Proposed Internet Draft
October 1996

R.K. deBry, IBM Boulder
K. Carter, IBM Austin
J. Barnett, IBM Somers

Initial Draft - Hypertext Printing Protocol - HTPP/1.0

1. Status of this Memo

This document is a proposed Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF) 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”.

2. Abstract

The Hypertext Printing Protocol (HTPP) is a lightweight, application-level protocol designed to accomplish the submission and management of printing over the Internet. Distributed printing systems require more functionality than simply submitting a print job, including retrieving the status of a print job, investigating the status of a printer or print queue, notification of job complete, and allowing job operations such as cancel a job. HTPP allows an open-ended set of methods to be used to indicate the purpose of a request. It builds on the discipline of reference provided by the Uniform Resource Location (URL) and message formats similar to those used by Internet Mail and the Multipurpose Internet Mail Extensions (MIME).

3. Introduction

The purpose of this paper is to initiate discussion on a proposed Internet Printing Protocol standard. Although many protocols exist for printing within a LAN environment, most of these are proprietary and product specific (e.g. Warp Server printing, Windows NT printing, Novell Printing). The ISO DPA standard does define an open standard for printing within a distributed environment. However, current implementations of the DPA standard are too heavy to meet all of the needs of printing in the dynamic environment of the Internet. At this point in time the authors know of no standards work specific to printing on the internet.

The notion of advertising and locating print resources in an internet environment is important, but will not be covered in this initial discussion. There is an Internet Engineering Task Force draft on "Service Location Protocol"[1] available from Charles Perkins at IBM research that describes this problem and one proposed solution. Others may exist. The remainder of this discussion assumes that a suitable printer has been located and is available. Printer in this case may be a *logical* printer, since the address known to the application desiring to print may actually be that of a print server or of a print queue capable of rendering the output on any number of physical devices.

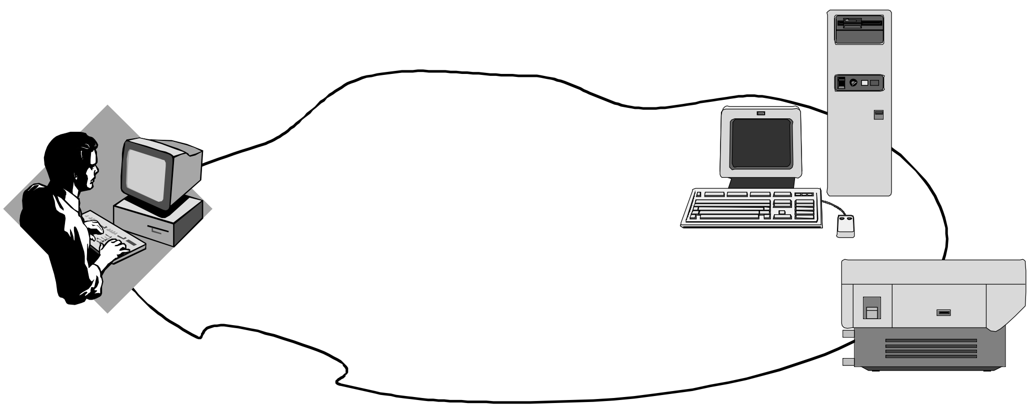
The HTTP protocol[2], upon which this standard is based, provides a simple challenge-response scheme which may be used by a server to challenge a client request and by a client to provide authentication information. This scheme is based on the model that the user agent must authenticate itself with a user-ID and a password for each realm. The realm value should be considered an opaque string which can only be compared for equality with other realms on that server. The server will authorize requests only if it can validate the user-ID and password for the protection space of the request-URL. This authentication scheme is a non-secure method of filtering unauthorized access to resources on an HTPP server.

4. Requirements

The following sections describe various printing scenarios for the internet. The flows shown are not intended to depict the proposed standard, but rather to describe the requirements for printing over the internet. These scenarios will clarify and validate the print operations that the standard should support.

4.1 Pushing a Formatted Document from a Client

Consider the case where a document exists and has been formatted for a particular output device. The print data would be in a traditional page description language, such as Postscript. The following flows of information would need to take place between the client a print server located on the internet.



"Here is a print job"

- Job Information (job owner, job name, etc.)
- Production Instructions (staple, duplex, etc.)
- Print Data

"I got it"

- What I did with it or error condition
- Job ID on queue/server

o
o
o

"What happened to my print job?"

