Rationale for the Structure of the Model and Protocol
for the Internet Printing Protocol

STATUS OF THIS MEMO

This document is an Internet-Draft. Internet-Drafts are working
documents of the Internet Engineering Task Force (IETF), its areas,
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."

To learn the current status of any Internet-Draft, please check the
"1id-abstracts.txt" listing contained in the Internet-Drafts Shadow
directories on ftp.is.co.za (Africa), nic.nordu.net Europe),
munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or
ftp.isi.edu (US West Coast).

ABSTRACT

This document is one of a set of documents, which together describe
all aspects of a new Internet Printing Protocol (IPP). IPP is an
application level protocol that can be used for distributed
printing using Internet tools and technologies. This document describes
IPP from a high level view, defines a roadmap for the various documents
that form the suite of IPP specifications, and gives background and
rationale for the IETF working group's major decisions.

The full set of IPP documents includes:

Design Goals for an Internet Printing Protocol [IPP-REQ]
Rationale for the Structure and Model and Protocol for the
    Internet Printing Protocol (this document)
Internet Printing Protocol/1.0: Model and Semantics [IPP-MOD]
Internet Printing Protocol/1.0: Encoding and Transport [IPP-PRO]
Internet Printing Protocol/1.0: Implementer's Guide [IPP-IIG]
Mapping between LPD and IPP Protocols [IPP-LPD]

The "Design Goals for an Internet Printing
Protocol" document takes a broad look at distributed printing
functionality, and it enumerates real-life scenarios that help to
clarify the features that need to be included in a printing
protocol for the Internet. It identifies requirements for three
types of users: end users, operators, and administrators. The
requirements document calls out a subset of end user requirements
that are satisfied in IPP/1.0. Operator and administrator
requirements are out of scope for version 1.0.

The "Internet Printing Protocol/1.0: Model and Semantics" document
describes a simplified model consisting of abstract objects, their
attributes, and their operations that is independent of encoding and
transport. The model consists of a Printer and a Job object. The Job
optionally supports multiple documents. This document also addresses
security, internationalization, and directory issues.

The "Internet Printing Protocol/1.0: Encoding and Transport" document is
a formal mapping of the abstract operations and attributes defined in
the model document onto HTTP/1.1. It defines the encoding rules for a
new Internet media type called "application/ipp".

The "Internet Printing Protocol/1.0: Implementer's Guide" document gives
insight and advice to implementers of IPP clients and IPP objects. It
is intended to help them understand IPP/1.0 and some of the
considerations that may assist them in the design of their client and/or
IPP object implementations. For example, a typical order of processing
requests is given, including error checking. Motivation for some of the
specification decisions is also included.

The "Mapping between LPD and IPP Protocols" document gives some advice
to implementers of gateways between IPP and LPD (Line Printer Daemon)
implementations.

[1]. ARCHITECTURAL OVERVIEW

The Internet Printing Protocol (IPP) is an application level
protocol that can be used for distributed printing on the Internet.
This protocol defines interactions between a client and a server.
The protocol allows a client to inquire about capabilities of a
printer, to submit print jobs and to inquire about and cancel print
jobs. The server for these requests is the Printer; the Printer is
an abstraction of a generic document output device and/or a print
service provider. Thus, the Printer could be a real printing
device, such as a computer printer or fax output device, or it
could be a service that interfaced with output devices.

The protocol is heavily influenced by the printing model introduced in
the Document Printing Application (DPA) [ISO10175] standard. Although
DPA specifies both end user and administrative features, IPP version 1.0
(IPP/1.0) focuses only on end user functionality.

The architecture for IPP defines (in the Model document [IPP-MOD])
an abstract Model for the data which is used to control the
printing process and to provide information about the process and
the capabilities of the Printer. This abstract Model is
hierarchical in nature and reflects the structure of the Printer
and the Jobs that may be being processed by the Printer.

