Robert Herriot (editor)
Sun Microsystems
Sylvan Butler
Hewlett-Packard
Paul Moore
Microsoft.
Randy Turner
Sharp Labs
July 2, 1997

# Internet Printing Protocol/1.0: Protocol Specification
draft-ietf-ipp-protocol-00.txt

## Status of this Memo



## 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 technology. The protocol is heavily influenced by the printing model introduced in the Document Printing Application (ISO/IEC 10175 DPA) standard. Although DPA specifies both end user and administrative features, IPP version 1.0 is focused only on end user functionality.

The full set of IPP documents includes:

Internet Printing Protocol: Requirements
Internet Printing Protocol/1.0: Model and Semantics
Internet Printing Protocol/1.0: Security
Internet Printing Protocol/1.0: Protocol Specification
Internet Printing Protocol/1.0: Directory Schema

The requirements 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 MUST be satisfied in the first version of IPP. Operator and administrator requirements are out of scope for v1.0. The model and semantics document describes a simplified model with abstract objects, their attributes, and their operations. The model introduces a Printer object and a Job object. The Job object supports multiple documents per job. The security document covers potential threats and proposed counters to those threats. The protocol specification is formal document which incorporates the ideas in all the other documents into a concrete mapping using clearly defined data representations and transport protocol mappings that real implementers can use to develop interoperable client and server side components. Finally, the directory schema document shows a generic schema for directory service entries that represent instances of IPP Printers.

This document is the "Internet Printing Protocol/1.0: Protocol Specification" document.

Table of Contents

1. Introduction ... 3
2. Conformance Terminology ... 3
3. Encoding of the Operation Layer ... 3
3.1 Syntax of Encoding ... 3
3.2 Diagram of Encoding ... 4
3.3 Version ... 6
3.4 Mapping of Operations ... 6
3.5 Mapping of Status ... 7
3.6 Name-Lengths ... 7
3.7 Mapping of Parameter Names ... 8
3.8 Value Lengths ... 10
3.9 Mapping of Attribute and Parameter Values ... 10
3.10 Data ... 11
4. Encoding of Transport Layer ... 11
4.1 General Headers ... 12
4.2 Request Headers ... 12
4.3 Response Headers ... 12
4.4 Entity Headers ... 13
5. Security Considerations ... 13
6. Appendix A: Requirements without HTTP/1.1 as a Transport Layer ... 13
6.1 Additional Parameter-group for HTTP header information ... 14
6.2 Chunking of Data ... 14
6.3 Revised Syntax for the Operation Layer ... 15
7. References ... 15
8. Author's Address ... 16
9. Other Participants: ... 17

# 1. Introduction

This document contains the rules for encoding IPP operations and describes two layers: the transport layer and the operation layer.

The transport layer consists of an HTTP/1.1 request or response. RFC 2068 [27] describes HTTP/1.1. This document specifies the HTTP headers that an IPP implementation supports.

The operation layer consists of a message body in an HTTP request or response. The document "Internet Printing Protocol/1.0: Model and Semantics" [21] defines the semantics of such a message body and the supported values. This document specifies the encoding of an IPP operation. The aforementioned document is henceforth referred to as the “IPP model document”

# 2. Conformance Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [25].

# 3. Encoding of the Operation Layer

The operation layer SHALL contain a single operation request or operation response.

The encoding is defined using both a diagram and Augmented Backus-Naur Form (ABNF), as specified by draft-ietf-drums-abnf-02.txt [29] except that ‘strings of literals’ SHALL be case sensitive. For example ‘A’ means upper-case ‘A’ and not lower case ‘a’. In addition, two-byte binary signed integer fields are represented as ‘%x’ values which show their range of values.

