

Robert Herriot (editor)
Sun Microsystems
Sylvan Butler
Hewlett-Packard
Paul Moore
Microsoft.
Randy Turner
Sharp Labs
September 04, 1997

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

Status of this Memo

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

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

To learn the current status of any Internet-Draft, please check the "Iid-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), 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. 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: Security
- Internet Printing Protocol/1.0: Protocol Specification
- Internet Printing Protocol/1.0: Directory Schema

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 security document covers potential threats and proposed counters to those threats. 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 server side components. Finally, the directory schema document shows a generic schema for directory service entries that represent instances of IPP Printers.

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

46 Table of Contents

47	1. Introduction	3
48	2. Conformance Terminology.....	3
49	3. Encoding of the Operation Layer.....	3
50	3.1 Picture of the Encoding.....	3
51	3.2 Syntax of Encoding.....	5
52	3.3 Version	6
53	3.4 Mapping of Operations.....	6
54	3.5 Mapping of Status-code.....	7
55	3.6 Tags	8
56	3.6.1 Delimiter Tags.....	8
57	3.6.2 Value Tags.....	8
58	3.7 Name-Lengths.....	9
59	3.8 Mapping of Attribute Names.....	10
60	3.9 Value Lengths.....	11
61	3.10 Mapping of Attribute Values.....	11
62	3.11 Data	12
63	4. Encoding of Transport Layer.....	12
64	4.1 General Headers.....	12
65	4.2 Request Headers.....	13
66	4.3 Response Headers.....	14
67	4.4 Entity Headers.....	14
68	5. Security Considerations.....	14
69	6. References.....	15
70	7. Author's Address.....	16
71	8. Other Participants:	16
72	9. Appendix A: Protocol Examples.....	17
73	9.1 Print-Job Request	17
74	9.2 Print-Job Response (successful).....	17
75	9.3 Print-Job Response (failure)	18
76	9.4 Print-URI Request.....	18
77	9.5 Create-Job Request.....	19
78	9.6 Get-Jobs Request.....	19
79	9.7 Get-Jobs Response.....	19
80	10. Appendix B: Mapping of Each Operation in the Encoding	20
81		
82		
83		

84 1. Introduction

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

87 The transport layer consists of an HTTP/1.1 request or response. RFC 2068 [27] describes HTTP/1.1. This document specifies
88 the HTTP headers that an IPP implementation supports.

89 The operation layer consists of a message body in an HTTP request or response. The document "Internet Printing
90 Protocol/1.0: Model and Semantics" [21] defines the semantics of such a message body and the supported values. This
91 document specifies the encoding of an IPP operation. The aforementioned document [21] is henceforth referred to as the "IPP
92 model document"

93 2. Conformance Terminology

94 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT",
95 "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [25].

96 3. Encoding of the Operation Layer

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

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

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

- 107 • informally through pictures and description
- 108 • formally through Augmented Backus-Naur Form (ABNF), as specified by draft-ietf-drums-abnf-02.txt [29]

109 3.1 Picture of the Encoding

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

111	-----		
112		version	2 bytes - required
113	-----		
114		operation (request) or status-code (response)	2 bytes - required
115	-----		
116		parameter-tag	1 byte
117	-----		optional
118		parameter-sequence	m bytes
119	-----		
120		attribute-tag	1 byte
121	-----		-0 or more
122		attribute-sequence	n bytes
123	-----		
124		data-tag	1 byte - required
125	-----		
126		data	q bytes - optional
127	-----		

128 ~~The parameter-tag and parameter-sequence may be omitted if the operation has no parameters.~~ The attribute-tag and attribute-
 129 sequence may be omitted if the operation has no attributes or it may be replicated in special cases, such as for an operation that
 130 contains attributes for multiple objects. The data is omitted from some operations, but the data-tag is present even when the
 131 data is omitted. Note, the ~~parameter-tag~~, attribute-tag and data-tag are called 'delimiter-tags'.

132 ~~A parameter-sequence consists of a sequence of zero or more compound-parameters.~~

133	-----		
134		compound-parameter	r bytes - 0 or more
135	-----		

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

137	-----		
138		compound-attribute	s bytes - 0 or more
139	-----		

140 ~~A compound-parameter consists of a parameter with a single value optionally followed by zero or more additional values.~~ A
 141 compound-attribute consists an attribute with a single value followed by zero or more additional values.