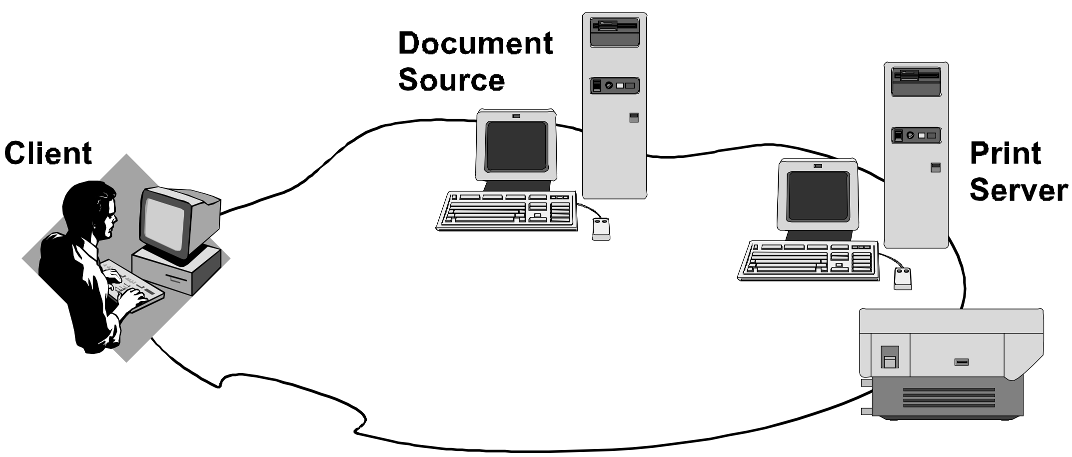
"Here is the status of you job (on the queue, printing, held, etc.)

o
o
o

"Print Complete (or some error condition)"

4.3 Pulling a Formatted Document

In a variation of the previous case, the document to be printed may be pulled from the client by the server when the print job is established. Also, in the internet world, a user will often want to print a document that is found at another location. Rather than bring the document to the client workstation, it should be possible to pull the document from the original source when printing. This scenario is described below:



"Here is a print job"

- Job Information (job owner, job name, etc.)
- Production Instructions (staple, duplex, etc.)
- Document URL

"I got it"

- What I did with it or error condition
- Job ID on queue/server

o
o
o

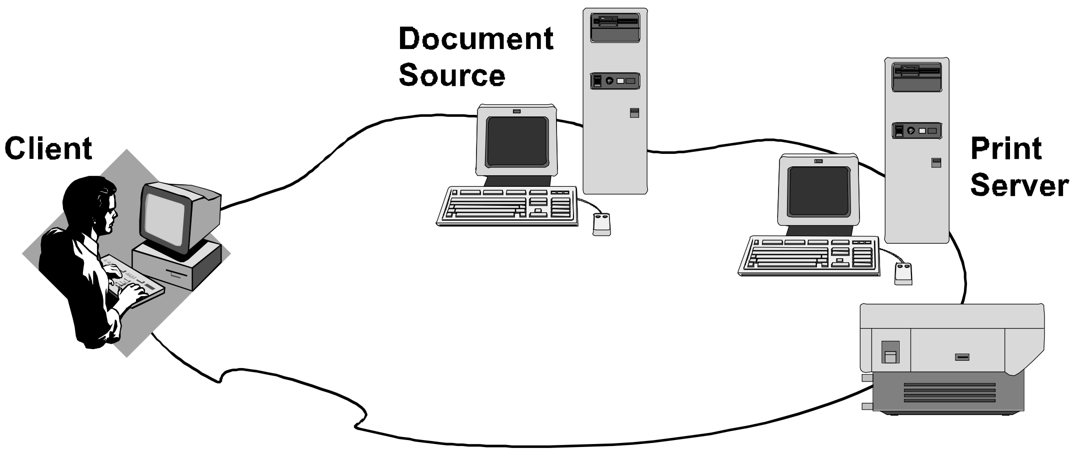
"Give Me the Print File"

"Here is the print file"

o
o
o

4.4 Pulling a Document Purchased From Another Source

In the previous case, the document to be printed might have actually been purchased from some source such as a for-profit digital library. Printing the document might now require additional protocols such as shown below:



“Here is a print job”

- Job Information (job owner, job name, etc.)
- Production Instructions (staple, duplex, etc.)
- Document URL
- “Proof of Purchase”

“I got it”

- What I did with it or error condition
- Job ID on queue/server

o
o
o

“Give Me the Print File”

- “Proof of Purchase”

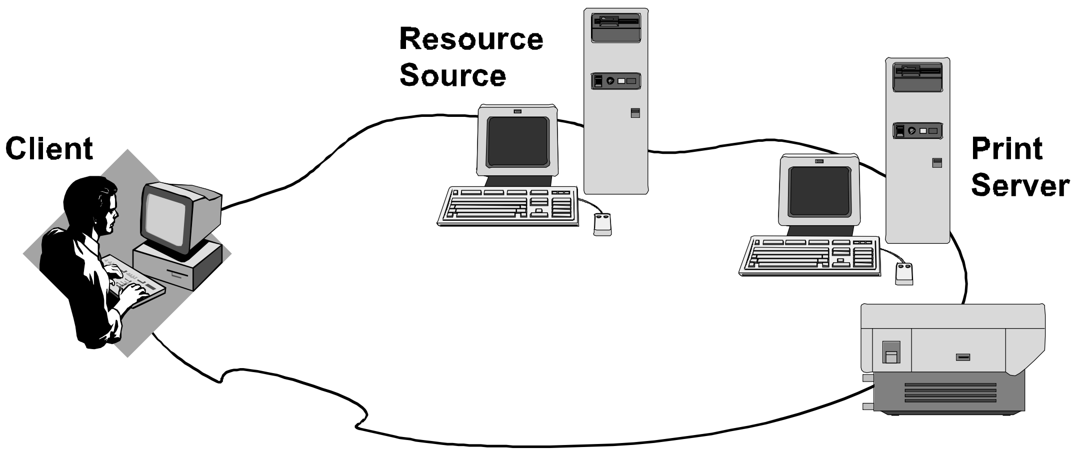
“Here is the print file”

