

Internet Printing Protocol/1.0: MIME Encoding

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 Internet-Draft specifies an Internet Printing Protocol (IPP) that is intended to be version 1.0. This protocol is heavily influenced by the semantic operations and attributes defined in ISO/IEC 10175 Document Printing Application (DPA) parts 1 and 3. It also incorporates some of the implementation and interoperability lessons learned from other printing related standards such as POSIX System Administration - Part 4 (POSIX 1378.4) and X/Open A Printing System Interoperability Specification (PSIS).

IPP is defined as a set of abstract data types and operations. The operations are implemented using a simple request and response mechanism built on top of HTTP. The abstract data types are encoded as simple ASCII text strings.

The IPP protocol covers only end user operations on basic print service objects. Authentication is realized by mechanisms outside the scope of the protocol, but the protocol does introduce some access control functionality so that only authorized end users are allowed to submit print jobs to printers whose implementation and site policy support access control. Also, the Cancel Job operation requires some authentication so that jobs can only be canceled by the end user who submitted the job. Extended monitoring and management is possible

46 through other protocols such as the SNMP Printer MIB. In the areas  
 47 where there are no existing standards, some proposed and emerging  
 48 standards are being worked (management, security, etc.). As these  
 49 services become more stable, this document (and hence the protocol)  
 50 can be updated to reflect the integration and relationships with  
 51 these other standards.

## 52 Table of Contents

53	1. Introduction	2
54	2. IPP Operations	3
55	2.1 HTTP Overview	3
56	2.2 IPP Operation Encoding	4
57	2.2.1 HTTP Request-Header Fields	4
58	2.2.1.1 IPP Request-Line	5
59	2.2.2 HTTP Response-Header Fields	5
60	2.2.2.1 IPP Status-Line	5
61	2.3 The Print Job	5
62	2.3.1 Print Job Object Header	6
63	2.3.2 Document Header	6
64	2.3.3 Document-Content Header	6
65	2.3.4 Job Attributes	7
66	2.3.5 Document Attributes	7
67	3. Security Considerations	7
68	4. References	8
69	5. Author's Address	8
70	6. Appendix A: Sample IPP Operations	9
71	6.1 Querying the printer	9
72	6.2 Print Operation - with print data included	9
73	6.3 Print Operation - with no data included	10
74	6.4 Querying the state of the job	10
75	6.5 Canceling a Job	11
76	6.6 Listing jobs on a Printer	11
77		

## 78 1. Introduction

79 The Internet Printing Protocol (IPP) is an application level protocol  
 80 that can be used for distributed printing on the Internet. The  
 81 protocol is heavily influenced by the printing model introduced in  
 82 the Document Printing Application (ISO/IEC 10175 DPA) standard, which  
 83 describes a distributed printing service. DPA identifies the end user  
 84 and administrative roles associated with a distributed printing  
 85 service, and defines the set of operations supported by the service.  
 86 This IPP specification (version 1.0) deals only with the end user  
 87 role. These ideas and concepts, when unified with other Internet  
 88 protocols and services, realize a distributed print service for the  
 89 Internet.

90 This specification uses the verbs: "shall", "should", "may", and  
 91 "need not" to specify conformance requirements as follows:

- 92 - "shall": indicates an action that the subject of the sentence  
93 must implement in order to claim conformance to this specification
- 94 - "may": indicates an action that the subject of the sentence does  
95 not have to implement in order to claim conformance to this  
96 specification, in other words that action is an implementation  
97 option
- 98 - "need not": indicates an action that the subject of the sentence  
99 does not have to implement in order to claim conformance to this  
100 specification. The verb "need not" is used instead of "may not",  
101 since "may not" sounds like a prohibition.
- 102 - "should": indicates an action that is recommended for the subject  
103 of the sentence to implement, but is not required, in order to  
104 claim conformance to this specification.

105  
106  
107

## 2. IPP Operations

108 This section introduces the IPP operations. Since IPP specifies the  
109 use of HTTP as the underlying communication protocol, the mapping of  
110 IPP operations on top of HTTP methods is also shown.

### 111 2.1 HTTP Overview

112 IPP is based on the existing HTTP standard. IPP is a lightweight  
113 application-level protocol designed with the Internet in mind. It is  
114 a generic, stateless, object-oriented protocol which can be used for  
115 any task through extension of its request methods (commands).

116 HTTP allows an open-ended set of methods to be used to indicate the  
117 purpose of a request. It builds on the discipline of reference  
118 provided by the Uniform Resource Location (URL) and message formats  
119 similar to those used by Internet Mail and the Multipurpose Internet  
120 Mail Extensions (MIME).

