it SHALL mean that the zero or more following attributes up to the next delimiter tag are printer attributes as  
272 defined in the model document. When an unsupported-job-attributes-tag occurs in the protocol, it SHALL mean that the zero or  
273 more following attributes up to the next delimiter tag are unsupported-job attributes as defined in the model document.

274 ~~The operation-attributes-tag SHALL precede those attributes defined as operation attributes in the model document. The job-~~  
275 ~~attributes-tag SHALL precede those attributes defined as job attributes in the model document. The printer-attributes-tag SHALL~~  
276 ~~precede those attributes defined as printer attributes in the model document. The unsupported-job-attributes-tag SHALL precede~~  
277 ~~those attributes defined as unsupported-job attributes in the model document.~~

278 Each of the four xxx-attributes-tags defined above is OPTIONAL in an operation and each SHALL occur at most once in an  
279 operation, except for job-attributes-tag in a Get-Jobs response which may occur zero or more times. The data-tag SHALL occur  
280 exactly once in an operation.

281 If an operation contains an operations-attribute-tag, it SHALL be the first tag delimiter. The data-tag SHALL be the last tag  
282 delimiter.



283 The order and presence of delimiter tags for each operation request and each operation response SHALL be that defined in the  
 284 model document. For further details, see Section 3.8 Mapping of Attribute Names and Appendix B: Mapping of Each Operation  
 285 in the Encoding.

### 286 3.6.2 Value Tags

287 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  
 288 value of the attribute. The value of the value-tag of an attribute SHALL either be a type value specified in the model document or  
 289 an “out-of-band” value, such as “unsupported” or “default”. If the value of value-tag for an attribute is not “out-of-band” and  
 290 differs from the value type specified by the model document, then a printer receiving such a request MAY reject the attribute or  
 291 just the value.it, and a A client receiving such a response MAY ignore the attribute or just the value.

292 If ipp-attribute-fidelity is true and a printer rejects a value, it is the same as rejecting the attribute. If ipp-attribute-fidelity is false  
 293 and a printer rejects a value, or it a client rejects a value, then it is as if the attribute didn't have that value. If after rejecting  
 294 values, the attribute no longer has any values the attribute is rejected.

295 Note: the intent of the above rule is for servers to be able to understand text and name values when they don't support the  
 296 naturalLanguage override for the value.

297 The following table specifies the “out-of-band” values for the value-tag.

Tag Value (Hex)	Meaning
0x10	unsupported
0x11	default
0x12	no-value
<u>0x13</u>	<u>compoundValue</u>
0x14-0x1F	reserved for future “out-of-band” values.

298 The “unsupported” value SHALL be used in the attribute-sequence of an error response for those attributes which the printer does  
 299 not support. The “default” value is reserved for future use of setting value back to their default value. The “no-value” value is  
 300 used for the “no-value” value in model, e.g. when a document-attribute is returned as a set of values and an attribute has no  
 301 specified value for one or more of the documents. The “compoundValue” SHALL be used to form a single value from a  
 302 collection of values, and its value is the number of members forming the compound value, excluding the compoundValue. For  
 303 example, a text value with a naturalLanguage override consists of 3 “values”: a compoundValue with value 2, a naturalLanguage  
 304 value and a text value.

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

Tag Value (Hex)	Meaning
0x20	reserved
0x21	integer
0x22	boolean
0x23	enum
0x24-0x2F	reserved for future integer types

306 NOTE: 0x20 is reserved for “generic integer” if should ever be needed.

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

Tag Value (Hex)	Meaning
0x30	octetString with an unspecified format
0x31	dateTime
0x32	resolution
0x33	rangeOfInteger
0x34	reserved for dictionary (in the future)
0x35-0x3F	reserved for future octetString types

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

Tag Value (Hex)	Meaning
0x40	reserved
0x41	text
0x42	name
0x43	language
0x44	keyword
0x45	uri
0x46	uriScheme
0x47	charSet
0x48 <sup>7</sup>	naturalLanguage
0x49 <sup>8</sup>	mediaType
0x4A <sup>9</sup> -0x5F	reserved for future character string types

309 NOTE: 0x40 is reserved for “generic character-string” if should ever be needed.

310 The values 0x60-0xFF are reserved for future types. There are no values allocated for private extensions. A new type must be  
311 registered via the type 2 process.

### 312 3.7 Name-Lengths

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

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

### 318 3.8 Mapping of Attribute Names

319 Some attributes are encoded in a special position. These attribute are:

- 320 • “printer-uri”: The target printer-uri of each operation in the IPP model document SHALL be specified outside of the  
321 operation layer as the request-URI on the Request-Line at the HTTP level.
- 322 • “job-uri”: The target job-uri of each operation in the IPP model document SHALL be specified outside of the operation  
323 layer as the request-URI on the Request-Line at the HTTP level.
- 324 • “document-content”: The attribute named “document-content” in the IPP model document SHALL become the “data”  
325 in the operation layer.
- 326 • “status-code”: The attribute named “status-code” in the IPP model document SHALL become the “status-code” field in  
327 the operation layer response.

