

1 **IPP Protocol White Paper**

2

3 1.Overview

4 IPP clients send requests to IPP printers and get responses back in
5 return. A request or a response is transmitted in one or more IPP
6 messages.

7

8 Request-Message = IPP Request-line
9 Entity-Header
10 **CRLF**
11 [Entity-Body]

12

13 Response-Message = IPP Status-line
14 Entity-Header
15 **CRLF**
16 [Entity-Body]

17

18 2.The Request-Line

19 The first line of any IPP request is the request-line. It has the form

20

21 IPP-Request-Line = Operation-token "IPP/1.0" **CRLF**

22

23 Operation-token = "Print"
24 | "Cancel-Job"
25 | "Get-Attributes"
26 | "Get-Jobs"

27

28 3. The Entity-Body

29 The data associated with an IPP request or response is transmitted in
30 the entity-body of one or more messages. An IPP entity-body also
31 contains Content-Headers which aid the receiver in interpreting and
32 responding to the data.

33

34 3.1.Content-Header Fields

35 Content-Header fields define the type of data and provide information
36 required by the receiver to properly interpret and respond to the data
37 transmitted in each entity-body.

38

39 Content-Header = Content-Type
40 | Content-Length
41 | Content-Segment-Flag
42 | Content-Sequence-Number
43 | Content-Response-Required-Flag

44

45 3.1.1. Header Syntax

46 IPP headers defined in this document conform to the rules of RFC 822.

47 All IPP headers are of the form:

48
49 IPP Header = "Content-" field-name ":" [field-value] **CRLF**.
50
51 IPP defines the octet sequence CRLF as the end-of-line marker for all
52 protocol elements except the entity-body. All IPP Headers begin with the
53 phrase "Content-".
54

56 3.1.2. Content-Type

57 The Content-Type field identifies the type of data that follows.

58
59 Content-Type = Print-Job
60 | Attribute-List
61 | Document-Part
62

63 3.1.3. Content-Length

64 Content-Length defines the length of the following content, in bytes.
65 For example, a document-part of 4096 bytes would have a content-length
66 field of the form

67 Content-Length : 4096
68

70 3.1.4. Content-Segment-Flag

71 The Content-Segment-Flag field provides a mechanism for senders to break
72 up the content of a document into pieces. This allows client to transmit
73 the document on the fly, as it is being generated, without having to
74 know the length of the entire document beforehand.
75

76 Use of a length field is preferred over the Boundary-string notion of
77 the Multipart/mixed MIME because the sender does not have to determine a
78 unique boundary string for each segment, which may be difficult for some
79 PDLs.
80

81 In addition, in combination with Content-Sequence-Number and Content-
82 Response-Required-Flag, this field enables an application to more
83 reliably recover from situations where a print request is being sent to
84 a Printer that cannot queue the document, and printing fails during
85 transmission.
86

87 Content-Segment-Flag = "only" | "first" | "middle" | "last"
88

89 A value of ONLY means that the entire document is contained within this
90 entity-body. FIRST, MIDDLE, and LAST refer to this content being the
91 first, a middle, or the last of a series of content-segments to be sent.
92 Each segment would be sent in a separate IPP message.
93

94 3.1.5.Content-Sequence-Number

95 When document content is being sent in segments, it is required that
96 each segment have an appropriate sequence number associated with it,
97 using this field. This field would have the format:

98
99 "Content-Sequence-Number" ":" integer

100

101 For example, if this segment were the third segment in a sequence of
102 content segments, the field would be

103

104 Content-Sequence-Number : 3.

105

106 3.1.6.Segment-Response-Required-Flag

107 This field allows the sender to request that a response be sent back on
108 each data segment. By setting this flag on, the sender promises not to
109 send another segment until receiving a positive response from the prior
110 segment. The receiver is obligated to send a response to each segment.
111 When the flag is off, the sender will not expect a response to each
112 segment and should send segments continuously until the entire document
113 has been transmitted. The receiver, on the other hand, would only send a
114 response if there were an error condition.

115

116

117 3.2. The Print Job

118 A Print Job contains print job attributes and one or more documents. A
119 Print Job is always terminated with an End-Of-Job-Marker. The End-Of-Job
120 Marker is required to complete a Print Request. If no End-Of-Job Marker
121 is sent, the Printer will wait for it until an established time-out
122 period has elapsed.

