IPP Protocol White Paper

1.Overview

IPP clients send requests to IPP printers and get responses back in return. A request or a response is transmitted in one or more IPP messages. When multiple IPP messages are required to transport a request or a response, each message will be called a segment of that request or response.

```
Request-Message = IPP Request-line
                  Entity-Header
                  CRLF
                  [Entity-Body]

Response-Message = IPP Status-line
                   Entity-Header
                   CRLF
                   [Entity-Body]
```

2.The Request-Line

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

```
IPP-Request-Line = Operation-token  “IPP/1.0”  CRLF

Operation-token = “Print”
                | “Cancel-Job”
                | “Get-Attributes”
                | “Get-Jobs”
```

3.The Status-Line

After receiving and interpreting a request-message, a server must respond with an IPP response-message if asked. The first line of a response is the status line, consisting of the protocol version followed by a numeric status code and its associated reason-phrase. When the requested IPP operation requires data (e.g. an attribute list) to be returned in the response, the data will be contained within the entity-body of the response. Elements of the status-line are separated by space characters and the line is terminated with **CRLF**. Thus, a status line might look like:

```
“IPP/1.0” status-code text CRLF
```

3.1.Status Codes

The status code is a three digit integer result code which defines the response of the server in attempting to understand and execute the operation specified in the request. A reason-phrase provides additional, human readable information about the status condition posted.

The first digit of the reason code defines the class of response:

- 1xx: Reserved
- 2xx: Success - the request was successfully received, understood, and accepted
- 3xx: Conditional Success - Recovery may be required
- 4xx: Client Error: the request contains bad syntax or cannot be fulfilled
- 5xx: Server Error: the server failed to fulfill an apparently valid request

4.Header Syntax

IPP headers defined in this document (Entity-Headers and Content-Headers) conform to the rules of RFC 822. All IPP headers are of the form:

IPP Header = field-name “:” [field-value] **CRLF.**

IPP defines the octet sequence CRLF as the end-of-line marker for all protocol elements except the entity-body. All IPP Content-Headers begin with the phrase “Content-“.

5. Entity-Headers

Entity-Headers contain optional information which applies to the entire entity-body, and is required to correctly process the entity-body. Examples include things like data compression, security tokens, localization information, and the like. Specific Entity-headers are to be defined.

6. The Entity-Body

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

6.1.Content-Header Fields

Content-Header fields appear throughout an entity-body, and define boundary markers and protocol flags that are required to correctly parse and respond to the content of the entity-body. Five types of Content-Headers are defined, which are described in subsequent sections of this document:

```
Content-Header = Content-Type
               | Content-Length
```

```
                | Content-Segment-Flag
                | Content-Sequence-Number
                | Content-Response-Required-Flag
```

6.1.1. Content-Type

The Content-Type field is used to mark content boundaries within the entity-body, e.g. the beginning of a print job. Four boundary markers are defined in subsequent sections of this document:

```
Boundary-Marker = Print-Job-Marker
                | Document-Marker
                | Document-Part-Marker
                | End-of-Job Marker
```

6.1.2. Content-Length

Content-Length defines the length of the following data, in bytes. It is used specifically to define the length of a document-part. For example, the content-length field

```
Content-Length : 4096
```

indicates that the next 4096 bytes in the stream is document data and is not to be interpreted by the IPP process. Use of a length field is preferred over the Boundary-string notion of the Multipart/mixed MIME because the sender does not have to determine a unique boundary string for each segment, which may be difficult for some PDLs.

6.1.3.Content-Segment-Flag

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

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

```
Content-Segment-Flag = “only” | “first” | “middle” | “last”
```

A value of ONLY means that the entire document is contained within this entity-body. FIRST, MIDDLE, and LAST refer to this content being the first, a middle, or the last of a series of content-segments to be sent. Each segment would be sent in a separate IPP message. If no Content-Segment-Flag is present in a message, Content-Segment is assumed to be “ONLY”.