328 The model document arranges the remaining attributes into groups for each operation request and response. Each such group  
 329 SHALL be represented in the protocol by an xxx-attribute-sequence preceded by the appropriate xxx-attributes-tag (See the table  
 330 below and Appendix B: Mapping of Each Operation in the Encoding). In addition, the order of these xxx-attributes-tags and xxx-  
 331 attribute-sequences in the protocol SHALL be the same as in the model document, but the order of attributes within each xxx-  
 332 attribute-sequence SHALL be unspecified. The table below maps the model document group name to xxx-attributes-sequence

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

333 ISSUE: coordinate this with the model document.

334 If an operation contains attributes from more than one job object (e.g. Get-Jobs response), the attributes from each job object  
 335 SHALL be in a separate job-attribute-sequence, such that the attributes from the ith job object are in the ith job-attribute-  
 336 sequence. See Section 10 “Appendix B: Mapping of Each Operation in the Encoding” for table showing the application of the  
 337 rules above.

### 338 3.9 Value Lengths

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

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

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

344 If a value-tag contains an “out-of-band” value, such as “unsupported”, the value-length SHALL be 0 and the value empty — the  
 345 value has no meaning when the value-tag has an “out-of-band” value. If a printer or client receives an operation with a nonzero  
 346 value-length in this case, it SHALL ignore the value field.

### 347 3.10 Mapping of Attribute Values

348 The following SHALL be the mapping of attribute values to their IPP encoding in the value field. The syntax types are defined in  
 349 the IPP model document.

Syntax of Attribute Value	Encoding
---------------------------	----------

text	an octet string whose <u>charset and language is that specified within the operation request or response.</u> <u>The attributes-charset attribute with a value of type ‘charset’ MUST be in the operations-attribute sequence unless the request or response contains no attributes. It specifies the charset for all text and name values of attributes.</u> <u>The attributes-natural-language attribute with a value of type ‘naturalLanguage’ MUST be in the operations-attribute sequence unless the request or response contains no attributes. It specifies the language for all text and name values of attributes unless</u>
------	---

**Syntax of Attribute Value      Encoding**

	<p><u>overridden. The language can be overridden on a per-object or a per-value basis. The language can be overridden on a per-object basis only for a job-sequence within a Get-Jobs response. If the attributes-natural-language attribute is present within such a context, it must have a value of type 'naturalLanguage' and it overrides the language for all text and name attributes within the job-attributes sequence containing it, but not for attributes in any other xxx-attribute-sequence.</u></p> <p><u>The language can be overridden on a per-value basis by syntactically preceding the text or name value by two values: a value of type compoundValue whose value is 2 and a value of type naturalLanguage whose value is the language override. From a protocol syntax view, the compoundValue, the naturalLanguage value and the text or name value appear as three separate values of a single attribute, but from a semantic view, the Printer treats them as a single value where the naturalLanguage value overrides the language of the immediately following text or name value in the attribute. Any text or name values in the same or other attributes are not affected by the override. If an attribute consists of a single text or name value, the language value turns it into an attribute with two values from a syntactic view.</u></p> <p><u>syntax comes from RFC 2184 [r2184] and is:</u></p> <p><del>— [character set] " ' " [language] " ' " octet string</del></p> <p><u>The character set, language and the quote characters " ' " are all in US-ASCII. The character set specifies the encoding of the octet string, and the language specifies the language of the octet string. If character set is omitted, it defaults to the explicit character set for the operation request or response. If language is omitted, it defaults to the explicit language for the operation request or response.</u> The text is encoded in "network byte order" with the first character in the text (according to reading order) being the first character in the encoding.</p>
name	same as text
<u>charset</u>	<u>an octet string of US-ASCII encoded characters specified in RFC 2046 [r2046] and contained in the IANA character-set Registry [iana] according to the IANA procedures [char].</u>
<u>naturalLanguage</u>	an octet string of US-ASCII encoded characters and with a syntax specified by RFC 1766 [r1766]
<u>mediaType</u>	<u>an octet string of US-ASCII encoded characters defined by RFC 2046 [r2046] and registered according to the procedures of RFC 2048 [r2048] for identifying a document format. The value MAY include a charset parameter, depending on the specification of the Media Type in the IANA Registry [iana]. Examples:</u>
keyword	an octet string of US-ASCII encoded characters. . Allowed values are defined in the IPP model document
uri	as specified by RFC 1630 [r1630]
uriScheme	same as keyword
boolean	one binary octet where 0x00 is 'false' and 0x01 is 'true'
integer	a SIGNED-INTEGER, defined previously as a signed integer using two's-complement binary encoding in four octets with big-endian format (also known as "network order" and "most significant byte first").
enum	same as integer. Allowed integer values are defined in the IPP model document
<u>compoundValue</u>	<u>has the same representation as an integer, but with a different meaning. If the value of a compoundValue is n, then the n following values of the attribute form a single value. For example, if an attribute has 3 successive values: compoundValue of 2, naturalLanguage of 'fr-CA' and name of 'bête', then these three "values" form a single value which is a name of 'bête' in Canadian French.</u>
dateTime	eleven octets whose contents are defined by "DateAndTime" in RFC 1903 [r1903]. Although RFC 1903 also defines an eight octet format which omits the time zone, a value of this type in the IPP protocol MUST use the eleven octet format.