142 Each ~~parameter or~~ attribute consists of:

143	-----		
144		value-tag	1 byte
145	-----		
146		name-length (value is u)	2 bytes
147	-----		
148		name	u bytes
149	-----		
150		value-length (value is v)	2 bytes
151	-----		
152		value	v bytes
153	-----		

154 An additional value consists of:

155	-----				
156		value-tag	1 byte	-0 or more	
157	-----				
158		name-length (value is 0x0000)	2 bytes		
159	-----				
160		value-length (value is w)	2 bytes		
161	-----				
162		value	w bytes		
163	-----				
164					

165 Note: an additional value is like an ~~parameter or~~ attribute whose name-length is 0.

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

167	-----			
168		version	2 bytes	- required
169	-----			
170		operation (request) or status-code (response)	2 bytes	- required
171	-----			
172		tag (delimiter-tag or value-tag)	1 byte	-0 or more
173	-----			
174		empty or rest of attribute	x bytes	
175	-----			
176		data-tag	2 bytes	- required
177	-----			
178		data	y bytes	- optional
179	-----			
180				

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

- 182 ~~• parameters~~
- 183 • attributes
- 184 • data
- 185 • the remainder of a single ~~parameter or~~ attribute where the tag specifies the type of the value.

186 3.2 Syntax of Encoding

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

```

190 ipp-message = ipp-request / ipp-response
191 ipp-request = version operation
192 [parameter-tag parameter-sequence]
193 *(attribute-tag attribute-sequence) data-tag data
194 ipp-response = version status-code
195 [parameter-tag parameter-sequence]
196 *(attribute-tag attribute-sequence) data-tag data
197
198 version = major-version minor-version
199 major-version = SIGNED-BYTE ; initially %d1
200 minor-version = SIGNED-BYTE ; initially %d0

```

201
 202 operation = SIGNED-SHORT ; mapping from model defined below
 203 status-code = SIGNED-SHORT ; mapping from model defined below
 204
 205 ~~parameter-sequence = *compound-parameter~~
 206 attribute-sequence = *compound-attribute
 207 ~~compound-parameter = parameter *additional-values~~
 208 compound-attribute = attribute *additional-values
 209
 210 ~~parameter = value-tag name-length name value-length value~~
 211 attribute = value-tag name-length name value-length value
 212 additional-values = value-tag zero-name-length value-length value
 213
 214 name-length = SIGNED-SHORT ; number of octets of 'name'
 215 name = LALPHA *(LALPHA / DIGIT / "-" / "_" / ":")
 216 value-length = SIGNED-SHORT ; number of octets of 'value'
 217 value = OCTET-STRING
 218
 219 data = OCTET-STRING
 220
 221 zero-name-length = %x00.00 ; name-length of 0
 222 ~~parameter-tag = %x01 ; tag of 1~~
 223 attribute-tag = %x02 ; tag of 2
 224 data-tag = %x03 ; tag of 3
 225 value-tag = %x10-FF
 226
 227 SIGNED-BYTE = BYTE
 228 SIGNED-SHORT = 2BYTE
 229 DIGIT = %x30-39 ; "0" to "9"
 230 LALPHA = %x61-7A ; "a" to "z"
 231 BYTE = %x00-FF
 232 OCTET-STRING = *BYTE
 233

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

239 3.3 Version

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

243 3.4 Mapping of Operations

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

245 ~~Note: the values 0x4000 to 0xFFFF are reserved for private extensions. The following SHALL be the mapping of operations~~
 246 ~~names to integer values which are encoded as a SIGNED SHORT. The operations are defined in the IPP model document. The~~

247 ~~table below includes a range of values for future extensions to the protocol and a separate range for private extensions. It is~~
 248 ~~RECOMMENDED that the private extension values be used for temporary experimental implementations and not for deployed~~
 249 ~~products.~~

Encoding (hex)	Operation
0x0	reserved (not used)
0x1	Get Operations
0x2	Print Job
0x3	Print URI
0x4	Validate Job
0x5	Create Job
0x6	Send Document
0x7	Send URI
0x8	Cancel Job
0x9	Get Attributes
0xA	Get Jobs
0xB-0x3FFF	reserved for future operations
0x4000-0xFFFF	reserved for private extensions

250 3.5 Mapping of Status-code

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

253 ~~The following SHALL be the mapping of status code names to integer values which are encoded as a SIGNED-SHORT. The~~
 254 ~~status code names are defined in the IPP model document.~~