121 HTTP is based on a request-response paradigm. A requesting program (a  
122 client) establishes a connection with a receiving program (a server)  
123 and sends a request to the server in the form of a request method, a  
124 URL, and protocol version, followed by a MIME-like message containing  
125 request modifiers, client information, and possibly print data. The  
126 server responds with a status line, including its protocol version,  
127 and a success or failure code, followed by a MIME-like message  
128 containing server information, entity meta-information, and possibly  
129 some content.

130 Current practice requires that the connection be established by the  
131 client prior to each request and closed by the server after sending  
132 the response. Both clients and servers shall be capable of handling

133 cases where either party closes the connection prematurely, due to  
134 user action, automated time out, or program failure.

## 135 2.2 IPP Operation Encoding

136 IPP messages consist of requests from client to server and responses  
137 from server to client.

138 IPP MESSAGE = Request | Response

139

140 Requests and responses use the generic message format of RFC 822 for  
141 transferring entities. Both messages may include optional header  
142 fields and an entity body. The entity body is separated from the  
143 headers by a null line (a line with nothing preceding the CRLF).

```
144 Request = Request-line
145           * (General-Header
146              | Request-Header
147              | Entity-Header)
148           CRLF
149           [ Entity-Body ]
```

150

```
151 Response = Status-line
152           * (General-Header
153              | Request-Header
154              | Entity-Header)
155           CRLF
156           [ Entity-Body ]
```

157

158 All IPP headers conform to the syntax

159 IPP-Header = field-name ":" [field-value] CRLF.

160

161 IPP/1.0 defines the octet sequence CRLF as the end-of-line marker for  
162 all protocol elements except the entity-body.

163 Note that HTTP 1.1 defines a slightly different syntax, allowing for  
164 dynamically generated messages to be transmitted. This would be  
165 required for cases such as PC driver generated Print Operations.  
166 HTTP 1.1 defines a message header which specifies a transfer encoding  
167 called "chunks".

168 IPP messages are contained within HTTP methods. The HTTP POST method  
169 is used for the Print operation and the Cancel Job operation. The  
170 HTTP GET method is used for the Get Attributes operation and the Get  
171 Jobs operation (section 5.4).

### 172 2.2.1 HTTP Request-Header Fields

173 HTTP request header fields allow the client to pass additional  
174 information about the request, and about the client itself, to the  
175 server. All header fields are optional and when used it is assumed

176 that IPP would use these headers in a standard way. IPP requests  
177 will be completely encapsulated within the entity body of an HTTP  
178 request. The HTTP Entity-Header has the form

```
179
180     HTTP-Entity-Header =      Content-Encoding
181                             | Content-Length
182                             | Content-Type
183                             | extension-header
184
```

185 The Content-Length field must always be a valid length, This means  
186 that for any Print Operations based on HTTP 1.0, the entire content  
187 must be generated before this header can be built. HTTP 1.1 provides  
188 the notion of "chunks" which will allow the content to be generated  
189 dynamically as the data is sent.

190  
191 Content-Type will always be "Application/IPP".

#### 192 2.2.1.1 IPP Request-Line

193 The first line of the entity body in an IPP operation is the IPP  
194 Request-Line. The Request-Line defines the Operation and the IPP  
195 Version.

```
196     IPP-Request-Line = Operation-token  IPP/1.0  CRLF
197
198     Operation-token  = Print | Cancel-Job |
199                       Get-Attributes | Get-Jobs
200
201
```

#### 202 2.2.2 HTTP Response-Header Fields

203 HTTP response fields allow the server to pass additional information  
204 about the response back to the client. IPP will use these headers in  
205 a standard way. IPP responses will be completely encapsulated within  
206 the entity body of an HTTP response.

#### 207 2.2.2.1 IPP Status-Line

208 The first line of the entity body in an IPP response is the IPP  
209 Status-Line. The status-line consists of a protocol version followed  
210 by a numeric status-code and an associated text message.

```
211     IPP-Status-Line = IPP/1.0 Status-Code Reason-Phrase  CRLF
```

#### 213 2.3 The Print Job

214 In section 5.4.1, the Print Operation is described. In order to  
215 understand that operation better, we first present the notion of a  
216 Print Job. The entity body of a print operation request will contain

217 a Print Job, as defined below. The headers defined here are IPP  
 218 headers, but follow the same syntax as the basic HTTP headers.

```

219
220     Print-Job = Print-Job-Object-Header ;section (5.3.1)
221                [Job-Attributes] ;section (5.3.4)
222                *(Documents)
223
224     Document = Document-Header ;section (5.3.2)
225                [Document-attributes] ;section (5.3.5)
226                [Content-Header ;section (5.3.3)
227                 content]
228
  