**Syntax of Attribute Value      Encoding**

resolution	nine octets consisting of 2 SIGNED-INTEGERS followed by a SIGNED-BYTE. The values are the same as those specified in RFC 1759 (Printer MIB) [r1759]. The first SIGNED-INTEGER contains the value of prtMarkerAddressabilityXFeedDir. The second SIGNED-INTEGER contains the value of prtMarkerAddressabilityFeedDir. The SIGNED-BYTE contains the value of prtMarkerAddressabilityUnit. Note: the latter value is either 3 (tenThousandsOfInches) or 4 (micrometers) and the addressability is in 10,000 units of measure. Thus the SIGNED-INTEGERS represent integral values in either dots-per-inch or dots-per-centimeter.
rangeOf integer	Eight octets consisting of 2 SIGNED-INTEGERS. The first SIGNED-INTEGERS contains the lowest value of the range and the second SIGNED-INTEGERS contains the highest value of the range
1setOf X	encoding according to the rules for an attribute with more than more value. Each value X is encoded according to the rules for encoding its type.

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

351 **3.11 Data**

352 The data part SHALL include any data required by the operation

353 **4. Encoding of Transport Layer**

354 HTTP/1.1 shall be the transport layer for this protocol.

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

- 356
- 357 • the URI of the target job or printer operation
  - 358 • 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.
- 358 ~~• the client's language, the character-set and the transport encoding.~~

359 It is REQUIRED that a printer support HTTP over port 80, though a printer may support HTTP over port 516 or some other port.  
360 In addition, a printer may have to support another port for secure connections.

361 Note: Consistent with RFC 2068 (HTTP/1.1), HTTP URI's for IPP implicitly reference port 80. If a URI references some other  
362 port, the port number must be explicitly specified in the URI.

363 Each HTTP operation shall use the POST method where the request-URI is the object target of the operation, and where the  
364 "Content-Type" of the message-body in each request and response shall be "application/ipp". The message-body shall contain the  
365 operation layer and shall have the syntax described in section 3.2 "Syntax of Encoding". A client implementation SHALL adhere  
366 to the rules for a client described in RFC 2068 [r2068]. A printer (server) implementation SHALL adhere the rules for an origin  
367 server described in RFC 2068. In the following sections, there are a tables of all HTTP headers which describe their use in an IPP  
368 client or server. The following is an explanation of each column in these tables.

- 369
- 370 • the "header" column contains the name of a header
  - 371 • the "request/client" column indicates whether a client sends the header.
  - 372 • the "request/ server" column indicates whether a server supports the header when received.
  - 373 • the "response/ server" column indicates whether a server sends the header.
  - 373 • the "response /client" column indicates whether a client supports the header when received.

- 374       • the “values and conditions” column specifies the allowed header values and the conditions for the header to be present in  
375       a request/response.

376       The table for “request headers” does not have columns for responses, and the table for “response headers” does not have columns  
377       for requests.

378       The following is an explanation of the values in the “request/client” and “response/ server” columns.

- 379       • **must:** the client or server **MUST** send the header,
- 380       • **must-if:** the client or server **MUST** send the header when the condition described in the “values and conditions” column  
381       is met,
- 382       • **may:** the client or server **MAY** send the header
- 383       • **not:** the client or server **SHOULD NOT** send the header. It is not relevant to an IPP implementation.

384       The following is an explanation of the values in the “response/client” and “request/ server” columns.

- 385       • **must:** the client or server **MUST** support the header,
- 386       • **may:** the client or server **MAY** support the header
- 387       • **not:** the client or server **SHOULD NOT** support the header. It is not relevant to an IPP implementation.

## 388    4.1 General Headers

389       The following is a table for the general headers.

390       ISSUE: an HTTP expert should review these tables for accuracy.

General-Header	Request		Response		Values and Conditions
	Client	Server	Server	Client	
Cache-Control	must	not	must	not	“no-cache” only
Connection	must-if	must	must-if	must	“close” only. Both client and server SHOULD keep a connection for the duration of a sequence of operations. The client and server <b>MUST</b> include this header for the last operation in such a sequence.
Date	may	may	must	may	per RFC 1123 [r1123]
Pragma`	must	not	must	not	“no-cache” only
Transfer-Encoding	must-if	must	must-if	must	“chunked” only . Header <b>MUST</b> be present if Content-Length is absent.
Upgrade	not	not	not	not	
Via	not	not	not	not	

391

## 392    4.2 Request Headers

393       The following is a table for the request headers.

394

Request-Header	Client	Server	Request Values and Conditions
Accept	may	must	“application/ipp” only. This value is the default if the client omits it