123

124

125 Print-Job = Print-Job-Header
126 [Job-Attribute-list]
127 1#(Documents)
128 End-Of-Job-Marker

129

130

131 3.2.1. Print-Job-Header

132 The Print-Job-Header identifies the following data as an IPP Print-Job.

133

134 Print-Job Header = "Content-Type : IPP Print-Job **CRLF**"

135

136 3.3. The Document

137 An IPP document contains the attributes of the document and optionally
138 the document content. If no content is not present, a reference to the
139 document must be provided as one of the document attributes.

140

```
141 Document = Document-Header
142           [Document-Attribute-List]
143           #1(Document-Part)
```

145 3.3.1. Document-Parts

146 An IPP Document may be split into multiple Document-Parts for
147 transmission. This makes it possible for IPP clients to send documents a
148 piece at a time, without requiring them to know the length of the entire
149 document beforehand.

150 3.3.2. Document-Content-Header Fields

152 Content-Header Fields are used to describe content of the Document-Part.

153 3.3.2.1. Content-Type

155 For document content, Content-Type is defined as :

```
156 Content-Type = Vendor "/" Data-Stream-Format "/" Version
```

158 Thus, for example, if the document to be printed was a Postscript Level
159 2 document, the Content-Type would be specified as:

```
161 Content-Type: Adobe/Postscript/2.0
```

163 3.3.2.2. Content-Length

165 For document content, Content-Length defines the length of this
166 Document-Part, in bytes.

167 3.3.3. End-Of-Job-Marker

170 An end-of-job marker is required to tell the receiver that no more
171 documents are to be sent as part of this job.

```
172 End-Of-Job-Marker = "Content-Type : End-Of-Job CRLF"
```

174 3.3.4. Attributes

176 Previous sections identified two types of attribute lists, which will be
177 further defined here.

```
178 Job-Attribute-List = Job-Attribute
179                     0#[Job-Attribute]
```

```
181 Document-Attribute-List = Document-Attribute
182                           0#[Document-Attribute]
```

```
184 Job-Attribute = Job-Informational-Attribute
185                | Job-Status-Attribute
```

```
187         | Notification-Attribute
188         | Job-Production-Attribute
189         | Conversion-Attribute
190         | Job-Resource-Attribute.
```

```
191
192     Document-Attribute = Document-format
```

```
193         | document-name
194         | document-URL
```

```
195
```

```
196 All attributes will be of the form
```

```
197
```

```
198     Attribute type = attribute value.
```

```
199
```

```
200
```

201 4. Mapping to MIME Types

```
202 If it is thought useful to map the IPP Entity-Body described in the
203 previous sections to a MIME type, the simplest approach would be to
204 define a new MIME-type, Application/IPP. Then one could simply declare
205 the Application/IPP MIME to be the IPP Message, as it has been defined
206 in this paper. However, it should be noted that this MIME type would
207 only operate within the IPP protocol.
```

```
208
```

209 5. Mapping to HTTP

```
210 If HTTP is used as the "transport" protocol, then the IPP Request
211 Message, as defined in this document, would be the Entity-Body of an
212 HTTP Post method. The IPP Response Message would be the Entity-Body of
213 the corresponding HTTP Response.
```

```
214
```

215 6. Example flows

```
216 Several examples will be shown to illustrate the use of the protocol as
217 defined in this section. Only the IPP operations and the contents of the
218 entity-bodies will be shown in these scenarios.
```

```

219
220 6.1.1.Scenario 1

221 In this scenario, a client sends a print job stored as a complete file
222 to an IPP printer implemented in a server. The server is capable of
223 spooling jobs. The job contains a single document which is received by
224 the server with no error and is queued for printing at a later time.
225
226 Client                                          Server
227
228 ----->
229     Print IPP/1.0                               ; Request-Line
230     Content-Segment-Flag : only
231     Content-Response-Required-Flag : Yes
232     Content-Type : IPP Print Job                ; Print Job Marker
233     <Job-Attribute-List>
234     Content-Type : IPP Document                 ; Document Marker
235     <Document-Attribute-List>
236     Content-Type : Adobe/Postscript/2.0
237     Content-Length : 12,150
238     <12,150 bytes of Postscript data>
239     Content-Type : End of Job                   ; End of job Marker
240
241
242 < -----
243     IPP Response = Received and Queued
244     Current-job-state = processing
245     Job-Identifier = 12
246

```

247
248

6.1.2.Scenario 2

249 This case is identical to scenario #1, except that an error occurred
250 during the transmission that made the request invalid. An error response
251 is returned.