All binary integers in this encoding SHALL be transmitted in big-endian format (also known as “network order” and “most significant byte first”)

## 3.1 Syntax of Encoding

The encoding consists of four parts:

- version
- operation (for a request) or status (for a response)
- parameters
- data (which is optional)

The ‘parameters’ consists of a sequence of one or more ‘parameter’s. Each ‘parameter’ consists of:

- name-length: a two byte binary integer which is the length of the following ‘name’
- name
- value-length: a two byte binary integer which is the length of the following ‘value’
- value

There are 3 values for name-length with special meanings that provide for structuring of ‘parameter’s.

- 0 (0x0000): denotes an additional value for the preceding parameter. The following name is empty. That is, the ‘value-length’ starts in the next octet. A ‘parameter’ with this value consists of only a ‘name-length’, ‘value-length’ and ‘value’.
- -1 (0xFFFF): denotes the end of the ‘parameters’. That is, the ‘data’ starts in the next octet. These 2 octets are present even when there is no ‘data’. A ‘parameter’ with this value consists of only a ‘name-length’.
- -2 (0xFFFE): denotes the end of a ‘parameter-group’. That is, the next ‘parameter-group’ starts in the next octet. This special value exists only when an operation contains 2 or more ‘parameter-groups’. Parameters are grouped to separate operation parameters from object attributes and to separate attributes of one object from those of another object. A ‘parameter-group’ is defined in greater detail in section 3.7 “Mapping of Parameter Names”. A ‘parameter’ with this value consists of only a ‘name-length’.

The syntax for the operation layer below shows the structure created by the 3 special ‘name-length’ values described above.

```
ipp-message = ipp-request / ipp-response
ipp-request = version operation parameters [ data ]
ipp-response = version status parameters [ data ]
version = major-version minor-version
major-version = SIGNED-BYTE  ; initially %d1
minor-version = SIGNED-BYTE  ; initially %d0
operation = SIGNED-SHORT  ; mapping from model defined below
status = SIGNED-SHORT     ; mapping from model defined below
parameters = parameter-group *(END-PARAMETER_GROUP parameter-group) END-PARAMETERS
parameter-group = *parameter-set
parameter-set = parameter *(more-values)
parameter = name-length name value-length value
more-values = ZERO-NAME-LENGTH value-length value
name-length = SIGNED-SHORT    ; number of octets of ‘name’
name = LALPHA *( LALPHA / DIGIT / “-” / “_” )
value-length = SIGNED-SHORT  ; number of octets of ‘value’
value = OCTET-STRING
data = OCTET-STRING

ZERO-NAME-LENGTH = %x00 %x00           ; name-length of 0
END-PARAMETERS = %xFF %xFF             ; name-length of -1
END-PARAMETER_GROUP =  %xFF %xFE  ; name-length of -2
SIGNED-BYTE = %x0..%xFF
SIGNED-SHORT = %x0..%xFF %x0..%xFF
DIGIT = "0".."9"
LALPHA = "a".."z"
BYTE = %x0..%xFF
OCTET-STRING = *BYTE
```

ISSUE: should there be a “type-of-value” byte so that a parser can decode a value without looking up the attribute name? If there were such a byte, it could also server as the flag byte for the negative values, solving two problems with one byte per parameter.

## 3.2 Diagram of Encoding

The following is a diagram of the encoding of a request operation.

Note: 1 or more ‘parameter groups’ may be omitted, and ‘data’ may be omitted.

```
 -----------------------------------------------
 |   major version      |    minor version     |     2 bytes
 -----------------------------------------------
 |                  operation                  |     2 bytes
 -----------------------------------------------
 |               parameter-group               |     k bytes
 --------------------------------------------------------------
 |                   0xFFFE                    |     2 bytes  |
 -----------------------------------------------              |
 |                     …                       |     m bytes  |
 -----------------------------------------------              |
 |               parameter-group               |     n bytes  |- optional
 -----------------------------------------------              |
 |                   0xFFFE                    |     2 bytes  |
 -----------------------------------------------              |
 |               parameter-group               |     p bytes  |
 --------------------------------------------------------------
 |                   0xFFFF                    |     2 bytes
 -----------------------------------------------
 |                    data                     |     q bytes - optional
 -----------------------------------------------
```