Request-Header	Client	Server	Request Values and Conditions
Accept-Charset	<u>notmay</u>	<u>notmust</u>	<del>per IANA Character Set registry. —ISSUE: is this useful for IPP? Charset information is within the application/ipp entity</del>
Accept-Encoding	may	must	empty and per RFC 2068 [r2068] and IANA registry for content-codings
Accept-Language	<u>notmay</u>	<u>notmust</u>	<del>see RFC 1766 [r1766]. A server SHOULD honor language requested by returning the values status-message, job-state-message and printer-state-reason in one of requested languages. language information is within the application/ipp entity</del>
Authorization	must-if	must	per RFC 2068. A client MUST send this header when it receives a 401 “Unauthorized” response and does not receive a “Proxy-Authenticate” header.
From	not	not	per RFC 2068. Because RFC recommends sending this header only with the user’s approval, it is not very useful
Host	must	must	per RFC 2068
If-Match	not	not	
If-Modified-Since	not	not	
If-None-Match	not	not	
If-Range	not	not	
If-Unmodified-Since	not	not	
Max-Forwards	not	not	
Proxy-Authorization	must-if	not	per RFC 2068. A client MUST send this header when it receives a 401 “Unauthorized” response and a “Proxy-Authenticate” header.
Range	not	not	
Referer	not	not	
User-Agent	not	not	

### 395 4.3 Response Headers

396 The following is a table for the request headers.

397

Response-Header	Server	Client	Response Values and Conditions
Accept-Ranges	not	not	
Age	not	not	
Location	must-if	may	per RFC 2068. When URI needs redirection.
Proxy-Authenticate	not	must	per RFC 2068
Public	may	may	per RFC 2068
Retry-After	may	may	per RFC 2068
Server	not	not	
Vary	not	not	
Warning	may	may	per RFC 2068
WWW-Authenticate	must-if	must	per RFC 2068. When a server needs to authenticate a client.

### 398 4.4 Entity Headers

399 The following is a table for the entity headers.

400

Entity-Header	Request	Response	Values and Conditions
---------------	---------	----------	-----------------------

	Client	Server	Server	Client	
Allow	not	not	not	not	
Content-Base	not	not	not	not	
Content-Encoding	may	must	must	must	per RFC 2068 and IANA registry for content codings.
Content-Language	<del>notmay</del>	<del>notmust</del>	<del>notmust</del>	<del>notmust</del>	<del>see RFC 1766 [r1766].</del> <del>Application/ipp handles language</del>
Content-Length	must-if	must	must-if	must	the length of the message-body per RFC 2068. Header MUST be present if Transfer-Encoding is absent..
Content-Location	not	not	not	not	
Content-MD5	may	may	may	may	per RFC 2068
Content-Range	not	not	not	not	
Content-Type	must	must	must	must	“application/ipp” only
ETag	not	not	not	not	
Expires	not	not	not	not	
Last-Modified	not	not	not	not	

## 401 5. Security Considerations

402 When utilizing HTTP 1.1 as a transport of IPP, the security considerations outlined in RFC 2068 [r2068] apply. Specifically, IPP  
403 servers can generate a 401 “Unauthorized” response code to request client authentication and IPP clients should correctly  
404 respond with the proper “Authorization” header. Both Basic Authentication (RFC 2068) and Digest Authentication (RFC 2069)  
405 [r2069] flavors of authentication SHALL be supported. The server chooses which type(s) of authentication to accept. Digest  
406 Authentication is a more secure method, and is always preferred to Basic Authentication.

407 For secure communication (privacy in particular), IPP SHOULD be run using a secure communications channel. For this purpose  
408 it is the intention to define standardization of IPP in combination with Transport Layer Security (TLS), currently under  
409 development in the IETF, when the TLS specifications are agreed and on the IETF standards track.

410 As an intercept solution for secure communication, the Secure Socket Layer 3.0 (SSL3) could be used, but be warned that such  
411 implementations may not be able to interoperate with a future standardized IPP and TLS solution. Appendix C gives some hints  
412 to implementors wanting to use SSL3 as intercept solution.

413 It is possible to combine the techniques, HTTP 1.1 client authentication (either basic or digest) with a secure communications  
414 channel. Together the two are more secure than client authentication and they perform user authentication.

415 See further discussion of IPP security concepts in the model document [ipp-m].

## 416 6. References

417 [822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", RFC 822, August 1982.

418 [r1123] Braden, S., "Requirements for Internet Hosts - Application and Support", RFC 1123, October, 1989,

419 [r1179] McLaughlin, L. III, (editor), "Line Printer Daemon Protocol" RFC 1179, August 1990.

420 [r1630] T. Berners-Lee, “Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and  
421 Addresses of Objects on the Network as used in the Word-Wide Web”, RFC 1630, June 1994.

422 [r1759] Smith, R., Wright, F., Hastings, T., Zilles, S., and Gyllenskog, J., "Printer MIB", RFC 1759, March 1995.