o
o
o

4.5 Printing a Complex HTML Document

The above case might be made even more complex when assuming that the data being printed is defined in HTML, and bits and pieces of the document are stored as separate resources which might actually be scattered around the internet. These pieces might be images or other HTML files. It is assumed that problem of printing collections of HTML pages is solved. The following assumes that interpretation of the HTML takes place in the print server, but it might also take place in the printer.

If the HTML is being interpreted in the printer, the printer may contain an HTTP agent and interact directly with the Internet to obtain the files to be printed, or it may interact with its print server using proprietary protocols, and the print server will act as a proxy for the printer to retrieve the files from the internet.



"Here is a print job"

- Job Information (job owner, job name, etc.)
- Production Instructions (staple, duplex, etc.)
- HTML Print Data

"I got it"

- What I did with it or error condition
- Job ID on queue/server

o
o
o

"Give Me the Resource "

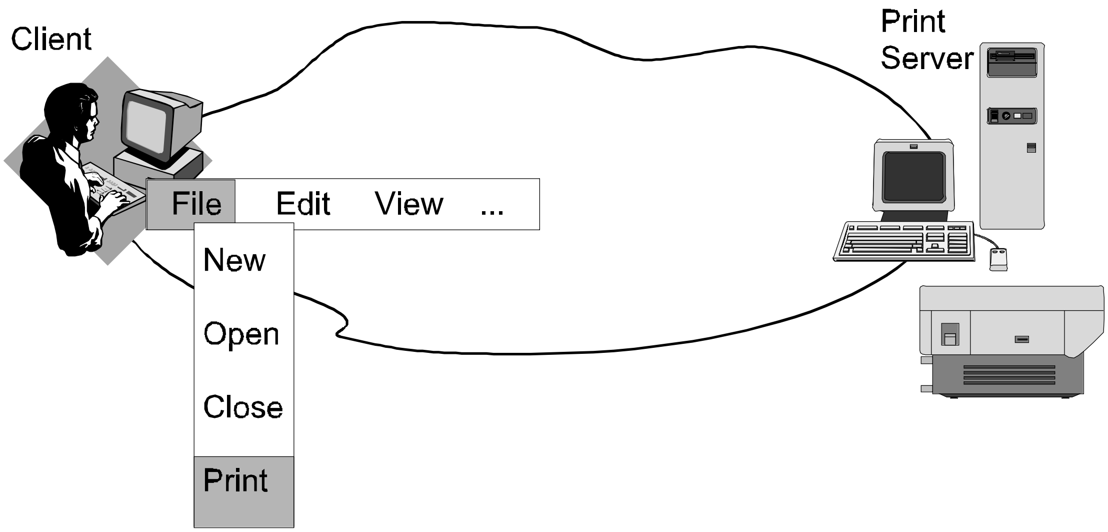
- URL

"Here is the resource"

o
o
o

4.6 Printing From an Application

Finally, we describe the case where an application generates the print file using “standard” print drivers and a graphics engine on the client.



At this point, there are probably a couple of options.

- *Client knows it has a driver for the selected printer, and generates print file*
  - *remainder of the flow is as in previous cases*
- *Client does not have driver, gets one from the print server*
  - *would require a flow to get the driver*
  - *concern is network load and performance*
- *Client does not have a driver, generates some agreed to canonical form (metafile, html, pdf)*
  - *as in previous case, print file is labeled as metafile or html or …*

5. Objectives

The protocol described here should meet the following objectives:

- Platform Independent **-** should not be based on a particular vendor's platform protocol, e.g. SMBs
- Should sit directly on top of TCP/IP and not require additional transport or distributed middleware
- Can ride on other reliable transports for heterogeneous intranet implementation
- Will use existing Internet standards where appropriate
- Will be based on the ISO 10175 (DPA) Standard
- Will be support the Printer Working Group Printer and Job MIB
- Will be extensible
- Will be easily transformable to printer specific protocols, e.g. NPAP, IPDS, or PJL, operating between the print server and the actual print hardware.

6. Syntax

The protocol proposed here is based on the existing HTTP standard[2,3,4] and for purposes of discussion is called the *HyperText Printing Protocol,* HTPP. HTPP is a lightweight application-level protocol designed with the Internet in mind and is proposed as an Internet Standard. Based on HTTP, it is a generic, stateless, object-oriented protocol which can be used for any print-related task through extension of its request methods (commands).