```

### 229 2.3.1 Print Job Object Header

```

230     Print-Job-Object Header = Content-Encoding
231                               | Content-Length
232                               | Content-Type
233                               | extension-header
234
  
```

235 Content-Type is always "IPP Print Object". Other header fields are as  
 236 defined for HTTP 1.0.

### 237 2.3.2 Document Header

238 The document header allows the insertion of multiple documents within  
 239 a job. At this point only a limited number of document attributes are  
 240 defined. However, this structure allows the addition of other  
 241 attributes which can be specified on a document boundary.

```

242     Document-Header = Content-Encoding
243                     | Content-Length
244                     | Content-Type
245                     | extension-header
246
  
```

247 Content type is always "IPP Document". Other header fields are as  
 248 defined in HTTP 1.0.

### 249 2.3.3 Document-Content Header

250 The document-content-header provides additional meta-information  
 251 about the document. The document content header is an optional field  
 252 and would not be present if the document was pointed to by a document  
 253 URL attribute. It is composed of a number of document header fields  
 254 as follows:

```

255     Document-Content-Header = Content-Encoding
256                               | Content-Length
257                               | Content-Type
258                               | extension-header
259
  
```

260 Content-Type is defined as :

261 Content-Type = Data-Stream-Format "/" Version

262

263 Thus, for example, if the document to be printed was a Postscript  
264 Level 2 document, the Content-Type would be specified as:

265 Content-Type: Postscript/2.0

266

267 Other header fields are as defined by HTTP 1.0.

#### 268 2.3.4 Job Attributes

269 Job attributes are defined in section 6.2. Attributes will always be  
270 sent as

271 Job-Attribute = Attr-name ":" Attr-value CRLF

272

273 Attr-value = 1#Value

274

275 In the above example, "1#Value" means one or more "," separated  
276 values.

#### 277 2.3.5 Document Attributes

278 Document attributes are defined in section 6.2.11. The syntax for a  
279 document attribute is

280 Document-Attribute = Attr-Name ":" Attr-Value CRLF

281

282 Attr-Value = 1#Value

283

284 In the above example, "1#Value" means one or more "," separated  
285 values.

### 286 3. Security Considerations

287 This protocol does not identify any new authentication mechanisms.  
288 The authentication mechanisms built into HTTP (such as SSL and SHTTP)  
289 are recommended.

290 This protocol does define a simple authorization mechanism by  
291 introducing the "end-user-acl" attribute as part of the Printer  
292 object. This ACL attribute is a multi-valued list of all of the  
293 authenticated names of end-users. This protocol does not specify  
294 what the domain is for names in this ACL attribute.

295 Issue: Will it always be possible for a Printer to obtain a  
296 meaningful authenticated name that the Printer can match against the  
297 end-user-acl, or will some other mechanism be necessary, such as a  
298 password?

## 299 4. References

- 300 [1] Smith, R., Wright, F., Hastings, T., Zilles, S., and  
301 Gyllenskog, J., "Printer MIB", RFC 1759, March 1995.  
302
- 303 [2] Berners-Lee, T, Fielding, R., and Nielsen, H., "Hypertext  
304 Transfer Protocol - HTTP/1.0", RFC 1945, August 1995.  
305
- 306 [3] Crocker, D., "Standard for the Format of ARPA Internet Text  
307 Messages", RFC 822, August 1982.  
308
- 309 [4] Postel, J., "Instructions to RFC Authors", RFC 1543, October  
310 1993.  
311
- 312 [5] ISO/IEC 10175 Document Printing Application (DPA), Final, June  
313 1996.  
314
- 315 [6] Herriot, R. (editor), X/Open A Printing System Interoperability  
316 Specification (PSIS), August 1995.  
317
- 318 [7] Kirk, M. (editor), POSIX System Administration - Part 4:  
319 Printing Interfaces, POSIX 1387.4 D8, 1994.  
320
- 321 [8] Borenstein, N., and Freed, N., "MIME (Multi-purpose Internet  
322 Mail Extensions) Part One: Mechanism for Specifying and  
323 Describing the Format of Internet Message Bodies", RFC 1521,  
324 September, 1993.  
325
- 326 [9] Braden, S., "Requirements for Internet Hosts - Application and  
327 Support", RFC 1123, October, 1989,  
328
- 329 [10] McLaughlin, L. III, (editor), "Line Printer Daemon Protocol"  
330 RFC 1179, August 1990.  
331
- 332 [11] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource  
333 Locators (URL)", RFC 1738, December, 1994.  
334

## 335 5. Author's Address

336  
337

338



## 339 6. Appendix A: Sample IPP Operations

340 The following examples illustrate typical flows using the IPP  
341 protocol. In these examples, the IPP Printer object named "printer-1"  
342 is located at the node identified by the DNS name "some.domain.com".  
343 A Job Template has been defined for printer-1 which establishes the  
344 print defaults.