- 423 [r1738] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource Locators (URL)", RFC 1738, December, 1994.
- 424 [r1543] Postel, J., "Instructions to RFC Authors", RFC 1543, October 1993.
- 425 [r1766] H. Alvestrand, " Tags for the Identification of Languages", RFC 1766, March 1995.
- 426 [r1903] J. Case, et al. "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903,  
427 January 1996.
- 428 [\[r2046\] N. Freed & N. Borenstein, Multipurpose Internet Mail Extensions \(MIME\) Part Two: Media Types. November 1996.  
429 \(Obsoletes RFC1521, RFC1522, RFC1590\), RFC 2046.](#)
- 430 [\[r2048\] N. Freed, J. Klensin & J. Postel. Multipurpose Internet Mail Extension \(MIME\) Part Four: Registration Procedures.  
431 November 1996. \(Format: TXT=45033 bytes\) \(Obsoletes RFC1521, RFC1522, RFC1590\) \(Also BCP0013\), RFC 2048.](#)
- 432 [r2068] R Fielding, et al, "Hypertext Transfer Protocol – HTTP/1.1" RFC 2068, January 1997
- 433 [r2069] J. Franks, et al, "An Extension to HTTP: Digest Access Authentication" RFC 2069, January 1997
- 434 [r2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119 , March 1997
- 435 [r2184] N. Freed, K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and  
436 Continuations", RFC 2184, August 1997,
- 437 [abnf] D. Crocker et al., "Augmented BNF for Syntax Specifications: ABNF", draft-ietf-drums-abnf-04.txt.
- 438 [\[char\] N. Freed, J. Postel: IANA CharSet Registration Procedures, Work in Progress \(draft-freed-charset-reg-02.txt\).](#)
- 439 [dpa] ISO/IEC 10175 Document Printing Application (DPA), June 1996.
- 440 [\[iana\] IANA Registry of Coded Character Sets: ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets](#)
- 441 [ipp-r] Wright, F. D., "Requirements for an Internet Printing Protocol:"
- 442 [ipp-m] Isaacson, S, deBry, R, Hasting, T, Herriot, R, Powell, P, "Internet Printing Protocol/1.0: Model and Semantics"
- 443 [ssl] Netscape, The SSL Protocol, Version 3, (Text version 3.02) November 1996.

## 444 7. Author's Address

445

Robert Herriot (editor)  
Sun Microsystems Inc.  
901 San Antonio.Road, MPK-17  
Palo Alto, CA 94303

Phone: 650-786-8995  
Fax: 650-786-7077  
Email: robert.herriot@eng.sun.com

Sylvan Butler  
Hewlett-Packard

Paul Moore  
Microsoft  
One Microsoft Way  
Redmond, WA 98053

Phone: 425-936-0908  
Fax: 425-93MS-FAX  
Email: paulmo@microsoft.com

Randy Turner  
Sharp Laboratories

11311 Chinden Blvd.  
Boise, ID 83714

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

5750 NW Pacific Rim Blvd  
Camas, WA 98607

Phone: 360-817-8456  
Fax: : 360-817-8436  
Email: rturner@sharplabs.com

IPP Mailing List: [ipp@pwg.org](mailto:ipp@pwg.org)  
IPP Mailing List Subscription: [ipp-request@pwg.org](mailto:ipp-request@pwg.org)  
IPP Web Page: <http://www.pwg.org/ipp/>

446

## 447 8. Other Participants:

Chuck Adams - Tektronix  
Ron Bergman - Data Products  
Keith Carter - IBM  
Angelo Caruso - Xerox  
Jeff Copeland - QMS  
Roger Debry - IBM  
Lee Farrell - Canon  
Sue Gleeson - Digital  
Charles Gordon - Osicom  
Brian Grimshaw - Apple  
Jerry Hadsell - IBM  
Richard Hart - Digital  
Tom Hastings - Xerox  
Stephen Holmstead  
Zhi-Hong Huang - Zenographics  
Scott Isaacson - Novell  
Rich Lomicka - Digital  
David Kellerman - Northlake Software  
Robert Kline - TrueSpectra  
Dave Kuntz - Hewlett-Packard  
Takami Kurono - Brother  
Rich Landau - Digital  
Greg LeClair - Epson

Harry Lewis - IBM  
Tony Liao - Vivid Image  
David Manchala - Xerox  
Carl-Uno Manros - Xerox  
Jay Martin - Underscore  
Larry Masinter - Xerox  
Bob Pentecost - Hewlett-Packard  
Patrick Powell - SDSU  
Jeff Rackowitz - Intermec  
Xavier Riley - Xerox  
Gary Roberts - Ricoh  
Stuart Rowley - Kyocera  
Richard Schneider - Epson  
Shigern Ueda - Canon  
Bob Von Anandel - Allegro Software  
William Wagner - Digital Products  
Jasper Wong - Xionics  
Don Wright - Lexmark  
Rick Yardumian - Xerox  
Lloyd Young - Lexmark  
Peter Zehler - Xerox  
Frank Zhao - Panasonic  
Steve Zilles - Adobe

## 448 9. Appendix A: Protocol Examples

### 449 9.1 Print-Job Request

450 The following is an example of a Print-Job request with job-name, copies, and sides specified.

Octets	Symbolic Value	Protocol field
0x0100	1.0	version
0x0002	PrintJob	operation
0x01	start operation-attributes	operation-attributes tag
<u>0x47</u>	<u>charset type</u>	<u>value-tag</u>