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

257 ~~Note: the integer encodings below were chosen to be similar to corresponding Status Code values in HTTP. The IPP client and~~
 258 ~~server errors have the same relative offset to their base as corresponding HTTP errors, but the IPP base is a multiple of 0x100~~
 259 ~~whereas the HTTP base is a multiple of 100. Despite this similarity, the Status Code returned at the HTTP level will always be~~
 260 ~~different except in the case where 'OK' is returned at both levels, 200 (OK) in HTTP and 0 (successful OK) in IPP.~~

261 ~~Note: some status code values, such as client error unauthorized, may be returned at the transport (HTTP) level rather than the~~
 262 ~~operation level.~~

263 ~~ISSUE: as implementations proceed, we will learn what error code need to be supported at the IPP level.~~

Encoding (hex)	Status-Code Name
0	successful OK
0x400	client error bad request
0x401	client error unauthorized
0x403	client error forbidden
0x404	client error not found
0x405	client error method not allowed
0x408	client error timeout
0x40A	client error gone
0x40D	client error request entity too large
0x40E	client error request URI too long

Encoding (hex)	Status-Code Name
0x40F	client-error-unsupported-document-format
0x410	client-error-attribute-not-supported
0x500	server-error-internal-server-error
0x501	server-error-operation-not-implemented
0x503	server-error-service-unavailable
0x504	server-error-timeout
0x505	server-error-version-not-supported
0x506	server-error-printer-error
0x507	server-error-temporary-error

264 3.6 Tags

265 There are two kinds of tags:

- 266 • delimiter tags: delimit major sections of the protocol, namely ~~parameters~~, attributes and data
- 267 • value tags: specify the type of each ~~parameter or~~ attribute value

268 3.6.1 Delimiter Tags

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

Tag Value (Hex)	Delimiter
0x00	reserved
0x01	reserved parameter-tag
0x02	attribute-tag
0x03	data-tag
0x04-0x0F	reserved for future delimiters

270 3.6.2 Value Tags

271 The remaining tables show values for the value-tag, which is the first octet of ~~an~~ ~~parameter or~~ attribute. The value-tag specifies
 272 the type of the value of the ~~parameter or~~ attribute. The value of the value-tag of ~~an~~ ~~parameter or~~ attribute SHALL either be a
 273 type value specified in the model document or an “out-of-band” value, such as “unsupported” or “default”. If the value of
 274 value-tag for ~~an~~ attribute ~~or parameter~~ is not “out-of-band” and differs from the value type specified by the model document,
 275 then a server receiving such a request MAY reject it, and a client receiving such a response MAY ignore the attribute ~~or~~
 276 ~~parameter~~.

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

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

278 The “unsupported” value SHALL be used in the attribute-sequence of an error response for those attributes which the server
 279 does not support. The “default” value is reserved for future use of setting value back to their default value. The “no-value” value

280 is 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
281 specified value for one or more of the documents.

282 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

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

284 The following table specifies the octet-string values for the value-tag

Tag Value (Hex)	Meaning
0x30	reserved
0x31	dateTime
0x32	resolution
0x33-0x3F	reserved for future octet-string types

285 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
0x47-0x5F	reserved for future character string types

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

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

289 ~~Issue: should this be a type 1 process.~~

290 3.7 Name-Lengths

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

293 If a name-length field has a value of zero, the following name field SHALL be empty, and the following value SHALL be
294 treated as an additional value for the preceding attributeparameter. ~~Within a parameter sequence, if two parameters have the~~

295 ~~same name, the first occurrence SHALL be ignored.~~ Within an attribute-sequence, if two attributes have the same name, the
 296 first occurrence SHALL be ignored. The zero-length name is the only mechanism for multi-valued ~~parameters and~~ attributes.

297 3.8 Mapping of Attribute Parameter Names

298 Some ~~attributes~~parameters are encoded in a special position. These ~~attribute~~parameters are:

- 299 • “printer-uri”: The printer-uri of each ~~printer object~~operation in the IPP model document SHALL be specified ~~both as~~
 300 ~~a parameter named “printer-uri” in the operation layer, and~~ outside of the operation layer as the request-URI on the
 301 Request-Line at the HTTP level,.
- 302 ~~• “job-uri”: The job-uri of each job object operation in the IPP model document SHALL be specified both as a~~
 303 ~~parameter named “job-uri” in the operation layer, and outside of the operation layer as the request URI on the~~
 304 ~~Request Line at the HTTP level,.~~
- 305 ~~•~~
- 306 • “document-content”: The ~~attribute~~parameter named “document-content” in the IPP model document SHALL become
 307 the “data” in the operation layer.
- 308 • “status-code”: The ~~attribute~~parameter named “status-code” in the IPP model document SHALL become the “status-
 309 code” field in the operation layer response.

