Proposed IPP Protcol

Robert Herriot, Sun Microsystems

Introduction

This document contains a proposal for the IPP protocol. It uses MIME rules for structuring. It draws ideas from HTTP, but it is not intended to be dependent on HTTP. The primary idea drawn from HTTP is the syntax.

You should view this document as the exploration of an idea. The idea is to take the parts of HTTP that apply to printing and ignore the rest of HTTP. The idea behind this borrowing is that it becomes easy to convert the IPP protocol to the HTTP/form-data protocol.

In this spirit, this document looks at the HTTP headers and the format of a MIME type application/IPP.

The remainder of this document defines the IPP protocol in much the same fashion as HTTP is defined.

Message Types

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

```
IPP-message = Request | Response
```

Requests

IPP-messages use the syntax similar to HTTP:

```
Request =      Request-Line
               *(general-header
                |request-header
                |entity-head)
               CRLF
               [ message-body]

message-body = entity-body |
                     <entity body encode as per Transfer-Encoding>
```

Request-Line

The Request-Line is similar to HTTP but has an IPP flavor.

```
Request-Line = Method SP Request-URI SP IPP-Version CRLF
```

Method

The methods include the operations currently defined for IPP. The print operation consists of the two sub-operations, CreateJob and SendJob.

```
Method = "CREATEJOB"
       | "SENDJOB"
       | "GETATTRIBUTES"
       | "GETJOBS"
       | "CANCELJOB"
       | extension-method
```

```
extension-method = token
```

Request-URI

The Request-URI would use the IPP scheme, but would otherwise follow the syntax rules for HTTP URLs.

IPP-Version

The IPP-Version is similar to the HTTP-Version:

```
IPP-Version = "IPP" "/" 1*DIGIT "." 1*DIGIT
```

Response

IPP-messages use the syntax similar to HTTP:

```
Response =     Status-Line
               *(general-header
                |response-header
                |entity-head)
               CRLF
               [ message-body]

message-body = entity-body |
                     <entity body encode as per Transfer-Encoding>
```

Status-Line

The Status-Line is similar to HTTP but has an IPP flavor.

```
Status-Line = IPP-Version SP Status-Code SP Reason-Phrase CRLF
```

The IPP-Version is the same as for the Request-Line.

The Status-Code is a 3 digit integer and the Reason Phrase gives a short textual description of the Status-Code. They consist of the HTTP Status-Codes with a few additional ones added.

Note: The tables below are my best guess at what should borrowed from HTTP. I may have made some error where I didn't understand the purpose of an HTTP status-code. This list is much in agreement with the list that Keith created, except the numbering is different.

Informational 1xx

Status-Code	Reason-Phrase	in IPP	in HTTP
100	Continue	future	yes
101	Switching Protocols	future	yes

Successful 2xx

Status-Code	Reason-Phrase	in IPP	in HTTP
200	OK	yes	yes
201	Created	yes	yes
202	Accepted	future	yes
203	Non-Authoritative Information	no	yes

Status-Code	Reason-Phrase	in IPP	in HTTP
204	No Content	yes	yes
205	Reset Content	no	yes
206	Partial Content	no	yes
251	OK Attribute Substitution/Attribute Ignored	ISSUE: should we add this?	No

64 Redirection 3xx

Status-Code	Reason-Phrase	in IPP	in HTTP
300	Multiple Choices	no	yes
301	Moved Permanently	future	yes
302	Moved Temporarily	future	yes
303	See Other	no	yes
304	Not Modified	future	yes
305	Use Proxy	no	yes