Octets	Symbolic Value	Protocol field
<a href="#">0x0012</a>		<a href="#">name-length</a>
<a href="#">attributes-charset</a>	<a href="#">attributes-charset</a>	<a href="#">name</a>
<a href="#">0x0008</a>		<a href="#">value-length</a>
<a href="#">US-ASCII</a>	<a href="#">US-ASCII</a>	<a href="#">value</a>
<a href="#">0x48</a>	<a href="#">natural-language type</a>	<a href="#">value-tag</a>
<a href="#">0x001B</a>		<a href="#">name-length</a>
<a href="#">attributes-natural-language</a>	<a href="#">attributes-natural-language</a>	<a href="#">name</a>
<a href="#">0x0005</a>		<a href="#">value-length</a>
<a href="#">en-US</a>	<a href="#">en-US</a>	<a href="#">value</a>
0x42	name type	value-tag
0x0008		name-length
job-name	job-name	name
0x0006		value-length
foobar	foobar	value
0x02	start job-attributes	job-attributes tag
0x21	integer type	value-tag
0x0005		name-length
copies	copies	name
0x0004		value-length
0x00000014	20	value
0x44	keyword type	value-tag
0x0005		name-length
sides	sides	name
0x0013		value-length
two-sided-long-edge	two-sided-long-edge	value
0x03	start-data	data-tag
% !PS...	<PostScript>	data

## 451 9.2 Print-Job Response (successful)

452 Here is an example of a Print-Job response which is successful:

Octets	Symbolic Value	Protocol field
0x0100	1.0	version
0x0000	OK (successful)	status-code
0x01	start operation-attributes	operation-attributes tag
<a href="#">0x47</a>	<a href="#">charset type</a>	<a href="#">value-tag</a>
<a href="#">0x0012</a>		<a href="#">name-length</a>
<a href="#">attributes-charset</a>	<a href="#">attributes-charset</a>	<a href="#">name</a>
<a href="#">0x0008</a>		<a href="#">value-length</a>
<a href="#">US-ASCII</a>	<a href="#">US-ASCII</a>	<a href="#">value</a>
<a href="#">0x48</a>	<a href="#">natural-language type</a>	<a href="#">value-tag</a>
<a href="#">0x001B</a>		<a href="#">name-length</a>
<a href="#">attributes-natural-language</a>	<a href="#">attributes-natural-language</a>	<a href="#">name</a>
<a href="#">0x0005</a>		<a href="#">value-length</a>
<a href="#">en-US</a>	<a href="#">en-US</a>	<a href="#">value</a>
0x41	text type	value-tag
0x000E		name-length
status-message	status-message	name
0x0002		value-length

Octets	Symbolic Value	Protocol field
OK	OK	value
0x02	start job-attributes	job-attributes tag
0x21	integer	value-tag
0x0007		name-length
job-id	job-id	name
0x0004		value-length
147	147	value
0x45	uri type	value-tag
0x0008		name-length
job-uri	job-uri	name
0x000E		value-length
http://foo/123	http://foo/123	value
0x25	name type	value-tag
0x0008		name-length
job-state	job-state	name
0x0001		value-length
0x03	pending	value
0x03	start-data	data-tag

### 453 9.3 Print-Job Response (failure)

454 Here is an example of a Print-Job response which fails because the printer does not support sides and because the value 20 for  
455 copies is not supported:

Octets	Symbolic Value	Protocol field
0x0100	1.0	version
0x0400	client-error-bad-request	status-code
0x01	start operation-attributes	operation-attribute tag
<u>0x47</u>	<u>charset type</u>	<u>value-tag</u>
<u>0x0012</u>		<u>name-length</u>
<u>attributes-charset</u>	<u>attributes-charset</u>	<u>name</u>
<u>0x0008</u>		<u>value-length</u>
<u>US-ASCII</u>	<u>US-ASCII</u>	<u>value</u>
<u>0x48</u>	<u>natural-language type</u>	<u>value-tag</u>
<u>0x001B</u>		<u>name-length</u>
<u>attributes-natural-language</u>	<u>attributes-natural-language</u>	<u>name</u>
<u>0x0005</u>		<u>value-length</u>
<u>en-US</u>	<u>en-US</u>	<u>value</u>
0x41	text type	value-tag
0x000E		name-length
status-message	status-message	name
0x000D		value-length
bad-request	bad-request	value
0x04	start unsupported-job-attributes	unsupported-job-attributes tag
0x21	integer type	value-tag
0x0005		name-length
copies	copies	name
0x0004		value-length
0x00000014	20	value
0x10	unsupported (type)	value-tag

Octets	Symbolic Value	Protocol field
0x0005		name-length
sides	sides	name
0x0000		value-length
0x03	start-data	data-tag