The Internet provides a channel between the client and the
server/Printer. Use of this channel requires flattening and
sequencing the hierarchical Model data. Therefore, the IPP also
defines (in the Encoding and Transport document [IPP-PRO]) an encoding
of the data in the model for transfer between the client and server.
This transfer of data may be either a request or the response to a
request.

Finally, the IPP defines (in the Encoding and Transport document [IPP-
PRO]) a protocol for transferring the encoded request and response data
between the client and the server/Printer.

An example of a typical interaction would be a request from the
client to create a print job. The client would assemble the Model
data to be associated with that job, such as the name of the job,
the media to use, the number of pages to place on each media
instance, etc. This data would then be encoded according to the
Protocol and would be transmitted according to the Protocol. The
server/Printer would receive the encoded Model data, decode it into

a form understood by the server/Printer and, based on that data, do one of two things: (1) accept the job or (2) reject the job. In either case, the server must construct a response in terms of the Model data, encode that response according to the Protocol and transmit that encoded Model data as the response to the request using the Protocol.

Another part of the IPP architecture is the Directory Schema described in the model document). The role of a Directory Schema is

to provide a standard set of attributes which might be used to
query a directory service for the URI of a Printer that is likely
to meet the needs of the client. The IPP architecture also
addresses security issues such as control of access to
server/Printers and secure transmissions of requests, response and
the data to be printed.

2. THE PRINTER

Because the (abstract) server/Printer encompasses a wide range
of implementations, it is necessary to make some assumptions
about a minimal implementation. The most likely minimal
implementation is one that is embedded in an output device
running a specialized real time operating system and with limited
processing, memory and storage capabilities. This printer will be
connected to the Internet and will have at least a TCP/IP
capability with (likely) SNMP [RFC1905, RFC1906] support for the
Internet connection. In addition, it is likely the the Printer will
be an HTML/HTTP server to allow direct user access to information
about the printer.

3. RATIONALE FOR THE MODEL

The Model [IPP-MOD] is defined independently of any encoding of the
Model data both to support the likely uses of IPP and to be robust
with respect to the possibility of alternate encoding.

It is expected that a client or server/Printer would represent
the Model data in some data structure within the
applications/servers that support IPP. Therefore, the Model was
designed to make that representation straightforward. Typically a
parser or formatter would be used to convert from or to the encoded
data format. Once in an internal form suitable to a product, the
data can be manipulated by the product. For example, the data sent
with a Print Job can be used to control the processing of that
Print Job.

The semantics of IPP are attached to the (abstract) Model.
Therefore, the application/server is not dependent on the encoding
of the Model data, and it is possible to consider alternative
mechanisms and formats by which the data could be transmitted from
a client to a server; for example, a server could have a direct,

client-less GUI interface that might be used to accept some kinds
of Print Jobs. This independence would also allow a different
encoding and/or transmission mechanism to be used if the ones
adopted here were shown to be overly limiting in the future. Such a
change could be migrated into new products as an alternate protocol
stack/parser for the Model data.

Having an abstract Model also allows the Model data to be aligned
with the (abstract) model used in the Printer [RFC1759], Job and

Host Resources MIBs. This provides consistency in interpretation of the data obtained independently of how the data is accessed, whether via IPP or via SNMP [RFC1905, RFC1906] and the Printer/Job MIBs.

There is one aspect of the Model that deserves some extra explanation. There are two ways for identifying a Job object: (a) with a Job URI and (b) using a combination of the Printer URI and a Job ID (a 32 bit positive integer). Allowing Job objects to have URIs allows for flexibility and scalability. For example a job could be moved from a printer with a large backlog to one with a smaller load and the job identification, the Job object URI, need not change. However, many existing printing systems have local models or interface constraints that force Job objects to be identified using only a 32-bit positive integer rather than a URI. This numeric Job ID is only unique within the context of the Printer object to which the create request was originally submitted. In order to allow both types of client access to Jobs (either by Job URI or by numeric Job ID), when the Printer object successfully processes a create request and creates a new Job, the Printer object SHALL generate both a Job URI and a Job ID for the new Job object. This requirement allows all clients to access Printer objects and Job objects independent of any local constraints imposed on the client implementation.