310 The remaining attributes defined in the model document represent either a specific attribute or a collection of attributes. These
 311 attributes SHALL be encoded in the attribute-sequence according to the order specified in the model document. If an attribute
 312 in the model document represents a collection of object attributes, those attributes are encoded as a sequence of attributes in the
 313 position specified by the model document, but the attributes in this collection may be in any order.

314 ~~The remaining parameters are encoded in the parameter-sequence or the attribute-sequence. The parameter-sequence is for~~
 315 ~~actual operation parameters and the attribute-sequence is for object attributes. Of the parameters defined in the IPP model~~
 316 ~~document, some represent an actual operation parameters and others represent a collection of object attributes.~~

317 ~~A parameter in the IPP model document SHALL represent a collection of object attributes if its name contains the word~~
 318 ~~“attributes” immediately preceded by a space; otherwise it SHALL represent an actual operation parameter. Note, some actual~~
 319 ~~operation parameters contain the word “attributes” preceded by a hyphen (“-”). They are not a collection of attributes.~~ISSUE:
 320 coordinate this with the model document.

321 ~~If a parameter in IPP model document represents an actual operation parameter and is not in a special position, it SHALL be~~
 322 ~~encoded in the parameter-sequence using the text name of the parameter specified in the IPP model document.~~

323 ~~If a parameter in IPP model document represents a collection of object attributes, the attributes SHALL be encoded in the~~
 324 ~~attribute-sequence using the text names of the attributes specified in the IPP model document. The IPP model document~~
 325 ~~specifies the members of such attribute collections, but does not require that all members of a collection be present in an~~
 326 ~~operation.~~

327 If a response contains unsupported attributes, these attributes SHALL be in a second attribute-sequence. Otherwise, if
 328 an operation contain attributes from exactly one object, there SHALL be exactly one attribute-sequence. If an operation contains
 329 attributes from more than one object (e.g. Get-Jobs response), the attributes from each object SHALL be in a separate attribute-
 330 sequence, such that the attributes from the ith object are in the ith attribute-sequence.

331 See Section 10 “Appendix B: Mapping of Each Operation in the Encoding” for table showing the application of the rules
 332 above.

333 3.9 Value Lengths

334 Each **attributeparameter** value SHALL be preceded by a SIGNED-SHORT which SHALL specify the number of octets in the
335 value which follows this length, exclusive of the two bytes specifying the length.

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

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

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

342 3.10 Mapping of Attribute ~~and Parameter~~ Values

343 The following SHALL be the mapping of attribute ~~and parameter~~ values to their IPP encoding in the value field. The syntax
344 types are defined in the IPP model document.

Syntax of Attribute Value	Encoding
text	an octet string where each character is a member of the UCS-2 coded character set and is encoded using UTF-8. 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.
name	same as text
language	same as text but with a syntax specified by RFC 1766
keyword	same as text. Allowed text values are defined in the IPP model document
uri	same as text
uriScheme	same as text
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
dateTime	eleven octets whose contents are defined by “DateAndTime” in RFC 1903. 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.
resolution	nine octets consisting of 2 SIGNED-INTEGERS followed by a SIGNED-BYTE. The values are the same as those specified in draft-ietf-printmib-mib-info-02.txt [30]. 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.
1setOf X	encoding according to the rules for an attributeparameter with more than more value. Each value X is encoded according to the rules for encoding its type.
rangeOf X	same 1setOf X where the number of values is 2.

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

346 **3.11 Data**

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

348 **4. Encoding of Transport Layer**

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

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

- 351
- 352 • the URI of the target job or printer operation
 - 353 • 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.
 - 354 • the client's language, the character-set and the transport encoding.

355 Each HTTP operation shall use the POST method where the request-URI is the object target of the operation, and where the
356 "Content-Type" of the message-body in each request and response shall be "application/ipp". The message-body shall contain
357 the operation layer and shall have the syntax described in section 3.2 "Syntax of Encoding". A client implementation SHALL
358 adhere to the rules for a client described in RFC 2068. A server implementation SHALL adhere the rules for an origin server
359 described in RFC 2068. In the following sections, there are a tables of all HTTP headers which describe their use in an IPP
360 client or server. The following is an explanation of each column in these tables.