6.1.4.Content-Sequence-Number

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

```
"Content-Sequence-Number" ":" integer
```

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

```
Content-Sequence-Number : 3
```

6.1.5.Segment-Response-Required-Flag

This field allows the sender to request that a response be sent back on each data segment. By setting this flag on, the sender promises not to send another segment until receiving a positive response from the prior segment. The receiver is obligated to send a response to each segment. When the flag is off, the sender will not expect a response to each segment and should send segments continuously until the entire document has been transmitted. The receiver, on the other hand, would only send a response if there were an error condition. If no Segment-Response-Required-Flag is present in a message, Segment-Response-Required is assumed to be “YES”.

6.2. The Print Job

A Print Job contains print job attributes and one or more documents. A Print Job is always terminated with an End-Of-Job-Marker. The End-Of-Job Marker is required to complete a Print Request. If no End-Of-Job Marker is sent, the Printer will wait for it until an established time-out period has elapsed. A Printer that does not spool print jobs must not receive any intervening jobs until it has received an end-of-job marker for the current job, or it times out.

```
Print-Job = Print-Job-Marker
      [Job-Attribute-list]
      1#Document
      End-Of-Job-Marker
```

6.2.1. Print-Job-Marker

The Print-Job-Marker identifies the data that follows as an IPP Print-Job.

```
Print-Job-Marker = "Content-Type : IPP-Print-Job"
```

6.3. The Document

An IPP document contains the attributes of the document and optionally the document content. If no content is present, a reference to the document must be provided as one of the document attributes.

```
Document =  Document-Marker
            [Document-Attribute-List]
            0#Document-Part
```

6.3.1. Document-Marker

A Document Marker is defined as

“Content-Type : IPP-Document”

6.3.2. Document-Parts

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

```
Document-Part = Document-Part-Marker
                Document-Part-Length-Field
                Document-Part-content
```

6.3.2.1. Document-Part-Marker

A Document-Part-Marker is defined as

“Content-Type” “:” Vendor “/” Data-Stream-Format “/” Version

Thus, for example, if the document-part contained part of a Postscript Level 2 document, the Document-Part-Marker would be specified as:

Content-Type: Adobe/Postscript/2.0

An alternative scheme would be to use existing registered MIME types to identify the data stream format. Some PDLs would have to obtain registered MIME types.

6.3.2.2.Document-Part-Length-Field

For a document-part, Content-Length defines the length of this document-part, in bytes.

6.3.2.3. Document-Content

Document-content is the actual PDL of the document-part being sent.

6.3.3.End-Of-Job-Marker

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

```
End-Of-Job-Marker = “Content-Type : End-Of-Job”
```

6.3.4. Attribute-Lists

6.3.4.1.1.IPP defines three types of attribute lists:

```
Job-Attribute-List = Job-Attribute
                     0#(; Job-Attribute)

Document-Attribute-List = Document-Attribute
                          0#(; Document-Attribute)

Printer-Attribute-List = Printer-Attribute
                         0#(; Printer-Attribute)
```

Attributes are described in detail in “Internet Printing Protocol/1.0: Model and Semantics”. All attributes will be of the form

```
Attribute type = attribute value
```

7. Mapping to MIME Types

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

8. Mapping to HTTP

If HTTP is used as the “transport” protocol, then the IPP Request Message, as defined in this document, would be the Entity-Body of an HTTP Post method. The IPP Response Message would be the Entity-Body of the corresponding HTTP Response.

9.Mapping directly to TCP Sockets

To use a TCP connection to transport IPP messages, a new port number has to be defined. The suggested port number to be registered with the IANA is nnn. When using this method, an IPP server opens the socket in LISTEN mode and a client connects to it with an unused source port in the unrestricted range. This socket pair is the unique identifier for the connection. Once the connection reaches ESTABLISHED state, it transports at least one complete IPP message exchange. One complete IPP message exchange is defined by one Request-Message followed by one Response-Message. A multi-segment print message, terminated by an End-of-Job Marker, is treated as one request. After the exchange the TCP connection can be disconnected by the server but may stay up to support further transactions. An idle timer of nn minutes must be used to terminate the connection if the option of keeping it open is used.

