1.1 IPP Operations Using HTTP

All IPP operations are defined using HTTP as the underlying communication protocol.

1.1.1 HTTP Overview

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

HTTP 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).

HTTP 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.

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 servers must be capable of handling cases where either party closes the connection prematurely, due to user action, automated time out, or program failure.

1.1.2 IPP Operation Encoding

IPP 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 IPP headers conform to the syntax

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

IPP/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**.

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

1.1.2.1 HTTP Request-Header Fields

HTTP 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 and when used it is assumed that IPP would use these headers in a standard way. IPP requests will be completely encapsulated within the entity body of an HTTP request. The HTTP Entity-Header has the form

```
HTTP Entity-Header =     Content-Encoding
                         | Content-Length
                         | Content-Type
                         | extension-header
```

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

Content-Type will always be “Application/IPP”.

1.1.2.2 IPP Request-Line

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

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

Operation token   =     Print | CancelJob | GetAttributes |
                        GetJobs
```

1.1.2.3 IPP Status-Line

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

IPP Status-Line = IPP/1.0 Status-Code Reason-Phrase **CRLF**

1.1.2.4 Print

When an end user submits a job, the client submits a Print Request according to the syntax and semantics of this standard and receives a Print Response according to this standard. The end-user or submitting application selects a Printer which may imply a Job Template.

[Further work needs to done to define the above concept.]

Note that the Printer name is not needed since it is the target of the entire operation. A Print Job contains the information needed by the Print object to print a document or set of documents. When the print operation is invoked, the Entity-Body in the HTTP request includes an IPP Print Job. The concrete syntax of the Print Job is defined in section xxx. The following abstract data types are part of the Print Request.

Job and Document Attributes	A set of Job object and Document attributes as defined in section xxx
Document Contents	Document content is optional and not included when a URL is provided to point to the content.

1.1.2.5 Print Response

The following abstract data types are part of the Print Response:

Job-Identifier	A URL Used for all other operations on this Job.
Job Status	Current-job-state
Printer State	Printer-state
Message	Optional message Note: Is this needed?
Errors	Optional Error Information

1.1.2.6 CancelJob

This operation allows a user to cancel one specific Print Job any time after the print job has been established on the Printer Object. Some pages may be printed before a job is terminated if printing has already started when the Cancel Job operation is received.

The Cancel HTTP request will be sent to the URL identifying the job to be canceled. The following abstract data types are part of the Cancel Job Request.

Message	Optional message to the operator.

1.1.2.7 CancelJob Response

The following abstract data types are part of the Cancel Job Response:

Job Status	Optional Job status information
Errors	Optional Error Information

1.1.2.8 GetAttributes

This operation allows a user to obtain information from the Print object concerning jobs, printers, and print queues, based on ISO 10175. The entity-body of the Get Attributes operation contains the set of attributes that the requester is interested in. However, the attribute values may be null and are ignored by the server. The attribute list 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. The following abstract data types are part of the Get Attributes Request:

Selector	Job-Identifier (URL) or Printer URL
Requested Attributes	A set of attributes in which the requestor is interested

1.1.2.9 GetAttributes Response

The following abstract data types are part of the Get Attributes Response:

Result Attributes	The requested attributes of the object
Errors	Optional error information

1.1.2.10 GetJobs

This operation allows a client to retrieve a list of print jobs belonging to the target Printer object. A list of attributes the client is interested in seeing may be appended to the request. If no attributes are asked for the default set of job-name and total-job-octets is returned for each job. Jobs will be returned in the order in which they are scheduled to print.

The following abstract data types are part of the Get Jobs Request:

Requested Attributes	An optional set of job attributes in which the requestor is interested

1.1.2.11 Get Jobs Response

The following abstract data types are part of the Get Jobs Response:

Result Attributes	Attribute set containing the returned results.
Errors	Optional Error Information

1.1.3 The Print Job

The entity body of a print request will contain a Print Job, as defined below. The headers defined here are IPP headers, but follow the same syntax as the basic HTTP headers.

```
Print Job = Print-Job-Object-Header             section (1.2.1)
            [Job Attributes]                    section (1.2.4)
            *(Documents)

    Job Attribute = Attribute name : Attribute value CRLF

    Document =     Document-Header              section (1.2.2)
                   [Document attributes]        section (1.2.5)
                   [Content-Header              section (1.2.3)
                    content]
```

1.1.3.1 Print Job Object Header

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

Content-Type is always “IPP Print Object”. Other header fields are as defined for HTTP 1.0.

1.1.3.2 Document Header

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

```
Document Header = Content-Encoding
        | Content-Length
        | Content-Type
        | extension-header
```

Content type is always “IPP Document”. Other header fields area as defined in HTTP 1.0.

1.1.3.3 Document-Content Header

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

```
Document-Content-Header =      Content-Encoding
             | Content-Length
             | Content-Type
             | extension-header
```

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
```

Other header fields are as defined by HTTP 1.0.

1.1.3.4 Job Attributes

Job attributes are defined in section xxx. Attributes will always be sent as

```
Job-Attribute = attribute name “:” Attribute value  CRLF

Attribute value = Value | *(Value “,” Value)
```

1.1.3.5 Document Attributes

Document attributes are defined in section yyy. At this point a limited number of attribute may be specified on a document basis. The syntax for a document attribute is

```
Document-Attribute = attribute name “:” Attribute value  CRLF

Attribute value = Value | *(Value “,” Value)
```