65 Client Error 4xx

Status-Code	Reason-Phrase	in IPP	in HTTP
400	Bad Request	yes	yes
401	Unauthorized	yes	yes
402	Payment Required	yes	yes
403	Forbidden	yes	yes
404	Not Found	yes	yes
405	Method Not Allowed	yes	yes
406	Not Acceptable	yes	yes
407	Proxy Authentication Required	no	yes
408	Request Timeout	future	yes
409	Conflict	no	yes
410	Gone	yes	yes
411	Length Required	yes	yes
412	Precondition Failed	no	yes
413	Request Entity Too Large	yes	yes
414	Request-URI Too Long	yes	yes
415	Unsupported Media Type	no	yes
451	Attribute Not Implemented/Attribute Value Not Supported	yes	no

66

Server Error 5xx

Status-Code	Reason-Phrase	in IPP	in HTTP
500	Internal Server Error	yes	yes
501	Not Implemented	yes	yes
502	Bad Gateway	no	yes
503	Service Unavailable	yes	yes
504	Gateway Timeout	no	yes
505	HTTP Version Not Supported	yes	yes
551	Printer Error	yes	no

General-Header

The following general headers from HTTP have relevance for IPP: Date, Transfer-Encoding, Upgrade. The general headers that do not seem relevant are: Cache-Control, Connection, Pragma, Via.

The Date header specifies the date the message was transmitted. This a useful but optional header.

The Transfer-encoding is the mechanism used to specify that the data should be chunked. This header is useful for sending data in chunks when the client or server doesn't know the length of the data. If this is not present then Content-Length is present and specifies the length of the entity-body. The Transfer-encoding of chunked may be used in the SendJob request when a client doesn't know the length of the document data and it may be used by a server for returning GetJobs information where the server doesn't know how much information it is going to send. But the usage is not limited to these two cases.

The Upgrade header allows the client to specify other protocols it is willing to use. Perhaps, for security, we will have an IPPS scheme too. This would allow a server to specify other protocols, such as a secure IPP or HTTP.

Request-Header

The following request headers from HTTP have relevance for IPP: Accept-Charset. Accept-Encoding, Accept-Language, Authorization, From, Host, If-Modified-Since. The request headers that do not seem relevant are: Accept, If-Match, If-None-Match, If-Range, If-Unmodified-Since, Max-Forwards, Proxy-Authorization, Range, Referer, User-Agent.

The Accept header doesn't seem necessary because the IPP can only accept application/IPP objects back.

The Accept-Charset header specifies the code-sets the client accepts responses in, e.g. 8859-1.

The Accept-Encoding header specifies the encoding the client accepts responses in, e.g. zip. This may not be necessary because it does seem as if enough data comes back in responses to require encoding for compression purposes.

The Accept-Language header specifies the language the client accepts in responses.

The Authorization header handles credentials for authentication.

The From header contains the email address of the end-user submitting the request. This seems like a useful information to send along for weak authentication.

The Host header specifies the target host for cases where the URI contains the path name only. This may not be relevant for printing.

The If-Modified-Since might be useful for the GetAttributes and GetJobs methods to reduce the amount of data transmitted when nothing has changed.

Response-Header

The following entity headers from HTTP have relevance for IPP: Location, Public, Retry-after, Server, WWW-authenticate. The request headers that do not seem relevant are: Age, Proxy-Authenticate, Vary, Warning,

The Location header specifies the URL for the client to use.

The Public header contains a list of Methods supported.

The Retry-After header tells the client when to retry if the server is currently giving no service.

The Server header describes the server.

The WWW-authenticate header gives information to be used in the Authenticate header.

Entity-Header

The following entity headers from HTTP have relevance for IPP: Allow, Content-Encoding, Content-Language, Content-Length, Content-Location, Content-MD5, Content-Type The request headers that do not seem relevant are: Content-Base, Content-Range, ETag, Expires, Last-Modified.

This Allow header specifies the methods supported by the server when it returns a 405 (method not allowed) error.

The Content-Encoding header specifies the encoding (e.g. zip) applied to the entity.

The Content-Language header specifies the language of the entity.

The Content-Length header specifies the length in bytes of the entity.

The Content-Location header specifies the location of the entity and is useful in the SendJob request for specifying the name of the document.

