Internet Printing Protocol/1.0: MIME Encoding

Status of this Memo

Abstract

This Internet-Draft specifies an Internet Printing Protocol (IPP)that is intended to be version 1.0. This protocol is heavily influence by the semantic operations and attributes defined in ISO/IEC 10175 Document Printing Application (DPA) parts 1 and 3. It also incorporates some of the implementation and interoperability lessons learned from other printing related standards such as POSIX System Administration - Part 4 (POSIX 1378.4) and X/Open A Printing System Interoperability Specification(PSIS).

IPP is defined as a set of abstract data types and operations. The operations are implemented using a simple request and response mechanism built on top of HTTP. The abstract data types are encoded as simple ASCII text strings.

The IPP protocol covers only end user operations on basic print service objects. Authentication is realized by mechanisms outside the scope of the protocol, but the protocol does introduce some access control functionality so that only authorized end users are allowed to submit print jobs to printers whose implementation and site policy support access control. Also, the Cancel Job operation requires some authentication so that jobs can only be canceled by the end user who submitted the job. Extended monitoring and management is possible

through other protocols such as the SNMP Printer MIB. In the areas where there are no existing standards, some proposed and emerging standards are being worked (management, security, etc.). As these services become more stable, this document (and hence the protocol) can be updated to reflect the integration and relationships with these other standards.

Table of Contents

1. Introduction

The Internet Printing Protocol (IPP) is an application level protocol that can be used for distributed printing on the Internet. The protocol is heavily influenced by the printing model introduced in the Document Printing Application (ISO/IEC 10175 DPA) standard, which describes a distributed printing service. DPA identifies the end user and administrative roles associated with a distributed printing service, and defines the set of operations supported by the service. This IPP specification (version 1.0) deals only with the end user role. These ideas and concepts, when unified with other Internet protocols and services, realize a distributed print service for the Internet.

This specification uses the verbs: "shall", "should", "may", and "need not" to specify conformance requirements as follows:

- "shall": indicates an action that the subject of the sentence must implement in order to claim conformance to this specification

- "may": indicates an action that the subject of the sentence does not have to implement in order to claim conformance to this specification, in other words that action is an implementation option

- "need not": indicates an action that the subject of the sentence does not have to implement in order to claim conformance to this specification. The verb "need not" is used instead of "may not", since "may not" sounds like a prohibition.

- "should": indicates an action that is recommended for the subject of the sentence to implement, but is not required, in order to claim conformance to this specification.

2. IPP Operations

This section introduces the IPP operations. Since IPP specifies the use of HTTP as the underlying communication protocol, the mapping of IPP operations on top of HTTP methods is also shown.

2.1 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 shall be capable of handling

cases where either party closes the connection prematurely, due to user action, automated time out, or program failure.

2.2 IPP Operation Encoding

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

```
IPP 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 CRLF as the end-of-line marker for all protocol elements except the entity-body.

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".

IPP messages are contained within HTTP methods. The HTTP POST method is used for the Print operation and the Cancel Job operation. The HTTP GET method is used for the Get Attributes operation and the Get Jobs operation (section 5.4).

2.2.1 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".

2.2.1.1 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 | Cancel-Job |
                      Get-Attributes | Get-Jobs
```

2.2.2 HTTP Response-Header Fields

HTTP response fields allow the server to pass additional information about the response back to the client. IPP will use these headers in a standard way. IPP responses will be completely encapsulated within the entity body of an HTTP response.

2.2.2.1 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
```

2.3 The Print Job

In section 5.4.1, the Print Operation is described. In order to understand that operation better, we first present the notion of a Print Job. The entity body of a print operation 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 (5.3.1)
            [Job-Attributes]          ;section (5.3.4)
            *(Documents)

   Document =     Document-Header       ;section (5.3.2)
                  [Document-attributes] ;section (5.3.5)
                  [Content-Header       ;section (5.3.3)
                    content]
```

2.3.1 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.

2.3.2 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 are as defined in HTTP 1.0.

2.3.3 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 :

```
261         Content-Type = Data-Stream-Format "/" Version
262
```

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

```
265             Content-Type: Postscript/2.0
266
```

Other header fields are as defined by HTTP 1.0.

2.3.4 Job Attributes

Job attributes are defined in section 6.2. Attributes will always be sent as

```
271         Job-Attribute = Attr-name ":" Attr-value     CRLF
272
273         Attr-value = 1#Value
274
```

In the above example, "1#Value" means one or more "," separated values.

2.3.5 Document Attributes

Document attributes are defined in section 6.2.11. The syntax for a document attribute is

```
280         Document-Attribute = Attr-Name ":" Attr-Value    CRLF
281
282         Attr-Value = 1#Value
283
```

In the above example, "1#Value" means one or more "," separated values.

3. Security Considerations

This protocol does not identify any new authentication mechanisms. The authentication mechanisms built into HTTP (such as SSL and SHTTP) are recommended.

This protocol does define a simple authorization mechanism by introducing the "end-user-acl" attribute as part of the Printer object. This ACL attribute is a multi-valued list of all of the authenticated names of end-users. This protocol does not specify what the domain is for names in this ACL attribute.