- 361
- 362 • the "header" column contains the name of a header
 - 363 • the "request/client" column indicates whether a client sends the header.
 - 364 • the "request/server" column indicates whether a server supports the header when received.
 - 365 • the "response/server" column indicates whether a server sends the header.
 - 366 • the "response /client" column indicates whether a client supports the header when received.
 - 367 • the "values and conditions" column specifies the allowed header values and the conditions for the header to be present in a request/response.

368 The table for "request headers" does not have columns for responses, and the table for "response headers" does not have
369 columns for requests.

370 The following is an explanation of the values in the "request/client" and "response/server" columns.

- 371
- 372 • **must:** the client or server MUST send the header,
 - 373 • **must-if:** the client or server MUST send the header when the condition described in the "values and conditions"
374 column is met,
 - 375 • **may:** the client or server MAY send the header
 - **not:** the client or server SHOULD NOT send the header. It is not relevant to an IPP implementation.

376 The following is an explanation of the values in the "response/client" and "request/server" columns.

- 377
- 378 • **must:** the client or server MUST support the header,
 - 379 • **may:** the client or server MAY support the header
 - **not:** the client or server SHOULD NOT support the header. It is not relevant to an IPP implementation.

380 **4.1 General Headers**

381 The following is a table for the general headers.

382 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 MUST include this header for the last operation in such a sequence.
Date	may	may	must	may	per RFC 1123 [9]
Pragma`	must	not	must	not	“no-cache” only
Transfer-Encoding	must-if	must	must-if	must	“chunked” only . Header MUST be present if Content-Length is absent.
Upgrade	not	not	not	not	
Via	not	not	not	not	

383

384 4.2 Request Headers

385 The following is a table for the request headers.

386

Request-Header	Client	Server	Request Values and Conditions
Accept	may	must	“application/ipp” only. This value is the default if the client omits it
Accept-Charset	may	must	per IANA Character Set registry. ISSUE: is this useful for IPP?
Accept-Encoding	may	must	empty and per RFC 2068 [27] and IANA registry for content-codings
Accept-Language	may	must	see RFC 1766 [26]. A server SHOULD honor language requested by returning the values status-message, job-state-message and printer-state-reason in one of requested languages.
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	

Request-Header	Client	Server	Request Values and Conditions
Referer	not	not	
User-Agent	not	not	

387 4.3 Response Headers

388 The following is a table for the request headers.

389

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.

390 4.4 Entity Headers

391 The following is a table for the entity headers.

392

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	may	must	must	must	see RFC 1766 [26].
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	

393 5. Security Considerations

394 When utilizing HTTP 1.1 as a transport of IPP, the security considerations outlined in RFC 2068 apply. Specifically, IPP
 395 servers can generate a 401 "Unauthorized" response code to request client authentication and IPP clients should correctly
 396 respond with the proper "Authorization" header. Both Basic Authentication (RFC 2068) and Digest Authentication (RFC
 397 2069) flavors of authentication should be supported. The server chooses which type(s) of authentication to accept. Digest
 398 Authentication is a more secure method, and is always preferred to Basic Authentication.

399 For secure communication (privacy in particular), IPP should be run using a secure communications channel. Both Transport
400 Layer Security - TLS (draft-ietf-tls-protocol-03) and IPSec (RFC 1825) provide necessary features for security. It is possible to
401 combine the two techniques, HTTP 1.1 client authentication (either basic or digest) with secure communications channel
402 (either TLS or IPSec). Together the two are more secure than client authentication and they perform user authentication.