The following is a diagram of the ‘parameter-group’ which is 0 or more ‘parameter’s

```
 -----------------------------------------------
 |                  parameter                  |     r bytes
 -----------------------------------------------
 |                     …                       |     s bytes
 -----------------------------------------------
 |                  parameter                  |     t bytes
 -----------------------------------------------
```

The following is a diagram of a ‘parameter’. The optional fields are present only when a ‘parameter’ has more than one value.

```
-----------------------------------------------
|              length-of-name                 |    2 bytes
-----------------------------------------------
|              parameter-name                 |    u bytes
-----------------------------------------------
|              length-of-value                |    2 bytes
-----------------------------------------------
|              parameter-value                |    v bytes
-------------------------------------------------------------
|                  0x0000                     |    2 bytes |
-----------------------------------------------            |
|              length-of-value                |    2 bytes |
-----------------------------------------------            |
|              parameter-value                |    w bytes |- optional
-----------------------------------------------            |
|                    …                        |    x bytes |
-----------------------------------------------            |
|                  0x0000                     |    2 bytes |
-----------------------------------------------            |
|              length-of-value                |    2 bytes |
-----------------------------------------------            |
|              parameter-value                |    y bytes |
-------------------------------------------------------------
```

The encoding of a response operation is identical to the encoding of a request operation except that the ‘status’ field SHALL replace the ‘operation’ field.

## 3.3 Version

The version SHALL consist of a major and minor version, each of which SHALL be represented by a one byte signed integer. The protocol described in this document SHALL have a major version of 1 (0x01) and a minor version of 0 (0x00).

## 3.4 Mapping of Operations

The following SHALL be the mapping of operations names to integer values which are encoded as two byte binary signed integers. The operations are defined in the IPP model document

| **Operation** | **Integer Encoding (in decimal)** |
|---|---|
| Get-Operations | 1 |
| Print-Job | 2 |
| Print-URI | 3 |
| Validate-Job | 4 |
| Create-Job | 5 |
| Send-Document | 6 |
| Send-URI | 7 |
| Cancel-Job | 8 |
| Get-Attributes | 9 |
| Get-Jobs | 10 |

## 3.5 Mapping of Status

The following SHALL be the mapping of status names to integer values which are encoded as two byte binary signed integers. The status names are defined in the IPP model document.

If an IPP status is returned, then the HTTP status MUST be 200 (OK). With any other HTTP status value, the HTTP response SHALL NOT contain an IPP message-body, and thus no IPP status is returned.

Note: the integer encodings below were chosen to be similar to corresponding status values in HTTP. But the status returned at the HTTP level will always be different except in the case where ‘OK’ is returned at both levels, 200 (OK) in HTTP and 0 (successful-OK) in IPP.

Note: some status values, such as client-error-unauthorized, may  be returned at the transport (HTTP) level rather than the operation level.

ISSUE: as implementations proceed, we will learn what error code need to be supported at the IPP level.

| **Status Name** | **Integer Encoding (in decimal)** |
|---|---|
| successful-OK | 0 |
| client-error-bad-request | 400 |
| client-error-unauthorized | 401 |
| client-error-payment-required | 402 |
| client-error-forbidden | 403 |
| client-error-not-found | 404 |
| client-error-method-not-allowed | 405 |
| client-error-timeout | 408 |
| client-error-gone | 410 |
| client-error-request-entity-too-large | 413 |
| client-error-request-URI-too-long | 414 |
| client-error-unsupported-document-format | 415 |
| client-error-attribute-not-supported | 416 |
| server-error-internal-server-error | 500 |
| server-error-operation-not-implemented | 501 |
| server-error-service-unavailable | 503 |
| server-error-timeout | 504 |
| server-error-version-not-supported | 505 |
| server-error-printer-error | 506 |
| server-error-temporary-error | 507 |

## 3.6 Name-Lengths

Each parameter ‘name’ SHALL be preceded by a two byte binary signed integer in big endian order which SHALL specify the number of octets in the ‘name’ which follows this length, exclusive of the two bytes specifying the length.