The Content-MD5 header specifies the MD5 digest of the entity body.

The Content-Type header specifies the content type of the entity body. For many operations the content type is application/IPP. This header also has a parameter charset which specifies the code-set used in the entity-body.

Entity-Body

The entity-body varies depending on the Method.

Request

For all Requests, except SendJob, the entity-body, if present, has a Content-Type application/IPP. For SendJob Requests, the entity-body can have any Content-Type that the Printer supports, such as text/plain, application/postscript, and application/vnd.hp-PCL. If a Printer supports multiple documents per job, then it also supports the Content-Type “multipart/mixed” where each sub-entity-body has a Content-Type that the Printer supports. An IPP Version 1.0 Printer, does not support Content-Type application/IPP with a SendJob request.

If the entity-body for SendJob is absent, the print job is aborted because there is nothing to print. If the entity-body for any other operation is absent, the Printer performs the operation with default values for the operation.

ISSUE: do we support other multipart subtypes, such as related for SendJob?

Response

For all Responses, the entity-body, if present, has a Content-Type of application/IPP.

Note: GetJobs could have an entity-body whose Content-Type is multipart/mixed with application/IPP sub-entity-bodies, but it is probably simpler to allow multiple objects within an application/IPP entity-body.

Content-Type: application/IPP

This section describes the syntax for an entity-body whose Content-Type is application/IPP. Such an entity consists of one or more attributes, such that each attribute is represented like an HTTP header. The code-set of the entity is determined by the ‘charset’ parameter in the Content-Type header.

```
IPP-attribute = attribute-name “:” attribute-value CRLF
```

ISSUE: is there a reason, as in email, to allow an attribute to take up more than one line and then have some rules for coalescing multiple lines into a single attribute. That is, do we wish to restrict the line length and thus add such a rule?

The application/IPP entity body contains all IPP except for the following attributes:

- job-originating-user
- job-originating-host
- user-locale
- document-name

The job-originating-user and job-originating-host may come from the “From” header or from some authentication method. This needs more work to define the rules.

The user-locale attribute needs to be split into user-language and user-codeset. The user-language comes from the header Content-language if it is specified with CreateJob request. Otherwise it comes from the Printers printer-locale attribute which should also be split. The user-codset comes from the charset parameter to the Content-Type of the CreateJob request.

The document-name comes from the Content-Location header for each entity in the SendJob request.

The first attribute in an application/IPP entity identifies its type, i.e. the operation and whether it is a request or response. This attribute is redundant in the IPP context but may be useful when translated to an HTTP context.

```
IPP-operation = “IPP-operation” “:” IPP-operation-value
IPP-operation-value =    “CreateJob-Request”
                       | “CreateJob-Response”
                       | “SendJob-Response”
                       | “GetAttributes-Request”
                       | “GetAttributes-Response”
                       | “GetJobs-Request”
                       | “GetJobs-Response”
                       | “CancelJob-Request”
                       | “CancelJob-Response”
```

For responses, the second attribute contains the type of the object in the response.

```
IPP-object-type = “object-type” “:” object-type-value
object-type-value = “Printer” | “Job”
```

If an attribute ends with a double CRLF, then the next attribute should be an IPP-object-type attribute and that line starts the description of another object. Only entities whose IPP-operation has the value of “GetJobs-response” should have multiple objects.

The remaining attributes are as specified below for each operation.

Attribute Values

All attribute values are strings. In order to allow a client to infer a value’s abstract type, such as ‘integer’, each abstract type has a syntax that allows it to be distinguished from other types. In addition, there are three values that have special meaning and can be the value of any attribute. These values are “unknown”, “unsupported”, and “default”.

The value “unknown” shall be allowed only in a Response and it means that the Printer does not know the value of the attribute.

The value “unsupported” shall be allowed only in a Response to a GetAttributes or GetJobs operations and it means that the Printer does not support the requested attribute.