#### 456 9.4 Print-URI Request

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

Octets	Symbolic Value	Protocol field
0x0100	1.0	version
0x0003	Print-URI	operation
0x01	start operation-attributes	operation-attributes tag
<u>0x47</u>	<u>charset type</u>	<u>value-tag</u>
<u>0x0012</u>		<u>name-length</u>
<u>attributes-charset</u>	<u>attributes-charset</u>	<u>name</u>
<u>0x0008</u>		<u>value-length</u>
<u>US-ASCII</u>	<u>US-ASCII</u>	<u>value</u>
<u>0x48</u>	<u>natural-language type</u>	<u>value-tag</u>
<u>0x001B</u>		<u>name-length</u>
<u>attributes-natural-language</u>	<u>attributes-natural-language</u>	<u>name</u>
<u>0x0005</u>		<u>value-length</u>
<u>en-US</u>	<u>en-US</u>	<u>value</u>
0x45	uri type	value-tag
0x000A		name-length
document-uri	document-uri	name
0x11		value-length
ftp://foo.com/foo	ftp://foo.com/foo	value
0x42	name type	value-tag
0x0008		name-length
job-name	job-name	name
0x0006		value-length
foobar	foobar	value
0x02	start job-attributes	job-attributes tag
0x21	integer type	value-tag
0x0005		name-length
copies	copies	name
0x0004		value-length
0x00000001	1	value
0x03	start-data	data-tag
%!PS...	<PostScript>	data

#### 458 9.5 Create-Job Request

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

Octets	Symbolic Value	Protocol field
0x0100	1.0	version
0x0005	Create-Job	operation
0x03	start-data	data-tag

460 **9.6 Get-Jobs Request**

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

<b>Octets</b>	<b>Symbolic Value</b>	<b>Protocol field</b>
0x0100	1.0	version
0x000A	Get-Jobs	operation
0x01	start operation-attributes	operation-attributes-tag
<a href="#">0x47</a>	<a href="#">charset type</a>	<a href="#">value-tag</a>
<a href="#">0x0012</a>		<a href="#">name-length</a>
<a href="#">attributes-charset</a>	<a href="#">attributes-charset</a>	<a href="#">name</a>
<a href="#">0x0008</a>		<a href="#">value-length</a>
<a href="#">US-ASCII</a>	<a href="#">US-ASCII</a>	<a href="#">value</a>
<a href="#">0x48</a>	<a href="#">natural-language type</a>	<a href="#">value-tag</a>
<a href="#">0x001B</a>		<a href="#">name-length</a>
<a href="#">attributes-natural-language</a>	<a href="#">attributes-natural-language</a>	<a href="#">name</a>
<a href="#">0x0005</a>		<a href="#">value-length</a>
<a href="#">en-US</a>	<a href="#">en-US</a>	<a href="#">value</a>
0x21	integer type	value-tag
0x0005		name-length
limit	limit	name
0x0004		value-length
0x00000032	50	value
0x44	keyword type	value-tag
0x0014		name-length
requested-attributes	requested-attributes	name
0x0006		value-length
job-id	job-id	value
0x44	keyword type	value-tag
0x0000	additional value	name-length
0x0008		value-length
job-name	job-name	value
0x03	start-data	data-tag

462 **9.7 Get-Jobs Response**463 The following is an of Get-Jobs response from previous request with 3 jobs. The Printer returns no information about the second  
464 job.

<b>Octets</b>	<b>Symbolic Value</b>	<b>Protocol field</b>
0x0100	1.0	version
0x0000	OK (successful)	status-code
0x01	start operation-attributes	operation-attribute-tag
<a href="#">0x47</a>	<a href="#">charset type</a>	<a href="#">value-tag</a>
<a href="#">0x0012</a>		<a href="#">name-length</a>
<a href="#">attributes-charset</a>	<a href="#">attributes-charset</a>	<a href="#">name</a>
<a href="#">0x0008</a>		<a href="#">value-length</a>
<a href="#">ISO-8859-1</a>	<a href="#">ISO-8859-1</a>	<a href="#">value</a>
<a href="#">0x48</a>	<a href="#">natural-language type</a>	<a href="#">value-tag</a>
<a href="#">0x001B</a>		<a href="#">name-length</a>
<a href="#">attributes-natural-language</a>	<a href="#">attributes-natural-language</a>	<a href="#">name</a>
<a href="#">0x0005</a>		<a href="#">value-length</a>
<a href="#">en-US</a>	<a href="#">en-US</a>	<a href="#">value</a>

Octets	Symbolic Value	Protocol field
0x41	text type	value-tag
0x000E		name-length
status-message	status-message	name
0x0002		value-length
OK	OK	value
0x02	start job-attributes (1st object)	job-attributes-tag
<u>0x48</u>	<u>natural-language type</u>	<u>value-tag</u>
<u>0x001B</u>	<u>attributes-natural-language</u>	<u>name-length</u>
<u>attributes-natural-language</u>	<u>attributes-natural-language</u>	<u>name</u>
<u>0x0005</u>		<u>value-length</u>
<u>fr-CA</u>	<u>fr-CA</u>	<u>value</u>
0x21	integer type	value-tag
0x0006		name-length
job-id	job-id	name
0x0004		value-length
147	147	value
0x42	name type	value-tag
0x0008		name-length
job-name	job-name	name
0x0003		name-length
<u>fo<del>ue</del></u>	<u>fo<del>ue</del></u>	name
0x02	start job-attributes (2nd object)	job-attributes-tag
0x02	start job-attributes (3rd object)	job-attributes-tag
0x21	interger type	value-tag
0x0006		name-length
job-id	job-id	name
0x0004		value-length
148	148	value
<u>0x42</u>	<u>name type</u>	<u>value-tag</u>
<u>0x13</u>	<u>compoundValue</u>	<u>value-tag</u>
0x0008		name-length
job-name	job-name	name
<u>0x0004</u>		<u>value-length</u>
<u>0x0002</u>	<u>2</u>	<u>value (number of values)</u>
<u>0x48</u>	<u>naturalLanguage</u>	<u>value-tag</u>
<u>0x0000</u>	<u>muli-value marker</u>	<u>name-length</u>
<u>0x0005</u>		<u>value-length</u>
<u>de-CH</u>	<u>de-CH</u>	<u>value</u>
<u>0x42</u>	<u>name type</u>	<u>value-tag</u>
<u>0x0000</u>	<u>muli-value marker</u>	<u>name-length</u>
0x0003		name-length
<u>b<del>ä</del>ar</u>	bar	name
0x03	start-data	data-tag