252
253
254
255

Client

Server

256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271

```
----- >
Print IPP/1.0 ; Request-Line
Content-Segment-Flag : only
Content-Response-Required-Flag : Yes
Content-Type : IPP Print Job ; Print-Job Marker
  <Job-Attribute-List>
Content-Type : IPP Document ; Document-Marker
  <Document-Attribute-List>
Content-Type : Adobe/Postscript/2.0
Content-Length : 12,150
  <12,150 bytes of Postscript data>
Content-Type : End of Job ; End of job marker

< -----
IPP Response = Invalid Request
```

272
273 6.1.3.Scenario 3

274 This case is identical to scenario #1 except that the document to be
275 printed is being sent a piece at a time. In this case, the driver
276 generates 4K segments. This requires 2 segments of 4K each and a final
277 segment of 3,958bytes (total = 12,150 bytes). Since the server can spool
278 the data, it is not necessary to ask for a response on each segment.

279 Client

Server

```

281 ----- >
282
283     Print  IPP/1.0                               ; Request-Line
284     Content-Segment-Flag : first
285     Content-Response-Required-Flag : no
286     Content-Sequence-Number : 1
287     Content-Type : Print Job                     ; Print-Job Marker
288     <Job-Attribute-List>
289     Content-Type : IPP Document                 ; Document-Marker
290     <Document-Attribute-List>
291     Content-Type : Adobe/Postscript/2.0
292     Content-Length : 4096
293     <4096 bytes of Postscript data>
294
295 ----- >
296
297     Print  IPP/1.0
298     Content-Segment-Flag : middle
299     Content-Response-Required-Flag : no
300     Content-Sequence-Number : 2
301     Content-Type : Adobe/Postscript/2.0
302     Content-Length : 4096
303     <4096 bytes of Postscript data>
304
305 ----- >
306
307     Print  IPP/1.0
308     Content-Segment-Flag : last
309     Content-Response-Required-Flag : yes
310     Content-Sequence-Number : 3
311     Content-Type : Adobe/Postscript/2.0
312     Content-Length : 3958
313     <3958 bytes of Postscript data>
314     Content-Type : End of Job                     ; End of job marker
315
316 < -----
317 IPP Response = Received and Queued
318     Current-job-state = processing
319     Job-Identifier = 12

```

319 6.1.4. Scenario 4

320 This scenario is similar to the previous one except that the client
 321 knows the Printer is not capable of spooling the print job before
 322 starting to print. Therefore, printer errors may occur during
 323 transmission, and the job may have to be aborted. The client therefore
 324 will ask for a response on each message. In this case a printer jam
 325 occurs during the processing of the second message. The client responds
 326 with last segment containing an End-of-Job Marker, which terminates the
 327 job.

328
 329 Client

Server

```

330 ----- >
331 Print IPP/1.0 ; Request-Line
332 Content-Segment-Flag : first
333 Content-Response-Required-Flag : Yes
334 Content-Sequence-Number : 1
335 Content-Type : IPP Print Job ; Print-Job Marker
336 <Job-Attribute-List>
337 Content-Type : IPP Document ; Document-Marker
338 <Document-Attribute-List>
339 Content-Type : Adobe/Postscript/2.0
340 Content-Length : 4096
341 <4096 bytes of Postscript data>
    
```

```

342 < -----
343 IPP Response = Received
344 Current-job-state = printing
    
```

```

345 ----- >
346 Print IPP/1.0
347 Content-Segment-Flag : middle
348 Content-Response-Required-Flag : Yes
349 Content-Sequence-Number : 2
350 Content-Type : Adobe/Postscript/2.0
351 Content-Length : 4096
352 <4096 bytes of Postscript data>
    
```

```

353 < -----
354 IPP Response = Received
355 Current-job-state = printer-jammed
    
```

```

356 ----- >
357 Print IPP/1.0
358 Content-Segment-Flag : last
359 Content-Response-Required-Flag : Yes
360 Content-Sequence-Number : 3
361 Content-Type : End of Job
    
```

```

362 < -----
363 IPP Response = job Terminated
    
```