4. RATIONALE FOR THE PROTOCOL

There are two parts to the Protocol: (1) the encoding of the Model data and (2) the mechanism for transmitting the model data between client and server.

4.1 The Encoding

To make it simpler to develop embedded printers, a very simple binary encoding has been chosen. This encoding is adequate to represent the kinds of data that occur within the Model. It has a simple structure consisting of sequences of attributes. Each attribute has a name, prefixed by a name length, and a value. The names are strings constrained to characters from a subset of ASCII. The values are either scalars or a sequence of scalars. Each scalar value has a length specification and a value tag which indicates the type of the value. The value type has two parts: a major class part, such as integer or string, and a minor class part

which distinguishes the usage of the major class, such as dateTime
string. Tagging of the values with type information allows for
introducing new value types at some future time.

A fully encoded request/response has a version number, an operation
(for a request) or a status and optionally a status message (for a
response), associated parameters and attributes which are encoded
Model data and, optionally (for a request), print data following
the Model data.

4.2 **The Transmission Mechanism**

The chosen mechanism for transmitting the encoded Model data is
HTTP 1.1 Post (and associated response). No modifications to HTTP
1.1 **are proposed or required. The sole role of the Transmission**
Mechanism is to provide a transfer of encoded Model data from/to
the client to/from the server. This could be done using any data
delivery mechanism. The key reasons why HTTP 1.1 Post is used are
given below. The most important of these is the first. With perhaps
this exception, these reasons could be satisfied by other
mechanisms. There is no claim that this list uniquely determines a
choice of mechanism.

   1. HTTP 1.0 is already widely deployed and, based on the
   recent evidence, HTTP 1.1 is being widely deployed as the
   manufacturers release new products. The performance benefits
   of HTTP 1.1 have been shown and manufactures are reacting
   positively.

   Wide deployment has meant that many of the problems of making
   a protocol work in a wide range of environments from local net
   to Intranet to Internet have been solved and will stay solved
   with HTTP 1.1 deployment.

   2. HTTP 1.1 solves most of the problems that might have
   required a new protocol to be developed. HTTP 1.1 allows
   persistent connections that make a multi-message protocol be
   more efficient; for example it is practical to have separate
   Create-Job and Send-Document messages. Chunking allows the
   transmission of large print files without having to pre-scan
   the file to determine the file length. The accept headers
   allow the client's protocol and localization desires to be
   transmitted with the IPP operations and data. If the Model
   were to provide for the redirection of Job requests, such as
   Cancel-Job, when a Job is moved, the HTTP redirect response
   allows a client to be informed when a Job he is interested in
   is moved to another server/Printer for any reason.

   3. Most network Printers will be implementing HTTP servers for
   reasons other than IPP. These network attached Printers want
   to provide information on how to use the printer, its current
   state, HELP information, etc. in HTML. This requires having an
   HTTP server which would be available to do IPP functions as

well.

4. Most of the complexity of HTTP 1.1 is concerned with the
implementation of HTTP proxies and not the implementation of
HTTP clients and/or servers. Work is proceeding in the HTTP
Working Group to help identify what must be done by a server.
As the Encoding and Transport document shows, that is not
very much.

   5. HTTP implementations provide support for handling URLs that
   would have to be provided if a new protocol were defined.

   6. An HTTP based solution fits well with the Internet security
   mechanisms that are currently deployed or being deployed. HTTP
   will run over SSL3. The digest access authentication mechanism
   of HTTP 1.1 provides an adequate level of access control. These
   solutions are deployed and in practical use; a new solution would
   require extensive use to have the same degree of confidence in its
   security. Note: SSL3 is not on the IETF standards track.