10. Example flows

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

10.1.1.Scenario 1

In this scenario, a client sends a print job stored as a complete file to an IPP printer implemented in a server. The server is capable of spooling jobs. The job contains a single document which is received by the server with no error and is queued for printing at a later time.

```
Client                                                     Server

----------------------------------------------------------->
      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/1.0 2xx Job Received and Queued
         Current-job-state = processing        ;attribute-list
         Job-Identifier = 12
```

10.1.2.Scenario 2

This case is identical to scenario #1, except that the request is invalid for some reason. An error response is returned.

```
Client                                                     Server

---------------------------------------------------------->
      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/1.0 4xx Invalid Request
```

10.1.3.Scenario 3

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

```
Client                                                    Server

---------------------------------------------------------->
      Print  IPP/1.0                         ; Request-Line
      Content-Segment-Flag : first
      Content-Response-Required-Flag : no
      Content-Sequence-Number : 1
      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 : 4096
        <4096 bytes of Postscript data>

---------------------------------------------------------->
      Print  IPP/1.0
      Content-Segment-Flag : middle
      Content-Response-Required-Flag : no
      Content-Sequence-Number : 2
      Content-Type : Adobe/Postscript/2.0
      Content-Length : 4096
        <4096 bytes of Postscript data>

---------------------------------------------------------->
      Print  IPP/1.0
      Content-Segment-Flag : last
      Content-Response-Required-Flag : yes
      Content-Sequence-Number : 3
      Content-Type : Adobe/Postscript/2.0
      Content-Length : 3958
        <3958 bytes of Postscript data>
      Content-Type : End of Job              ; End of job marker

< ----------------------------------------------------------
      IPP/1.0 2xx Job Received and Queued
      Current-job-state = processing         ; Attribute-list
      Job-Identifier = 12
```

10.1.4. Scenario 4

This case is identical to the previous one except that the first segment contains a syntax error and cannot be processed. Since Content-Response-Required is set to NO, a response to the first segment is sent asynchronously. The server flushes any subsequently received segments. The client cannot recover so terminates the job with an end-of-job marker.

```
Client                                                    Server

---------------------------------------------------------->
      Print  IPP/1.0                          ; Request-Line
      Content-Segment-Flag : first
      Content-Response-Required-Flag : no
      Content-Sequence-Number : 1
      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 : 4096
        <4096 bytes of Postscript data>

---------------------------------------------------------->
      Print  IPP/1.0
      Content-Segment-Flag : middle
      Content-Response-Required-Flag : no
      Content-Sequence-Number : 2
      Content-Type : Adobe/Postscript/2.0
      Content-Length : 4096
        <4096 bytes of Postscript data>

< ---------------------------------------------------------
      IPP/1.0 4xx Client syntax error
      Content-Sequence-Number : 1

----------------------------------------------------------- >
      Print  IPP/1.0
      Content-Segment-Flag : last
      Content-Response-Required-Flag : yes
      Content-Sequence-Number : 3
      Content-Type : End-Of-Job

< -----------------------------------------------------------
      IPP/1.0 2xx Job Terminated
```

10.1.5. Scenario 5

The client knows the Printer is not capable of spooling the print job before starting to print. The client therefore will ask for a response on each message. In this case a printer jam occurs The client can receive the second message, but cannot continue printing. The client responds with last segment containing an End-of-Job Marker, which terminates the job.

```
Client                                                        Server

----------------------------------------------------------->
      Print  IPP/1.0                              ; Request-Line
      Content-Segment-Flag : first
      Content-Response-Required-Flag : Yes
      Content-Sequence-Number : 1
      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 : 4096
        <4096 bytes of Postscript data>

< ----------------------------------------------------------
      IPP/1.0 2xx Segment Received and Printing
         Content-Sequence-Number : 1
         Current-job-state = printing

---------------------------------------------------------->
      Print  IPP/1.0
      Content-Segment-Flag : middle
      Content-Response-Required-Flag : Yes
      Content-Sequence-Number : 2
      Content-Type : Adobe/Postscript/2.0
      Content-Length : 4096
        <4096 bytes of Postscript data>

< ----------------------------------------------------------
      IPP/1.0 3xx = Segment Received, cannot print
         Content-Sequence-Number : 2
         Current-printer-state = printer-jammed

------------------------------------------------------------ >
      Print  IPP/1.0
      Content-Segment-Flag : last
      Content-Response-Required-Flag : Yes
      Content-Sequence-Number : 3
      Content-Type : End of Job

< -----------------------------------------------------------
      IPP/1.0 2xx job Terminated
```