403 Complete discussion of IPP security considerations can be found in the IPP Security document

404 ISSUE: how does each security mechanism supply the job-originating-user and job-originating-host values?

405 6. References

- 406 [1] Smith, R., Wright, F., Hastings, T., Zilles, S., and Gyllenskog, J., "Printer MIB", RFC 1759, March 1995.
- 407 [2] Berners-Lee, T, Fielding, R., and Nielsen, H., "Hypertext Transfer Protocol - HTTP/1.0", RFC 1945, August 1995.
- 408 [3] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", RFC 822, August 1982.
- 409 [4] Postel, J., "Instructions to RFC Authors", RFC 1543, October 1993.
- 410 [5] ISO/IEC 10175 Document Printing Application (DPA), June 1996.
- 411 [6] Herriot, R. (editor), X/Open A Printing System Interoperability Specification (PSIS), August 1995.
- 412 [7] Kirk, M. (editor), POSIX System Administration - Part 4: Printing Interfaces, POSIX 1387.4 D8, 1994.
- 413 [8] Borenstein, N., and Freed, N., "MIME (Multi-purpose Internet Mail Extensions) Part One: Mechanism for Specifying
414 and Describing the Format of Internet Message Bodies", RFC 1521, September, 1993.
- 415 [9] Braden, S., "Requirements for Internet Hosts - Application and Support", RFC 1123, October, 1989,
- 416 [10] McLaughlin, L. III, (editor), "Line Printer Daemon Protocol" RFC 1179, August 1990.
- 417 [11] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource Locators (URL)", RFC 1738, December, 1994.
- 418 [20] Wright, F. D., "Requirements for an Internet Printing Protocol:"
- 419 [21] Isaacson, S, deBry, R, Hasting, T, Herriot, R, Powell, P, "Internet Printing Protocol/1.0: Model and Semantics"
- 420 [22] Internet Printing Protocol/1.0: Security
- 421 [23] Herriot, R, Butler, S, Moore, P, Turner, R, "Internet Printing Protocol/1.0: Protocol Specification" (This document)
- 422 [24] Carter, K, Isaacson, S, "Internet Printing Protocol/1.0: Directory Schema"
- 423 [25] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119 , March 1997
- 424 [26] H. Alvestrand, " Tags for the Identification of Languages", RFC 1766, March 1995.
- 425 [27] R Fielding, et al, "Hypertext Transfer Protocol – HTTP/1.1" RFC 2068, January 1997
- 426 [28] J. Case, et al. "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC
427 1903, January 1996.

- 428 [29] D. Crocker et al., "Augmented BNF for Syntax Specifications: ABNF", draft-ietf-drums-abnf-03.txt.
 429 [30] R. Turner, "Printer MIB", draft-ietf-printmib-mib-info-02.txt, July 12, 1997.

430 7. Author's Address

431

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
 11311 Chinden Blvd.
 Boise, ID 83714

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

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

432

433 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

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 Andel - Allegro Software
 William Wagner - Digital Products
 Jasper Wong - Xionics
 Don Wright - Lexmark

Robert Kline - TrueSpectra
 Dave Kuntz - Hewlett-Packard
 Takami Kurono - Brother
 Rich Landau - Digital
 Greg LeClair - Epson

Rick Yardumian - Xerox
 Lloyd Young - Lexmark
 Peter Zehler - Xerox
 Frank Zhao - Panasonic
 Steve Zilles - Adobe

434 9. Appendix A: Protocol Examples

435 9.1 Print-Job Request

436 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
0x02	start attributes	attribute tag
0x42	name type	value-tag
0x0008		name-length
job-name	job-name	name
0x0006		value-length
foobar	foobar	value
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
0x00130+		value-length
two-sided-long-edge	two-sided-long-edge	value
0x03	start-data	data-tag
%!PS...	<PostScript>	data

437 9.2 Print-Job Response (successful)

438 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
0x02+	start attributes parameters	attribute parameter tag
0x41	text type	value-tag
0x000E		name-length
status-message	status-message	name
0x0002		value-length
OK	OK	value
0x2145	integer type	value-tag
0x00078		name-length
job- id uri	job- id uri	name

Octets	Symbolic Value	Protocol field
0x0004E <u>147</u>	<u>147</u>	value-length <u>value</u>
0x25	name type	value-tag
0x0008		name-length
job-state	job-state	name
0x0001		value-length
0x03	pending	value
0x03	start-data	data-tag

439 9.3 Print-Job Response (failure)

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

Octets	Symbolic Value	Protocol field
0x0100	1.0	version
0x0400	client-error-bad-request	status-code
0x024 <u>0x024</u>	start <u>attriburesparameters</u>	<u>attributeparameter</u> tag
0x41	text type	value-tag
0x000E		name-length
status-message	status-message	name
0x000D		value-length
bad-request	bad-request	value
0x02	start attributes	attribute tag
0x21	integer type	value-tag
0x0005		name-length
copies	copies	name
0x0004		value-length
0x00000014	20	value
0x10	unsupported (type)	value-tag
0x0005		name-length
sides	sides	name
0x0000		value-length
0x03	start-data	data-tag