345 For brevity in the following flows, none of the HTTP headers are  
346 shown. CRLF sequences are not shown.

## 347 6.1 Querying the printer

348 Client some.domain.com

349 ----->

351 Post http://some.domain.com/printer-1 http/1.0

352 Get-Attributes IPP/1.0

353 printer-state :

354 sides-supported :

355 media-supported :

356 document-formats-supported :

357 <-----

359 http/1.0 201 "Created" (a response)

360 IPP/1.0 xxx "attribute list returned"

361 printer-state : idle

362 sides-supported : 1-sided

363 media-supported : iso-a4-white, iso-b4-white

364 document-formats-supported : Postscript/2.0

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

## 382 6.2 Print Operation - with print data included

383 Client some.domain.com

384 ----->

385

```
386 Post http://some.domain.com/printer-1 http/1.0
387 Print IPP/1.0
388 Print-Job-Object Header
389     job-name : My Job
390     medium : iso-a4-white
391     notification-events : Job-completion
392     notification-address : joe@pc.domain.com
393 Document Header
394     document-name : Letter to Mom
395 Document-Content Header (content type = Postscript/2.0)
396     <Document in Postscript level 2 format>
397
398
399 <----->
400 http/1.0 200 "accepted"
401     IPP/1.0 xxx "print job accepted and queued"
402     job-identifier : some.domain.com/printer-1/0037
403     current-job-state : pending
404     printer-state : needs-attention
405
406 6.3 Print Operation - with no data included
```

```
407 Client some.domain.com
408
409 ----->
410 Post http://some.domain.com/printer-1 http/1.0
411 Print IPP/1.0
412 Print-Job-Object Header
413     job-name : My Job
414     medium : iso-a4-white
415     notification-events : Job-completion
416     notification-address : joe@some.domain.com
417 Document Header
418     document-name : Letter to Mom
419     document-URL : joe@pc.domain.com/Docs/To-mom.ps
420
421 <----->
422 http/1.0 200 "accepted"
423     IPP/1.0 xxx "print job accepted and queued"
424     job-identifier : some.domain.com/printer-1/0037
425     current-job-state : pending
426     printer-state : processing
427 6.4 Querying the state of the job
```

428 In this example, no attributes are specified, so all job attributes  
429 are returned.

```
430 Client some.domain.com
431
432 ----->
433 Post http://some.domain.com/printer-1/0037 http/1.0
434 Get-Attributes IPP/1.0
```

434  
435  
436 <-----  
437 http/1.0 201 "Created" (a response)  
438 IPP/1.0 xxx "attribute list returned"  
439 job-Name : My Job  
440 job-Originator : Joe@some.domain.com  
441 job-originating-host : pc.domain.com  
442 notification-address : joe@pc.domain.com  
443 job-locale : xx:xx:xx  
444 current-job-status : printing  
445 submission-time : 1996 Nov 22 1214  
446 media-sheets-completed : 2  
447

#### 448 449 6.5 Canceling a Job

450 Client some.domain.com

451 ----->  
452 Post: http://some.domain.com/printer-1/0037  
453 Cancel-Job IPP/1.0  
454

455  
456 <-----  
457 http/1.0 200 "okay"  
458 Current-job-state : terminating  
459

460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470

#### 6.6 Listing jobs on a Printer

471 List jobs on printer-1, only return job sizes. Jobs are returned in  
472 the order they are scheduled for printing. A Job-identifier attribute  
473 precedes the attributes returned for each job to delimit job  
474 boundaries.

475 Client some.domain.com

476 ----->  
477 Post http/1.0 some.domain.com/printer-1  
478 Get-Jobs IPP/1.0  
479 total-job-octets :  
480

```
481 <-----  
482 http/1.0 201 "Created" (a response)  
483     IPP/1.0 xxx "created an attribute list"  
484     job-identifier : 0033  
485     total-job-octets : 4567  
486     job-identifier : 0034  
487     total-job-octets : 12345  
488     job-identifier : 0035  
489     total-job-octets : 12356  
490
```