The ‘name-length’ field SHALL consist of a two byte binary signed integer in big endian order. This field SHALL specify the number of octets in the ‘name’ which follows the name-length field, excluding the two bytes of the ‘name-length’ field.

If a ‘name-length’ field has a value of zero, the following ‘name’ field SHALL be empty, and the following value SHALL be treated as an additional value for the preceding parameter.

If a ‘name-length’ field as a negative value, it SHALL act as a separator field and the meaning of the following octets SHALL be as specified by the syntax above.

## 3.7 Mapping of Parameter Names

Some parameters are encoded in a special position. These parameters are:

- "target URI": The target URI of each operation in the IPP model document SHALL be specified outside of the operation layer and SHALL not be specified within the operation layer.
- "document-content": The parameter named "document-content" in the IPP model document SHALL become the "data" in the operation layer.
- "status": The parameter named "status" in the IPP model document SHALL become the "status" field in the operation layer response. A response may optionally include the parameter "reason-phase" to specify human-readable text associated with the status. A client MAY display such text to an end-user and a server MAY return such text in the language requested by the client. If this parameter is in the response, it SHALL be the first one in the first parameter-group.

**ISSUE:** Should the target-URI be outside the operation layer?

The remaining parameters are encoded in one of the parameter-groups. The first parameter group is for actual operation parameters and subsequent parameter-groups are for object attributes. Of the parameters defined in the IPP model document, some represent an actual operation parameters and others represent a collection of object attributes.

A parameter in the IPP model document SHALL represent a collection of object attributes if it contains the word "attributes" in its name with no hyphen before "attributes" ; otherwise it SHALL represent an actual operation parameter.

ISSUE: need to align a few names in the IPP model document to make this rule be correct.

If a parameter in IPP model document represents an actual operation parameter and is not in a special position, it SHALL be encoded in the first parameter-group using the text name of the parameter specified in the IPP model document.

If a parameter in IPP model document represent a collection of object attributes, the attributes SHALL be encoded in the second or subsequent parameter-groups using the text names of the attributes specified in the IPP model document. The IPP model document specifies the members of such attribute collections, but does not require that all members of a collection be present in an operation.

If an operation contain attributes from exactly one object, there SHALL be a second parameter-group, but no additional ones. If an operation contains attributes from more than one object (e.g. Get-Jobs response), the attributes from each object SHALL be in a separate parameter-group, such that the attributes from the first object are in the second 'parameter-group', the attributes from the second object are in the third 'parameter-group', etc.

The next three tables show the results of applying the rules above to the operations defined in the IPP model document.

The following table shows the mapping of all request parameters (except target URI) to a parameter-group or special position in the protocol, as described in the IPP model document.

| **Operation** | **first parameter-group** | **second parameter-group** | **special position** |
|---|---|---|---|
| Get-Operations | | | |
| Print-Job | | job-template attributes | document-content |
| Validate-Job or Create-Job | | job-template attributes | |
| Print-URI | document-uri | job-template attributes | |
| Send-Document | last-document | document attributes | document-content |

| Operation | first parameter-group | second parameter-group | special position |
|---|---|---|---|
| Send-URI | last-document<br>document-uri | document attributes | |
| Cancel -Job | message<br>reply-with-status (?) | | |
| Get-Attributes | document-format<br>requested-attributes | | |
| Get-Jobs | limit<br>requested-attributes | | |

The following table shows the mapping of all response parameters to a parameter-group or special position in the protocol, as described in the IPP model document.

| Operation | first parameter-group | second parameter-group | special position |
|---|---|---|---|
| Get-Operations | supported-operations | reason-phrase | status |
| Print-Job, Print-URI or Create-Job | job-uri | reason-phrase<br>job-status attributes | status |
| Send-Document or Send-URI | | reason-phrase<br>job-status attributes | status |
| Validate-Job | | reason-phrase | status |
| Cancel-Job | | reason-phrase<br>job-status attribute ? | status |
| Get-Attributes | | reason-phrase<br>requested attributes | status |
| Get-Jobs | | reason-phrase<br>requested attributes<br>(see the Note below) | status |