Issue: Will it always be possible for a Printer to obtain a meaningful authenticated name that the Printer can match against the end-user-acl, or will some other mechanism be necessary, such as a password?

4. References

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

[2] Berners-Lee, T, Fielding, R., and Nielsen, H., "Hypertext Transfer Protocol - HTTP/1.0", RFC 1945, August 1995.

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

[4] Postel, J., "Instructions to RFC Authors", RFC 1543, October 1993.

[5] ISO/IEC 10175 Document Printing Application (DPA), Final, June 1996.

[6] Herriot, R. (editor), X/Open A Printing System Interoperability Specification (PSIS), August 1995.

[7] Kirk, M. (editor), POSIX System Administration - Part 4: Printing Interfaces, POSIX 1387.4 D8, 1994.

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

[9] Braden, S., "Requirements for Internet Hosts - Application and Support", RFC 1123, October, 1989,

[10] McLaughlin, L. III, (editor), "Line Printer Daemon Protocol" RFC 1179, August 1990.

[11] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource Locators (URL)", RFC 1738, December, 1994.

5. Author's Address

6. Appendix A: Sample IPP Operations

The following examples illustrate typical flows using the IPP protocol. In these examples, the IPP Printer object named "printer-1" is located at the node identified by the DNS name "some.domain.com". A Job Template has been defined for printer-1 which establishes the print defaults.

For brevity in the following flows, none of the HTTP headers are shown. CRLF sequences are not shown.

6.1 Querying the printer

```
Client                                    some.domain.com

------------------------------------------------>
Post http://some.domain.com/printer-1   http/1.0
Get-Attributes  IPP/1.0
   printer-state :
   sides-supported :
   media-supported :
   document-formats-supported :

<------------------------------------------------
http/1.0  201 "Created" (a response)
   IPP/1.0 xxx "attribute list returned"
   printer-state : idle
   sides-supported : 1-sided
   media-supported : iso-a4-white, iso-b4-white
   document-formats-supported : Postscript/2.0
```

6.2 Print Operation - with print data included

```
Client                                    some.domain.com

------------------------------------------------>
```

```
   Post http://some.domain.com/printer-1  http/1.0
      Print IPP/1.0
      Print-Job-Object Header
         job-name : My Job
         medium : iso-a4-white
         notification-events : Job-completion
         notification-address : joe@pc.domain.com
      Document Header
         document-name : Letter to Mom
      Document-Content Header (content type = Postscript/2.0)
         <Document in Postscript level 2 format>


   <-------------------------------------------------
   http/1.0  200 "accepted"
      IPP/1.0 xxx "print job accepted and queued"
         job-identifier : some.domain.com/printer-1/0037
         current-job-state : pending
         printer-state : needs-sttention
```

6.3 Print Operation - with no data included

```
   Client                                    some.domain.com

   ------------------------------------------------->
   Post http://some.domain.com/printer-1  http/1.0
      Print IPP/1.0
      Print-Job-Object Header
         job-name : My Job
         medium : iso-a4-white
         notification-events : Job-completion
         notification-address : joe@some.domain.com
      Document Header
         document-name : Letter to Mom
         document-URL : joe@pc.domain.com/Docs/To-mom.ps

   <-------------------------------------------------
   http/1.0  200 "accepted"
      IPP/1.0 xxx "print job accepted and queued"
         job-identifier : some.domain.com/printer-1/0037
         current-job-state : pending
         printer-state : processing
```

6.4 Querying the state of the job

In this example, no attributes are specified, so all job attributes are returned.

```
   Client                                  some.domain.com

   ------------------------------------------------->
   Post http://some.domain.com/printer-1/0037  http/1.0
      Get-Attributes  IPP/1.0
```

```
<-----------------------------------------------
http/1.0 201 "Created" (a response)
   IPP/1.0  xxx "atribute list returned"
   job-Name : My Job
   job-Originator : Joe@some.domain.com
   job-originating-host : pc.domain.com
   notification-address : joe@pc.domain.com
   job-locale : xx:xx:xx
   current-job-status : printing
   submission-time : 1996 Nov 22 1214
   media-sheets-completed : 2
```

6.5 Canceling a Job

```
Client                                  some.domain.com

------------------------------------------------->
Post: http://some.domain.com/printer-1/0037
   Cancel-Job  IPP/1.0



<-----------------------------------------------
http/1.0  200 "okay"
Current-job-state : terminating
```

6.6 Listing jobs on a Printer

List jobs on printer-1, only return job sizes. Jobs are returned in the order they are scheduled for printing. A Job-identifier attribute precedes the attributes returned for each job to delimit job boundaries.

```
Client                                  some.domain.com

------------------------------------------------->
Post http/1.0 some.domain.com/printer-1
   Get-Jobs  IPP/1.0
      total-job-octets :
```

```
   <-------------------------------------------------
   http/1.0  201 "Created" (a response)
      IPP/1.0 xxx "created an attribute list"
      job-identifier : 0033
      total-job-octets : 4567
      job-identifier : 0034
      total-job-octets : 12345
      job-identifier : 0035
      total-job-octets : 12356
```