Distributed printing systems require more functionality than simply submitting a print job, including retrieving the status of a print job, investigating the status of a printer or print queue, notification of job complete, and allowing job operations such as cancel a job. HTPP allows an open-ended set of methods to be used to indicate the purpose of a request. It builds on the discipline of reference provided by the Uniform Resource Location (URL) and message formats similar to those used by Internet Mail and the Multipurpose Internet Mail Extensions (MIME).

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

On the internet, the communication generally takes place over a TCP/IP connection. The default port is TCP 80, but other ports can be used. This does not preclude HTPP from running over other transport protocols on the internet or on other networks. HTPP only presumes a reliable transport. The mapping of the HTPP request and response structites onto the transport data units is beyond the scope of this specification.

Current practice requires that the connection be established by the client prior to each request and closed by the server after sending the response. Both clients and serevsr must be capable of handling cases where either party closes the connection prematurely, due to user action, automated time out, or program failure.

6.1 Terminology

This specification uses a number of terms to refer to the roles played by participants in, and objects of, the HTPP communication.

Connection **-** A virtual circuit established between two parties for the purpose of communication.

Message **-** A structured sequence of octets transmitted via the connection as the basic component of communication.

Request **-** An HTPP request message.

Response **-** an HTPP response message.

Resource **-** A network data object or service which can be identified by a URL.

Entity **-** A particular representation or rendition of a resource that may be enclosed within a request or response message. An entity includes meta-information in the form of entity headers and content.

Document **-** The actual data to be rendered on the printer. It may be in one of many standard page description formats.

Job Ticket **-**An entity that describes the production rules for a document.

Print Job - One or more documents, their job tickets, and job data.

Job Data - Data specific to this print job, which will be reflected in the job MIB.

Client **-** A program that establishes connections for the purpose of sending a request.

User agent **-** The client program which is closest to the user which initiates the request.

Server **-** A program that accepts connections in order to service requests.

Origin server **-** The server on which a given resource (such as a print file) resides.

6.2 Notational Conventions and Generic Grammar

All of the mechanisms specified in this document are described in prose and modified BNF notation as described in the HTTP specification and RFCs 822[5] and 1521[6].

6.2.1 HTPP Messages

HTPP messages consist of requests from client to server and responses from server to client.

```
HTTP MESSAGE = Request | Response
```

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

```
Request = Request-line
          * (General-Header
          |  Request-Header
          |  Entity-Header)
          CRLF
          [ Entity-Body ]

Response = Status-line
          * (General-Header
          |  Request-Header
          |  Entity-Header)
          CRLF
          [ Entity-Body ]
```

All HTPP headers conform to the syntax

```
HTPP Header = field name “:” [field-value] CRLF.
```

HTPP/1.0 defines the octet sequence CR LF as the end-of-line marker for all protocol elements except the entity-body. In this document, the sequence CR LF is shown as **CRLF**.

6.3 HTPP Versions

HTPP uses a "<major>.<minor>" numbering scheme to indicate versions of the protocol. The protocol versioning policy is intended to allow the sender to indicate the format of the message and its capacity for understanding further HTPP communication. No change is made to the version number for the addition of message components which do not affect communication behavior or which only add to extensible field values. The <minor> number is incremented when the changes made to the protocol add features which do not change the general message parsing algorithm, but which may add to the message semantics and imply additional capabilities of the sender. The <major> number is incremented when the format of a message within the protocol is changed.

Applications sending Request or Response messages, as defined by this specification, MUST include an HTPP-version of HTPP/1.0. When an application receiving a request cannot support the HTPP version indicated, it must respond to the request with a return code of '505', HTPP version not supported.

6.4 Request

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource specific, the identifier for the resource, and the protocol version in use.

6.4.1 The Request-Line

The request line begins with a method token, followed by the request URL and the protocol version, and ends with **CRLF**. The elements are separated by space characters. No CR or LF is allowed except in the final **CRLF** sequence.

6.4.1.1 HTPP Methods

The method token indicates the method to be performed on the resource identified by the Request-URL. The method is case-sensitive. The methods described here reflect the end user abstract print operations of ISO 10175[7]. Administrative operations (PromoteJOb, PauseJob, ResumeJob) are candidates for further extensions to the standard.

```
Method = "Print" | "ModifyJob" | "CancelJob" |
         "ListObjectAttributes" | "GetPrintFile"
```

The list of methods acceptable by a specific resource can change dynamically; the client is notified if through the return code of the response if the method is not allowed on the resource. Servers should return the status code 501 (not implemented) if the method is unknown or not implemented.

6.4.1.1.1 Print

The Print method allows a user to submit a print request to the print server. A print request contains the information needed by the print server to print a particular document or set of documents. When the print method is invoked, the Entity-Body included in the Request is a Print Job (see section 6.5).

6.4.1.1.2 ModifyJob

The ModifyJob method is used to alter the values of specified job and document attributes after the job has been established by the print server. This operation may impact the scheduling of the job.