Note for Get-Jobs: there is a separate ‘parameter-group’ containing requested-attributes for each job object in the response

The following table shows the mapping of all error response parameters to a parameter-group or special position in the protocol, as described in the IPP model document. Those operations omitted don’t have special parameters for an error return.

| Operation | first parameter-group | second parameter-group | special position |
|---|---|---|---|
| Print-Job, Print-URI, Validate-Job, Create-Job, Send-Document or Send-URI | | unsupported attributes | status |

### 3.8 Value Lengths

Each parameter value SHALL be preceded by a two byte binary signed integer in big endian order which SHALL specify the number of octets in the value which follows this length, exclusive of the two bytes specifying the length.

### 3.9 Mapping of Attribute and Parameter Values

The following SHALL be the mapping of attribute and parameter values to their IPP encoding. The syntax types are defined in the IPP model document.

| Syntax of Attribute Value | Encoding |
|---|---|
| text | an octet string where each character is a member of the UCS-2 coded character set and is encoded using UTF-8. The text is encoded in “network byte order” with the first character in the text (according to reading order) being the the first character in the encoding. |
| name | same as text |
| fileName | same as text |
| keyword | same as text. Allowed text values are defined in the IPP model document |
| uri | same as text |
| uriScheme | same as text |
| boolean | one binary octet where 0x00 is ‘false’ and 0x01 is ‘true’ |
| integer | number of octets is a power of 2 (i.e. 1, 2 or 4). These octets represent a signed binary integer in big endian order (also known as “network byte order” and MSB first). |
| enum | same as integer. Allowed integer values are defined in the IPP model document |
| dateTime | same as text. Syntax of data and time is defined by ISO 8601<br>ISSUE: should ISO 8601 be called out in the IPP model document? |
| seconds | same as integer |
| milliseconds | same as integer |
| 1setOf X | encoding according to the rules for a parameter with more than more value. Each value X is encoded according to the rules for encoding its type. |
| rangeOf X | same 1setOf X where the number of values is 2. |

There is sometimes a need for a parameter to have some special ‘out-of-band’ values. Such value are represented by empty values with special negative lengths as specified by the table below.

| Special Value | Value of Value-length |
|---|---|
| default | -1 |
| unsupported | -2 |

If a response contains a parameter of “unsupported attribute”, the value of “unsupported” shall be used to denote that the Printer does not support the attribute.

ISSUE: The above sentence belongs in the IPP model document.

### 3.10 Data

.The dataSHALL be included within the operation as is. For print operations where the data is document data, this data consists of the identical octet-string that the client specified to print.

NOTE: In HTTP, the data may be encoded as part of the encoding of the entire message-body.

## 4. Encoding of Transport Layer

HTTP/1.1 shall be the transport layer for this protocol.

The operation layer has been designed with the assumption that the transport layer contains the following information:

- the URI of the target job or printer operation
- the total length of the data in the operation layer, either as a single length or as a sequence of chunks each with a length.
- the client's language, the character-set and the transport encoding.

Each HTTP operation shall use the POST method where the URI is the object target of the operation, and where the "Content-Type" of the message-body in each request and response shall be "application/ipp". The message-body shall contain the operation layer and shall have the syntax described in section 2 "Conformance Terminology".

ISSUE: Should the URI of the operation be in the operation layer? Should the URI in the POST be a server/printer always with the target object as a parameter within the message-body?

In the following sections, there are a tables of all HTTP headers which describe their use in an IPP client or server. The following is an explanation of each column in these tables.