   7. HTTP provides an extensibility model that a new protocol
   would have to develop independently. In particular, the
   headers, intent-types (via Internet Media Types) and error
   codes have wide acceptance and a useful set of definitions and
   methods for extension.

   8. Although not strictly a reason why IPP should use HTTP as
   the transmission protocol, it is extremely helpful that there
   are many prototyping tools that work with HTTP and that CGI
   scripts can be used to test and debug parts of the protocol.

   9. Finally, the POST method was chosen to carry the print data
   because its usage for data transmission has been established,
   it works and the results are available via CGI scripts or
   servlets. Creating a new method would have better identified
   the intended use of the POSTed data, but a new method would be
   more difficult to deploy. Assigning a new default port for
   IPP provided the necessary identification with minimal impact
   to installed infrastructure, so was chosen instead.

5. RATIONALE FOR THE DIRECTORY SCHEMA

Successful use of IPP depends on the client finding a suitable IPP
enabled Printer to which to send a IPP requests, such as print a
job. This task is simplified if there is a Directory Service which
can be queried for a suitable Printer. The purpose of the Directory
Schema is to have a standard description of Printer attributes that
can be associated the URI for the printer. These attributes are a
subset of the Model attributes and can be encoded in the
appropriate query syntax for the Directory Service being used by
the client.

[6](). RATIONALE FOR SECURITY

Security is an areas of active work on the Internet. Complete
solutions to a wide range of security concerns are not yet
available. Therefore, in the design of IPP, the focus has been on
identifying a set of security protocols/features that are
implemented (or currently implementable) and solve real problems
with distributed printing. The two areas that seem appropriate to

support are: (1) authorization to use a Printer and (2) secure interaction with a printer. The chosen mechanisms are the digest authentication mechanism of HTTP 1.1 and SSL3 [SSL] secure communication mechanism.

7. COPYRIGHT

8. REFERENCES

[IPP-IIG]
Hastings, T., Manros, C., "Internet Printing Protocol/1.0: Implementer's Guide", draft-ietf-ipp-implementors-guide-00.txt, November 1998, work in progress.

[IPP LPD]
Herriot, R., Hastings, T., Jacobs, N., Martin, J., "Mapping between LPD and IPP Protocols", draft-ietf-ipp-lpd-ipp-map-05.txt, November 1998.

[IPP-MOD]
deBry, R., Isaacson, S., Hastings, T., Herriot, R., Powell,
P. "Internet Printing Protocol/1.0: Model and Semantics"
draft-ietf-ipp-mod-11.txt, November, 1998.

[IPP-PRO]
Herriot, R., Butler, S., Moore, P., Tuner, R., "Internet Printing
Protocol/1.0: Encoding and Transport", draft-ietf-ipp-pro-07.txt,

November, 1998.

[IPP-REQ]
Wright, D., "Design Goals for an Internet Printing Protocol",
draft-ietf-ipp-req-03.txt, November, 1998.

[ISO10175]
ISO/IEC 10175 "Document Printing Application (DPA)", June 1996

[RFC1759]
Smith, R., Wright, F., Hastings, T., Zilles, S., and Gyllenskog,
J., "Printer MIB", RFC 1759, March 1995.

[RFC1905]
J. **Case, et al. "Protocol Operations for Version 2 of the Simple**
Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[RFC1906]
J. **Case, et al. "Transport Mappings for Version 2 of the Simple**
Network Management Protocol (SNMPv2)", RFC 1906, January 1996.

[SSL]
Netscape, The SSL Protocol, Version 3, (Text version 3.02), November
1996.

9. AUTHORS ADDRESS

Stephen Zilles
Adobe Systems Incorporated
345 Park Avenue
MailStop W14
San Jose, CA 95110-2704

Phone: +1 408 536-4766
Fax: +1 408 537-4042
Email: szilles@adobe.com

Expires May 16, 1999