The effect of modifying a job shall be the same as if the job were originally submitted with the modified attributes, whether the client had supplied values for those attributes in the original Print request or the server supplied default values.

The enitity-body included in the ModifyJob request must contain the **Print-ID-On-Server** value returned in the initial response to the Print request (page 20). The entity-body included with the ModifyJob request is an attribute-entity, composed of an Object-Identifier-field followed by an attribute list.

```
Attribute-Entity = Object-Identifier-field  Attribute-List

Object-Identifier-field  = Print-ID-On-Server
                           | Printer name
                           | Queue name
                            CRLF

Attribute-List = *(Attribute “:” [value] CRLF)
```

Thus,

```
Print-ID-On-Server: 37 CRLF
Priority: 3 CRLF
```

would change the priority of print job number 37 to be a 3.

6.4.1.1.3 CancelJob

This method allows a user to cancel one specific print job request, or one document in a multi-document job, any time after the print job has been established on the server. Some pages may be printed before a job is terminated if printing has already started when the CancelJob request is received.

The enitity-body included in the CancelJob request will contain the **Print-ID-On-Server** value returned in the initial response to the Print request (page 20).

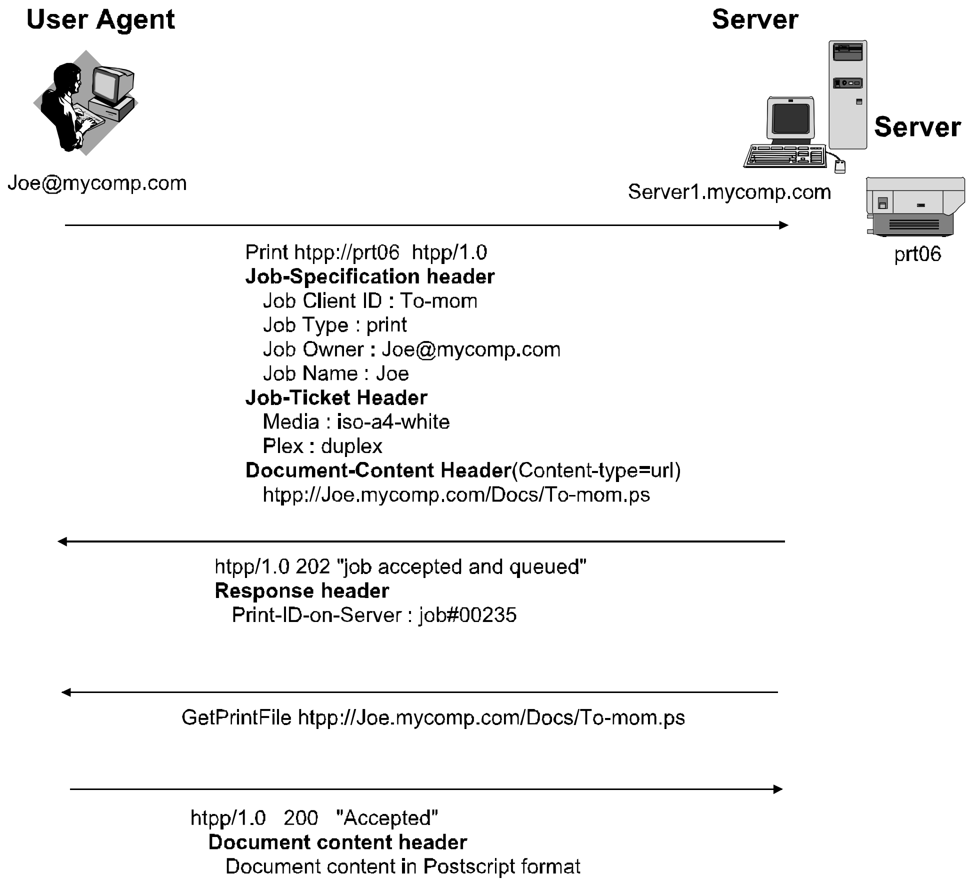
6.4.1.1.4 ListObjectAttributes

This method allows a user to obtain information from the server concerning jobs, printers, and print queues, based on ISO 10175. The entity-body of the ListObjectAttributes is an attribute-entity, as described in ModifyJob. However, the attribute values may be null and are ignored by the server. The attribute-entity is returned in the response with the appropriate attribute values filled in. If no attribute list is supplied, then all attributes defined for that object are returned.

If the Object-Identifier-field is a **Printer Name,** then the attributes that may be specified are as defined in the Printer MIB. If the Object-Identifier-field is a **Print-ID-On-Server**, or **Queue Name**, then the attributes that may be specified are as defined in the Job MIB and as defined in section 6.5.1.1, Job Specification. For a queue, a list of jobs will be returned in the order in which they appear on the queue.