10.1.6. Scenario 6

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

```
Client                                                    Server

----------------------------------------------------------->
      Print  IPP/1.0                         ; Request-Line
      Content-Segment-Flag : first
      Content-Response-Required-Flag : Yes
      Content-Sequence-Number : 1
      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 : 4096
        <4096 bytes of Postscript data>

< -----------------------------------------------------------
      IPP/1.0 2xx Segment received and printing
           Content-Sequence-Number : 1
           Current-job-state = printing


---------------------------------------------------------->
      Print  IPP/1.0
      Content-Segment-Flag : middle
      Content-Response-Required-Flag : Yes
      Content-Sequence-Number : 2
      Content-Type : Adobe/Postscript/2.0
      Content-Length : 4096
        <4096 bytes of Postscript data>

< -----------------------------------------------------------
      IPP/1.0 3xx Segment received, cannot print
         Content-Sequence-Number : 2
         Current-printer-state = printer-jammed

------------------------------------------------------------ >
      Print  IPP/1.0
      Content-Segment-Flag : last
      Content-Response-Required-Flag : Yes
      Content-Sequence-Number : 3
      Content-Type : Adobe/Postscript/2.0
      Content-Length : 3958
        <43958 bytes of Postscript data>
      Content-Type : End of Job
```

```
< ------------------------------------------------------------
      IPP/1.0 2xx = Segment received and printing
            Content-Sequence-Number : 3
```

10.1.7. Scenario 7

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

```
Client                                                          Server

---------------------------------------------------------->
      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 : HP/PCL/5e
      Content-Length : 3,568
        <3,568 bytes of PCL/5e>
      Content-Type : End of Job                    ; End of job  Marker


< -----------------------------------------------------------
      IPP/1.0 2xx Job received and Queued
         Current-job-state = processing
         Job-Identifier = 12
```

10.1.8. Scenario 8

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

```
Client                                                       Server

-------------------------------------------------------------->
      Print IPP/1.0                               ; Request-Line
      Content-Segment-Flag : first
      Content-Response-Required-Flag : no
      Content-Sequence-Number : 1
      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 : 4096
        <4096 bytes of Postscript data>

--------------------------------------------------------------- >

      Client Print IPP/1.0                        ; Request-Line
      Content-Segment-Flag : middle
      Content-Response-Required-Flag : no
      Content-Sequence-Number : 2
      Content-Type : Adobe/Postscript/2.0
      Content-Length : 4096
        <4096 bytes of Postscript data>

-------------------------------------------------------------->

      Print IPP/1.0                               ; Request-Line
      Content-Segment-Flag : middle
      Content-Response-Required-Flag : no
      Content-Sequence-Number : 3
      Content-Type : Adobe/Postscript/2.0
      Content-Length : 3958
        <3958 bytes of Postscript data>

-------------------------------------------------------------- >

      Print IPP/1.0                               ; Request-Line
      Content-Segment-Flag : last
      Content-Response-Required-Flag : yes
      Content-Sequence-Number : 4
      Content-Type : IPP Document                 ; Document Marker
        <Document-Attribute-List>
      Content-Type : HP/PCL/5e
      Content-Length : 3568
        <3568 bytes of PCL/5e data>
      Content-Type : End of Job
```

```
< -------------------------------------------------------------

      IPP/1.0 2xx = Job Received and Queued
         Current-job-state = processing
         Job-Identifier = 12
```