The value “default” shall be allowed only in a Request and it means that the Printer should uses its default value for the attribute.

The abstract types are: integer, integer units, Boolean, key-word, text, name, URL, range, set, date-time. Their syntax is:

```
IPP-type = integer | integer-units | Boolean | key-word
    | text | name | URL | range | set | date-time
integer = 1*10DIGIT
integer-units = 1*10DIGIT *SP 1*10ALPHA
Boolean = “true” | “false”
key-word = ALPHA 0*254( ALPHA | DIGIT | “-“ | “_” )
text = <”> *<any OCTET except the double quote <”>
```

NOTE: we need an escape rule for text

```
name = text ; appropriate for job name and document name because
            ; a user can modify them like text
            ; user-names and host-names are more like keywords
            ; because there is a finite set like key-words.
URL = scheme “:” scheme-specific-part ; RFC1738
scheme = 1(ALPHA | DIGIT | “+” | “.” | “-“)
range = range-type “..” range-type ;both range-types must be the same
range-type = integer | integer-units | date-time
set = “{“ #IPP-type “}”
date-time = [ day “,” ] date time ; dd mm yy hh:mm:ss zzz
                                  ; from RFC822 and RFC1123
day = “Mon” | “Tue” | “Wed” | “Thu” | “Fri” | “Sat” | “Sun”
date = 1*2DIGIT month 2*4DIGIT  ; day month year, e.g. 20 Jun 92
month = “Jan” | “Feb” | “Mar” | “Apr” | “May” | “Jun”
       |”Jul” | “Aug” | “Sep” | “Oct” | “Nov” | “Dec”
```

```
time = hour zone
hour = 2DIGIT “:” 2DIGIT [ “:” 2DIGIT ] ; 00:00:00 - 23:59:59 ]
zone =  “UT” | “GMT”          ; Universal Time
        | “EST” | “EDT”       ; Eastern US: -5 / -4
        | “CST” | “CDT”       ; Eastern US: -6 / -5
        | “MST” | “MDT”       ; Eastern US: -7 / -6
        | “PST” | “PDT”       ; Eastern US: -8 / -7
        | 1ALPHA              ; Military: Z = UT;
                              ; A=+1; M =+12 (J not used)
                              ; N=-1; Y=-12
        | ((“+” | “-“) 4DIGIT); Local differential,
                              ; hours and minutes (hhmm)
```

CreateJob

Request

The follow attributes are in a CreateJob Request:

```
IPP-operation: CreateJob-Request CRLF
<job attribute one per line> CRLF
```

Example 1:

```
CreateJob ipp://killtree IPP/1.0 CRLF
From: fred@foobar CRLF
CRLF
```

Example 2:

```
CreateJob ipp://killtree IPP/1.0 CRLF
From: fred@foobar CRLF
Content-Length:43 CRLF
Content-Type:application/IPP CRLF
CRLF
IPP-operation: CreateJob-Request CRLF
jobSheet:default CRLF
sides:two-sided-long-edge CRLF
```

Response

The follow attributes are in a CreateJob Response:

```
IPP-operation: CreateJob-Response CRLF
job-URL: <job-URL of new job> CRLF
job-state: <job-state value> CRLF
job-state-reasons: <job-state-reasons value> CRLF
job-state-as-text: <job-state-as-text value> CRLF
output-device-assigned: <output-device-assigned value> CRLF
submission-time: <submission-time value> CRLF
```

```
number-of-intervening-jobs: <number-of-intervening-jobs value> CRLF
number-of-intervening-jobs: <number-of-intervening-jobs value> CRLF
completion-time: <completion -time value> CRLF
job-message-from-operator: <job-message-from-operator value> CRLF
```

Issue: do we really need the job state to return if the client can follow the SendJob operation with a GetAttributes operation on the same open socket?