6.4.1.1.5 GetPrintFile

If a Print request specifies a document URL, the GetPrintFile is used to fetch the document when the server is ready to process the print job. The GetPrintFile request is sent to the URL specified in the original print request. A sample flow would be as shown below:



6.4.1.2 Request-URL

The Request-URL is a Uniform Resource Location and identifies the resource upon which to apply the request. The URL identifies the printer and/or print queue, using the following notation:

```
Request URL = Printer Name | IP address of Printer | Queue Name
```

For example, the request line

```
PRINT HTPP://PSM_B3 HTPP/1.0
```

will print the job described in this request on the printer PSM_B3 attached to the server that this request is directed to.

6.4.2 Request-Header Fields

The request header fields allow the client to pass additional information about the request, and about the client itself, to the server. All header fields are optional.

```
Request Header = Authorization | From | Referer | User-Agent
```

Authorization: A user agent that wishes to authenticate itself with a server – usually, but not necessarily after receiving a 401 response (unauthorized) – may do so by including an authentication header field with the request. The authorization field value consists of credentials containing the authentication information of the user agent for the realm of the resource being requested.

```
Authorization = “Authorization” : credentials
```

From: The From header field, if provided should contain an internet e-mail address for the human user who controls the requesting user agent. The address should be machine readable, as defined in RFC 822 and updated in RFC 1123. An example is

```
From: debry@vnet.ibm.com
```

The server may use this address for notification purposes.

Referer: The referer request header allows the client to specify, for the server’s benefit, the address (URL) of the resource from which the request-URL was obtained. In the case of pulled print requests, this allows the server pulling the print data to point to the client originating the print request.

User Agent: The user-agent field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. The field can contain multiple product tokens and comments, identifying the agent and any subproducts which form a significant part of the user agent. By convention, the product tokens are listed in order of their significance for identifying the application. For example,

```
User-Agent: IBM WarpServer/4.0
```

6.5 The Print-Job

Most of the methods described in this standard deal with print jobs. A print job is defined as

```
Print Job = Job-Specification-Header
            [Job Specification]
            *(Documents)


Document = [Job-Ticket-Header
            Job-Ticket]
            Document-Content


Document-Content = Content-Header
                   URL | Content
```

Thus, a print job may be made up of one or more documents, where each document may optionally have job-ticket information preceding it. The parameters of a job-ticket are valid until another job-ticket or the end of the job is encountered.

6.5.1 Job Specification

The job specification is composed of a set of Job-Specification-Header fields, followed by the actual job parameters as a string of *type : value* pairs. The syntax of the Job-Specification is

```
Job-Specification = *(Job-Specification-Header field)
                    CRLF
                    *(Job-Specification-Attribute)
```

Job-Specification-Header fields provide meta-information about the Job-Specification. The syntax of the Job-Specification-Header follows the general form for entity headers in HTTP, and is

```
Job-Specification-Header = Content-Encoding
                         | Content-Length
                         | Content-Type
                         | extension-header
```

Content-Encoding allows the entity content to have additional coding defined on it. This might include compression, or encryption. An example would be

```
Content-Encoding: x-gzip.
```

Content-Length indicates the size of the Job-Specification data in decimal number of octets sent to the recipient. An example might be

```
Content-Length: 4098.
```

Content-Type is provided for consistent definition of entity headers. In this case, Content-Type is always "text" in this case.

The extension-header mechanism allows additional Job-Specification-Header fields to be defined.

6.5.1.1 Job Specification Attributes

Job specification attributes provide identification information about the print job. These attributes are intended to match the parameters defined in the Printer Working Group Job MIB [8], and are based on ISO 10175.

```
Job-Specification-Attribute = Attribute-type: [value] CRLF
```

The Job specification attributes are defined below. Fields are optional and their usage may depend upon the request method. For example, only Print-ID-on-Server is required when querying the status of a job. These definitions are subject to change as the definition Job MIB is finalized.

Job Client ID: A human readable name for the job, supplied by the originating client, e.g. the application program generating the print file.

Job upstream ID: Uniquely identifies the job on the client upstream of the print server.

Job Type: Specifies the type of service (print, fax, …)

Job Owner: Name of the Human owner of the print job.

Job Name: Human readable string for naming the job. This would normally be printed on a separator sheet for the job.

Date/Time: Date and Time that the job was submitted. Date and Time will be specified as in RFC 1123[9].

Job Comment: Arbitrary human readable string associated with this print job.

Job Priority: a priority for job scheduling purposes

Print-ID-On-Server: is returned when a new print job is submitted. It provides the job ID assigned by the server, and should be used in subsequent requests that deal with that job.

Print-ID-On-Printer: is returned when a print job has been transmitted to the printer. It allows operations, e.g. Cancel, to be performed on jobs that are in the printer.

6.5.2 The Job Ticket

The job ticket is composed of a set of Job-Ticket-Header fields, followed by the actual job ticket attributes. Job Ticket attributes provide production information for document(s) that follow. The syntax of the Job-Ticket is

```
Job-Ticket = *(Job-Ticket-Header field)
              CRLF
              *(Job-Ticket-Attribute)
```

Job-Ticket-Header fields provide meta-information about the Job-Ticket. The syntax of the Job-Ticket-Header follows the general form for entity headers in HTTP, and is

```
Job-Ticket-Header =      Content-Encoding
                         | Content-Length
                         | Content-Type
                         | extension-header
```

These fields are defined as for the Job-Specification-Header on page 16.