370 6.1.5. Scenario 5

371 This scenario is identical to the previous one, except that the end-user
 372 walks to the printer and clears the jam. The client starts transmitting
 373 at the next unsent document-part. The Printer must recover any pages not
 374 printed in the last document-part sent.

```

375 Client
376                                     Server
377
378 ----- >
379     Print IPP/1.0                               ; Request-Line
380     Content-Segment-Flag : first
381     Content-Response-Required-Flag : Yes
382     Content-Sequence-Number : 1
383     Content-Type : IPP Print Job                 ; Print-Job Marker
384     <Job-Attribute-List>
385     Content-Type : IPP Document                 ; Document-Marker
386     <Document-Attribute-List>
387     Content-Type : Adobe/Postscript/2.0
388     Content-Length : 4096
389     <4096 bytes of Postscript data>
390
391 < -----
392     IPP Response = Received
393     Current-job-state = printing
394
395 ----- >
396     Print IPP/1.0
397     Content-Segment-Flag : middle
398     Content-Response-Required-Flag : Yes
399     Content-Sequence-Number : 2
400     Content-Type : Adobe/Postscript/2.0
401     Content-Length : 4096
402     <4096 bytes of Postscript data>
403
404 < -----
405     IPP Response = Received
406     Current-job-state = printer-jammed
407
408 ----- >
409     Print IPP/1.0
410     Content-Segment-Flag : last
411     Content-Response-Required-Flag : Yes
412     Content-Sequence-Number : 3
413     Content-Type : Adobe/Postscript/2.0
414     Content-Length : 3958
415     <43958 bytes of Postscript data>
416     Content-Type : End of Job
417
418 < -----
419     IPP Response = job Completed
  
```

420 6.1.6. Scenario 6

421 In this scenario, the client send a print job containing two documents
 422 to the Printer. The Job is sent as one IPP Message. It is spooled on the
 423 Printer.

```

424 Client
425                                     Server
426
427 ----->
428     Print IPP/1.0                               ; Request-Line
429     Content-Segment-Flag : only
430     Content-Response-Required-Flag : Yes
431     Content-Type : IPP Print Job                ; Print Job Marker
432     <Job-Attribute-List>
433     Content-Type : IPP Document                 ; Document Marker
434     <Document-Attribute-List>
435     Content-Type : Adobe/Postscript/2.0
436     Content-Length : 12,150
437     <12,150 bytes of Postscript data>
438     Content-Type : HP/PCL/5e
439     Content-Length : 3,568
440     <3,568 bytes of PCL/5e>
441     Content-Type : End of Job                    ; End of job Marker
    
```

```

442
443
444 < -----
445     IPP Response = Received and Queued
446     Current-job-state = processing
447     Job-Identifier = 12
448
    
```

449 6.1.7. Scenario 7

450 This scenario is identical to the previous one, except that the
 451 documents are generated on the fly, in 4K blocks.

```

452
453
454 Client                                     Server
455
456 ----->
457     Print IPP/1.0                           ; Request-Line
458     Content-Segment-Flag : first
459     Content-Response-Required-Flag : no
460     Content-Sequence-Number : 1
461     Content-Type : IPP Print Job           ; Print Job Marker
462     <Job-Attribute-List>
463     Content-Type : IPP Document           ; Document Marker
464     <Document-Attribute-List>
465     Content-Type : Adobe/Postscript/2.0
466     Content-Length : 4096
467     <4096 bytes of Postscript data>
468
469 ----->
470
471     Client Print IPP/1.0                   ; Request-Line
472     Content-Segment-Flag : middle
473     Content-Response-Required-Flag : no
474     Content-Sequence-Number : 2
475     Content-Type : Adobe/Postscript/2.0
476     Content-Length : 4096
477     <4096 bytes of Postscript data>
478
479 ----->
480
481     Print IPP/1.0                           ; Request-Line
482     Content-Segment-Flag : middle
483     Content-Response-Required-Flag : no
484     Content-Sequence-Number : 3
485     Content-Type : Adobe/Postscript/2.0
486     Content-Length : 4096
487     <4096 bytes of Postscript data>
488
489 ----->
490
491     Print IPP/1.0                           ; Request-Line
492     Content-Segment-Flag : last
493     Content-Response-Required-Flag : yes
494     Content-Sequence-Number : 4
495     Content-Type : IPP Document           ; Document Marker
496     <Document-Attribute-List>
497     Content-Type : HP/PCL/5e
498     Content-Length : 3568
499     <3568 bytes of PCL/5e data>
500     Content-Type : End of Job
    
```

```
501 < -----  
502  
503     IPP Response = Received and Queued  
504     Current-job-state = processing  
505     Job-Identifier = 12  
506
```