Example:

```
IPP/1.0 201 Created CRLF
Content-Length:65 CRLF
Content-Type:application/IPP CRLF
CRLF
IPP-operation: CreateJob-Response CRLF
Job-URL:ipp://killtree/job4567 CRLF
```

Note: the above example assumes that only the Job-URL is returned.

Issue: is there any reason to return the object type for this operation?

SendJob

Request

A SendJob Request does not have any application/IPP entities for version 1.0

Example:

```
SendJob ipp://killtree/job4567 IPP/1.0 CRLF
From: fred@foobar CRLF
Content-Length:1025 CRLF
Content-Type:application/postscript CRLF
CRLF
%!PS-Adobe-3.0 CRLF
…
```

Response

A SendJob Response may return no entity body:

Example:

```
IPP/1.0 200 OK CRLF
CRLF
```

Note: the above example assumes that no job state information is returned.

GetAttributes

Request

The follow attributes are in a GetAttributes Request:

```
IPP-operation: GetAttributes-Response CRLF
requested-format: <document format> CRLF
requested-attributes: <set of attribute names> CRLF
```

The last two attributes are optional.

For example 1:

```
GetAttributes ipp://killtree IPP/1.0 CRLF
From: fred@foobar CRLF
Content-Length:125 CRLF
Content-Type:application/IPP CRLF
CRLF
IPP-operation: GetAttributes-Request CRLF
requested-format:application/postscript CRLF
requested-attributes:jobTemplate CRLF
```

Issue: is the formst “application/postscript”?

For example 1:

```
GetAttributes ipp://killtree/job345 IPP/1.0 CRLF
From: fred@foobar CRLF
Content-Length:61 CRLF
Content-Type:application/IPP CRLF
CRLF
IPP-operation: GetAttributes-Request CRLF
requested-attributes:job-state CRLF
```

Response

The follow attributes are in a GetAttributes Request:

```
IPP-operation: GetAttributes-Response CRLF
object-type: <Printer or Job> CRLF
<requested attributes> CRLF
```

For example 1:

```
IPP/1.0 200 OK CRLF
Content-Length:546 CRLF
Content-Type:application/IPP CRLF
CRLF
IPP-operation: GetAttributes-Response CRLF
object-type:Printer CRLF
job-sheets-supported:default CRLF
job-sheets:default CRLF
notification-events-supported:{job-completion,job-canceled,job-
      problem} CRLF
notification-events:{job-completion,job-canceled} CRLF
job-priority-supported:20..50 CRLF
job-priority:40 CRLF
...
```

Note: the attributes without the “supported” suffix are the default values.

For example 2:

```
IPP/1.0 200 OK CRLF
Content-Length:72 CRLF
Content-Type:application/IPP CRLF
CRLF
IPP-operation: GetAttributes-Response CRLF
object-type:Job CRLF
job-state:processing CRLF
```

GetJobs

Request

The follow attributes are in a GetJobs Request:

```
IPP-operation: GetJobs-Response CRLF
requested-user:  <owner of job> CRLF
requested-states: <set of states> CRLF
requested-attributes: <set of attribute names> CRLF
```

Note: if we allows job and printer attributes to be retrieved with GetJobs, then the requested-attributes needs to be changed to requested-job-attributes and requested-printer-attributes.

The last three attributes are optional.

For example:

```
GetJobs ipp://killtree IPP/1.0 CRLF
From: fred@foobar CRLF
Content-Length:154 CRLF
Content-Type:application/IPP CRLF
CRLF
IPP-operation: GetJobs-Request CRLF
requested-states:{pending,processing} CRLF
requested-attributes:{job-originating-user,job-state,submission-
      time,completion-time,foo} CRLF
```

Response

The follow attributes are in a GetJobs Request:

```
IPP-operation: GetJobs-Response CRLF
object-type: <Printer or Job> CRLF
<requested attributes> CRLF
CRLF
object-type: <Printer or Job> CRLF
<requested attributes> CRLF
CRLF
… CRLF
```