6.5.2.1 Job-Ticket Attributes

Job ticket attributes provide production information about the print job. These parameters are based on ISO 10175, and the precedence rules of ISO 10175 apply.

```
Job-Ticket-Attribute = Attribute-type: [value] CRLF
```

The Job ticket attributes are defined below.

Media: defines the media to be used in printing this document. Media is specified by the type, size, color and dimension. For example,

```
Media: iso-a4-white
```

Input-Tray: specifies the input tray from which paper is to be taken in printing this document. The following standard values are defined:

```
Input-Tray = top | middle | bottom | envelope |
             manual | large-capacity | main | side
```

Print-Resolution: defines the desired output resolution. Standard values include Lowest, medium, and highest. The client may query the printer MIB to determine actual resolution parameters supported, and these may be used instead. If the document contains imbedded PDL commands which affect resolution, they override this specification.

Plex (sides): defines the number of sides to be printed, and the relative orientation of consecutive page images. Standard values are

```
Plex = simplex | duplex | tumble
```

Print-Quality: defines the desired output quality. Standard values include

```
print-quality = draft | normal | high
```

Copy-count specifies the number of copies of this document to be printed
For example, to print 6 copies of a document, one would specify

```
Copy-count: 6
```

Content-orientation: allows the specification of several standard orientations of the content on the printed media. Values include

```
Content-Orientation = portrait
                    | landscape
                    | reverse-portrait
                    | reverse-landscape
```

Finishing: specifies finishing operations to be performed on the output. Standard finishing values include:

```
Finishing = staple | stitch | punch | cover |
            wrap | bind
```

6.5.3 The Document-Content Header

The document-content-header provides additional meta-information about the document. It is composed of a number of document header fields as follows:

```
Document-Content-Header =     Content-Encoding
                              | Content-Length
```

```
| Content-Type
| extension-header
```

The definition of these fields is as defined for the Job Specification Header on page 16, with the exception of the Content-Type. Content-Type is defined as :

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

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

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

6.6 Responses

After receiving and interpreting a request, a server responds in the form of an HTPP response message. 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. Elements are separated by space characters and the line is terminated with **CRLF.** Thus, a status line might look like:

```
HTPP/1.0 status-code text crlf
```

6.6.1 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 method 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 action was successfully received, understood, and accepted
- 3xx: Reserved
- 4xx: Client Error: the request contains bad syntax or cannot be fulfilled
- 5xx: Server Error: the server failed to fulfill an apparently valid request

6.6.1.1 200 OK

The request was successful. The information returned with the response is dependent on the method used in the request, as follows:

Print: This status code would not be returned for the print method. Acceptance of a print job is signaled with the 202 code, which indicates that the job has been accepted but not printed. A client will be notified of completion (or failure) of a print job if a **FROM** request-header field was included in the request, or by issuing a ListJobAttributes method requesting status of the print job. .

ModifyJob: No data returned.

CancelJob: No data returned.

ListJobAttributes: returns the list of *attribute: value* pairs asked for in the request.

6.6.1.2 202 Accepted

This is the normal response when submitting a print job. The 202 response indicates that the request has been accepted for processing, but the processing has not been completed. The processing may or may not actually be acted upon, as it may be disallowed when processing actually takes place or the processing fails for some reason. A client will be notified of completion (or failure) of a request if a **FROM** request-header field was included in the request, or by issuing a ListJobAttributes method requesting status of the print job.

6.6.1.3 400 Bad Request

This status code indicates that the request could not be understood by the server because of malformed syntax. The client should not repeat the request without modifications.

6.6.1.4 401 Unauthorized

The request requires authentication. The response must include a **WWW-Authenticate** header field containing a challenge applicable to the requested action. Normally this would be returned if the end-user was not allowed to print on the specified printer, or tried to cancel or modify a job that did not belong to him or her. If the request already included authorization credentials, then the 401 response indicates that authorization has been refused.

6.6.1.5 404 Forbidden

The server understood the request, but is refusing to perform the request for some unspecified reason. Authorization will not help and the request should not be repeated.

6.6.1.6 404 Not Found

Some resource required to complete this action was not found. This might be the case, for example, when trying to cancel a job which has already been printed, or print on a device that the server does not know about.

6.6.1.7 500 Internal Server Error

The server encountered some unexpected condition when trying to satisfy the request.

6.6.1.8 501 Not Implemented

The server does not support the operation requested.

6.6.1.9 503 Service Unavailable

The server is temporarily unable to handle the request due to some condition such as the specified printer is off-line. The implication is that this is a temporary condition which will be alleviated after some delay.

6.6.1.10 505 HTPP Version not Supported

The server does not support, or refuses to support, the HTTP protocol version that was used in the request message. The server is indicating that it is unable or unwilling to complete the request using the same major version as the client. The response should contain an entity describing why that version is not supported, and what other protocols are supported by the server.