442 9.4 Print-URI Request

443 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
0x024 <u>0x024</u>	start- <u>attriburesparameters</u>	<u>attributeparameter</u> tag
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

Octets	Symbolic Value	Protocol field
job-name	job-name	name
0x0006		value-length
foobar	foobar	value
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

444 9.5 Create-Job Request

445 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

446 9.6 Get-Jobs Request

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

Octets	Symbolic Value	Protocol field
0x0100	1.0	version
0x000A	Get-Jobs	operation
0x021	start- attributes parameters	attribute parameter-tag
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
0x00067		value-length
job- id i	job- id i	value
0x44	keyword type	value-tag
0x0000	additional value	name-length
0x0008		value-length
job-name	job-name	value
0x03	start-data	data-tag

448 9.7 Get-Jobs Response

449 The following is an of Get-Jobs response from previous request with 3 jobs. The Printer returns no information about the
450 second job.

Octets	Symbolic Value	Protocol field
0x0100	1.0	version
0x0000	OK (successful)	status-code

Octets	Symbolic Value	Protocol field
0x021	start- attributes parameters	attribute parameter-tag
0x41	text type	value-tag
0x000E	status-message	name-length
status-message	status-message	name
0x0002		value-length
OK	OK	value
0x02	start-attributes (1st object)	attribute-tag
0x2145	integer uri type	value-tag
0x00067		name-length
job- id uri	job- id uri	name
0x0004E		value-length
147http://foo/123	147http://foo/123	value
0x42	name type	value-tag
0x0008		name-length
job-name	job-name	name
0x0003		name-length
foo	foo	name
0x02	start-attributes (2nd object)	attribute-tag
0x02	start-attributes (3rd object)	attribute-tag
0x2145	interger uri type	value-tag
0x00067		name-length
job- id uri	job- id uri	name
0x0004E		value-length
148http://foo/124	148http://foo/124	value
0x42	name type	value-tag
0x0008		name-length
job-name	job-name	name
0x0003		name-length
bar	bar	name
0x03	start-data	data-tag

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

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

455 The following table shows the mapping of all IPP model document request attributesparameters to an parameter sequence,
 456 attribute-sequence or special position in the protocol. A request contains at most one attribute sequence.

Operation	<u>attribute</u> sequence	special position
Get-Operations	printer-uri	
Print-Job	printer-uri best-effort job-name <u>job-template attributes</u> <u>document attributes</u>	document-content
Print-URI	printer-uri best-effort job-name document-uri	

Operation	<u>attribute</u> sequence	special position
Validate-Job or Create-Job	<u>job-template attributes</u>	
	<u>document attributes</u>	
Send-Document	printer-uri best-effort job-name	document-content
	<u>job-template attributes</u> job- id uri last-document	
Send-URI	<u>document attributes</u> job- id uri last-document	
	document-uri	
Cancel-Job	<u>document attributes</u> job- id uri message	
	printer-uri document-format requested-attributes	
Get-Attributes (for a Printer)	job-uri document-format requested-attributes	
	printer-uri limit requested-attributes	

457 The following table shows the mapping of all IPP model document response attributes~~parameters~~ to a parameter-sequence, 1st
 458 attribute-sequence, other attribute sequences or special position in the protocol.

Operation	<u>1st</u> attribute-sequence	other attribute-sequences	special position
Get-Operations	status-message supported-operations		status-code
Print-Job, Print-URI or Create-Job	status-message job- id uri <u>job-status attributes</u>	job-status attributes	status-code
Send-Document or Send-URI	status-message <u>job-status attributes</u>	job-status attributes	status-code
Validate-Job	status-message		status-code
Cancel-Job	status-message		status-code
Get-Attributes	status-message <u>requested attributes</u>	requested attributes	status-code
Get-Jobs	status-message	requested attributes (see the Note below)	status-code

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

460 The following table shows the mapping of all IPP model document error response ~~attributeparameters~~ to an ~~parameter-~~
 461 ~~sequence-~~1st attribute-sequence, 2nd attribute sequence or special position in the protocol. Those operations omitted don't have
 462 special ~~attributesparameters~~ for an error return.

Operation	<u>1st attribute-sequence</u>	2nd attribute-sequence	special position
Print-Job, Print-URI, Validate-Job, Create-Job, Send-Document or Send- URI	status-message	unsupported attributes	status-code

463