For example:

```
IPP/1.0 200 OK CRLF
Content-Length:333CRLF
Content-Type:application/IPP CRLF
CRLF
IPP-operation: GetJobs-Response CRLF
object-type:Job CRLF
job-originating-user:"fred@foobar" CRLF
job-state:processing CRLF
submission-time:Thu, 04 May 1997 17:34:03 PDT CRLF
completion-time:unknown CRLF
foo:unsupported CRLF
CRLF
object-type:Job CRLF
job-originating-user:"barb@bazz" CRLF
```

```
job-state:pending CRLF
submission-time:Thu, 04 May 1997 17:47:18 PDT CRLF
completion-time:unknown CRLF
foo:unsupported CRLF
```

CancelJob

Request

The follow attributes are in a CancelJob Request which is submitted to a job URL:

```
IPP-operation: CancelJob-Request
message: <text of message>
```

Example 1:

```
CancelJob ipp://killtree/job123 IPP/1.0 CRLF
Authorization:A3Z35MLQ3ee93 CRLF
Content-Length:55 CRLF
Content-Type:application/IPP CRLF
<CRLF>
IPP-operation: CancelJob-Request CRLF
message: run-away job CRLF
```

Example 2:

```
CancelJob ipp://killtree/job124 IPP/1.0 CRLF
From:fred@foobar CRLF
CRLF
```

Response

The CancelJob response does not have an entity-body. The Status-Code and Reason-Phrase are sufficient.

Example 1:

```
IPP/1.0 200 OK CRLF
```

Translation to HTTP

It is easy to translate from IPP to HTTP.

On the Request-Line the method is changed to POST, the scheme is changed from “ipp” to “http”, and the version is changed from “IPP/1.0” to “HTTP/1.1”.

On the Status-Line, the version is changed as on the Request-Line.

For operations without any application/IPP, no other changes occur. If there is an application/IPP entity-body it is converted to an multipart/form-data entity-body. Each attribute in the application/IPP entity becomes a sub-entity with a ‘content-disposition’ header containing the attribute name and a sub-entity-body containing the value. The double CRLF that delimit objects for GetJobs becomes a content-disposition entity with a name of “” and an empty sub-entity-body.

The inverse operation (HTTP to IPP) is self evident.

Table of Header Usage in Operations

A value of ‘r’ means that the header is required and a value of ‘o’ means that the header is optional. If the cell is empty, then the header is not expected to be present.

	CreateJob		SendJob		GetAttributre		GetJobs		CancelJob	
	Req	**Resp**	**Req**	**Resp**	**Req**	**Resp**	**Req**	**Resp**	**Req**	**Resp**
Date	o	o	o	o	o	o	o	o	o	
Transfer-encoding [1]			o					o		
Upgrade	o				o		o		o	
Accept-Charset.	o		o		o		o			
Accept-Encoding	o		o		o		o			
Accept-Language.	o		o		o		o			
Authorization	o		o		o		o		o	
From	o		o		o		o		o	
Host	o		o		o		o		o	
If-Modified-Since					o		o			
Location		o		o		o		o		o
Public		o		o		o		o		o
Retry-After		o		o		o		o		o
Server		o		o		o		o		o
WWW-Authenticate		o		o		o		o		o
Allow [2]		o		o		o		o		o
Content-Encoding	o	o	o	o	o	o	o	o	o	
Content-Language	o	o	o	o	o	o	o	o	o	
Content-Length[3]	r	r	r	r	r	r	r	r	r	
Content-MD5	o		o		o		o		o	
Content-Type[4]	r	r	r	r	r	r	r	r	r	

428

[1] Transfer-encoding means chunking and is only necessary for large transmissions.
[2] Actually 'Allow' is in a response only when the server receives a bad method and returns status-code 405,
[3] If Transfer-encoding of chunked is used, then Content-Length is not required.
[4] The Content-Type also contains the charset used in the entity body.