6.6.2 Response Header Fields

Response Header fields are not intended to give information about an entity returned in the response, but rather to give information about the server itself.

```
Response Header =   Location
                  | Server
                  | WWW-Authenticate
                  | Print-ID-on-Server
                  | Print-ID-on-Printer.
```

Location is the URL of the server.

Server provides a text string describing the software on the server that handled this request. For example,

```
Server: IBM Warp Server/4.0
```

WWW-Authenticate tells the client that authentication is required to service this request, and specifies the authentication scheme and parameters required.

Print-ID-On-Server is returned when a new print job is submitted. It provides the job ID assigned by the server, and should be used in subsequent requests that deal with that job.

Print-ID-On-Printer is returned when a print job has been transmitted to the printer. It allows operations, e.g. Cancel, to be performed on jobs that are in the printer.

6.7 Printer Attributes

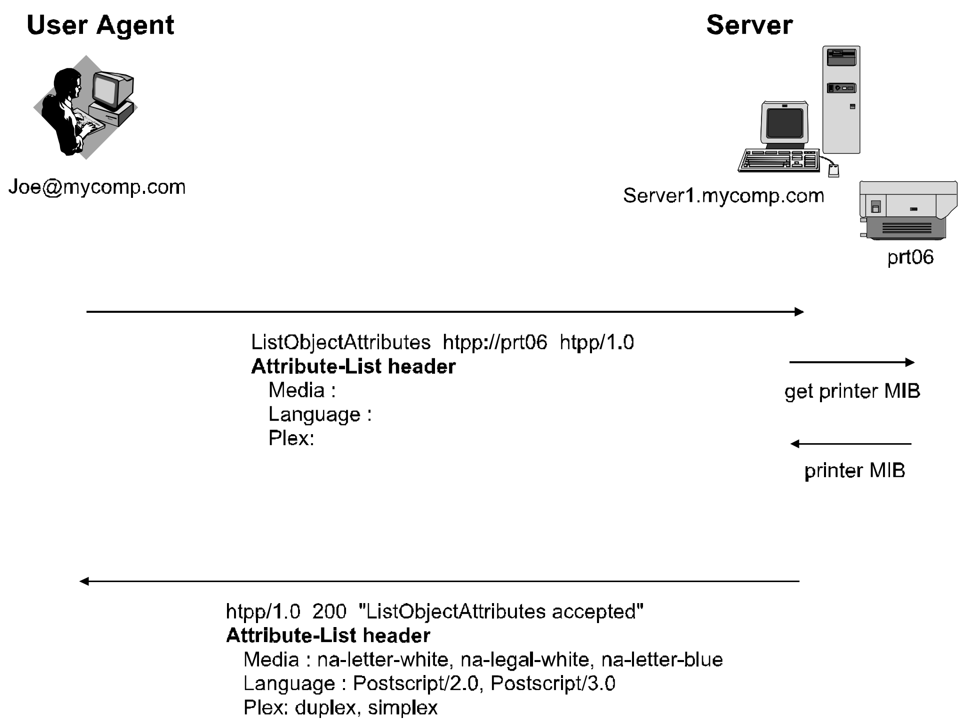
The GetObjectAttributes, when directed to a printer, will return the physical attributes and status of the printer, as defined by elements of the Printer MIB[10]. Printer attributes are returned in a Printer-attribute-list, defined as

Printer-attribute-list = *(Printer-attribute).

Where

Printer-attribute = Printer-Attribute-type : [value] **CRLF.**

Consider the following flow, which would return a specified set of printer attributes:



6.8 References

Note: Internet drafts referenced in this document are to be considered Work in Process.

---

[1] Veizades, J., Guttman, E., Perkins, C., and Kaplan, S., "Service Location Protocol - draft 1.3", Internet-Draft, June 1996

[2]Berners-Lee, T., Fielding, R., and Nielsen, H., "Hypertext Transfer Protocol – HTTP/1.0" 05/17/1996, RFC 1945

[3] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol – HTPP/1.0", Internet-Draft

[4] Fielding, R., Gettys, J., Mogul, J.C., Frystyk, H., and Berners-Lee, T., "Hypertext Transfer Protocol – HTTP/1.1", Internet Working Draft dated August 12, 1996

[5] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, UDEL, August 1982.

[6] Borenstein, N., and Freed, N., "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanism for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, September 1993

[7] ISO/IEC 10175-1 "Distributed Printing Application (DPA) Standard", Final Editor's Draft, January 1995

[8] Hastings, T., "Proposed Specification of Information Objects/Attributes for the Printer Job Monitoring MIB/MIF", Printer Working Group, August 1996

[9] S R. Braden, "Requirements for Internet hosts - application and support", RFC 1123, 10/01/1989.

[10] Smith, R., Wright, F., Hastings, T., Zilles, S., Gyllenskog, J., and Turner, R., "Printer MIB", RFC 1759, March 1995