## 465 10. Appendix B: Mapping of Each Operation in the Encoding

466 The next three tables show the results of applying the rules above to the operations defined in the IPP model document. There is  
467 no information in these tables that cannot be derived from the rules presented in Section 3.8 "Mapping of Attribute Names".

468 The following table shows the mapping of all IPP model-document request attributes to an appropriate xxx-attribute-sequence or  
469 special position in the protocol.

470 The table below shows the attributes for operations sent to a Printer URI.

Operation	operation attributes	job attributes	special position
Print-Job	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> job-name <a href="#">document-name</a> ipp-attribute-fidelity <a href="#">document-charset</a> <a href="#">document-natural-language</a>	job-template attributes	document-content
Create-Job or Validate-Job	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> job-name ipp-attribute-fidelity	job-template attributes	
Print-URI	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> job-name ipp-attribute-fidelity document-uri <a href="#">document-charset</a> <a href="#">document-natural-language</a>	job-template attributes	
Send-Document	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> job-id last-document document-name <a href="#">document-charset</a> <a href="#">document-natural-language</a>		document-content
Send-URI	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> job-id last-document document-name document-uri <a href="#">document-charset</a> <a href="#">document-natural-language</a>		
Cancel-Job	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> job-id message		
Get-Attributes (for a Printer)	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> requested-attributes document-format		
Get-Attributes (for a Job)	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> job-id requested-attributes		
Get-Jobs	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> limit requested-attributes which-jobs		



471 The table below shows the attributes for operations sent to a Job URI.

Operation	operation attributes	job attributes	special position
Send-Document	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> last-document document-name		document-content
Send-URI	<a href="#">document-charset</a> <a href="#">document-natural-language</a> <a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> last-document document-name document-uri		
Cancel-Job	<a href="#">document-charset</a> <a href="#">document-natural-language</a> <a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> message		
Get-Attributes (for a Job)	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> requested-attributes		

472 The following two tables shows the mapping of all IPP model-document response attributes to an appropriate xxx-attribute-  
473 sequence or special position in the protocol.

Operation	operation attributes	job-attributes	unsupported-job-attributes	special position
Print-Job, Print-URI, Create-Job, Send- Document or Send-URI	<a href="#">attributes-charset</a> <a href="#">attributes-natural- language</a> status-message	job-id job-uriURI _job-name job-state job-state-reasons job-state-message number-of- intervening-jobs	unsupported attributes	status-code
Validate-Job	<a href="#">attributes-charset</a> <a href="#">attributes-natural- language</a> status-message		unsupported attributes	status-code

474 Note: the unsupported-job-attributes are present only if the client included some job attributes that the Printer doesn't support.

475 Note: the job-attributes are present only if the server returns the status code of successful-ok or successful-ok-ignored-or-  
476 substituted-attributes.

Operation	operation attributes	job-attributes	printer-attributes	special position
Cancel-Job	<a href="#">attributes-charset</a> <a href="#">attributes-natural- language</a>			status-code

Operation	operation attributes	job-attributes	printer-attributes	special position
	status-message			
Get-Attributes (of a job)	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> status-message	requested attributes		status-code
Get-Attributes (of a printer)	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> status-message		requested attributes	status-code
Get-Jobs	<a href="#">attributes-charset</a> <a href="#">attributes-natural-language</a> status-message	requested attributes (see the Note below)		status-code

477 Note for Get-Jobs: there is a separate job-attribute-sequence containing requested-attributes for each job object in the response

## 478 11. Appendix C: Hints to implementors using IPP with SSL3

479 WARNING: Clients and IPP objects using intermediate secure connection protocol solutions such as IPP in combination with  
480 Secure Socket Layer Version 3 (SSL3), which are developed in advance of IPP and TLS standardization, might not be  
481 interoperable with IPP and TLS standards-conforming clients and IPP objects.

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

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

486 connect (URI), returns a status.

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

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

492 disconnect (session)

493 to disconnect a particular session.

494 The session-id available from the “connect” could be used.

495 resume (session)

496 to reconnect using a previous session-id.

497 The availability of this information as administration features are left for implementors, and need not be standardized at this time