- the "header" column contains the name of a header
- the "request/client" column indicates whether a client sends the header.
- the "request/server" column indicates whether a server supports the header when received.
- the "response/server" column indicates whether a server sends the header.
- the "response /client" column indicates whether a client supports the header when received.
- the "values and conditions" column specifies the allowed header values and the conditions for the header to be present in a request/response.

The table for "request headers" does not have columns for responses, and the table for "response headers" does not have columns for requests.

The following is an explanation of the values in the "request/client" and "response/server" columns.

- man: (mandatory) the client or server MUST send the header,
- c-man: (conditionally mandatory) the client or server MUST send the header when the condition described in the "values and conditions" column is met,
- opt: (optional) the client or server MAY send the header
- not: (not needed) the client or server SHOULD NOT send the header. It is not relevant to an IPP implementation.

The following is an explanation of the values in the "response/client" and "request/server" columns.

- man: (mandatory) the client or server MUST support the header,
- opt: (optional) the client or server MAY support the header
- not: (not needed) the client or server SHOULD NOT support the header. It is not relevant to an IPP implementation.

### 4.1 General Headers

The following is a table for the general headers.

ISSUE: an HTTP expert should review these tables for accuracy.

| **General-Header** | **Request** | | **Response** | | **Values and Conditions** |
|---|---|---|---|---|---|
| | **Client** | **Server** | **Server** | **Client** | |
| Cache-Control | man | not | man | not | "no-cache" only |
| Connection | c-man | man | c-man | man | "close" only. Header MUST be present only for last request/response before the connection is closed. |
| Date | opt | opt | man | opt | per RFC 1123 [9] |
| Pragma` | man | not | man | not | "no-cache" only |
| Transfer-Encoding | c-man | man | c-man | man | "chunked" only . Header MUST be present if Content-Length is absent. |
| Upgrade | not | not | not | not | |
| Via | not | not | not | not | |

### 4.2 Request Headers

The following is a table for the request headers.

| **Request-Header** | **Client** | **Server** | **Request Values and Conditions** |
|---|---|---|---|
| Accept | opt | man | "application/ipp" only. This value is the default if the client omits it |
| Accept-Charset | opt | man | per IANA Character Set registry. ISSUE: is this useful for IPP? |
| Accept-Encoding | opt | man | empty and per RFC 2068 [27] and IANA registry for content-codings |
| Accept-Language | opt | man | see RFC 1766 [26] |
| Authorization | c-man | man | per RFC 2068. A client sends this header when it receives a 401 "Unauthorized" response and no "Proxy-Authenticate" header. |
| From | not | not | per RFC 2068. Because RFC recommends sending this header only with the user's approval, it is not very useful |
| Host | man | man | per RFC 2068 |
| If-Match | not | not | |
| If-Modified-Since | not | not | |
| If-None-Match | not | not | |
| If-Range | not | not | |
| If-Unmodified-Since | not | not | |
| Max-Forwards | not | not | |
| Proxy-Authorization | c-man | not | per RFC 2068. A client MUST send this header when it receives a 401 "Unauthorized" response and a "Proxy-Authenticate" header. |
| Range | not | not | |
| Referer | not | not | |
| User-Agent | not | not | |

### 4.3 Response Headers

The following is a table for the request headers.

| **Response-Header** | **Server** | **Client** | **Response Values and Conditions** |
|---|---|---|---|
| Accept-Ranges | not | not | |

| Response-Header | Server | Client | Response Values and Conditions |
|---|---|---|---|
| Age | not | not | |
| Location | c-man | opt | per RFC 2068. When URI needs redirection. |
| Proxy-Authenticate | not | man | per RFC 2068 |
| Public | opt | opt | per RFC 2068 |
| Retry-After | opt | opt | per RFC 2068 |
| Server | not | not | |
| Vary | not | not | |
| Warning | opt | opt | per RFC 2068 |
| WWW-Authenticate | c-man | man | per RFC 2068. When a server needs to authenticate a client. |

### 4.4 Entity Headers

The following is a table for the request headers.

| Entity-Header | Request | | Response | | Values and Conditions |
|---|---|---|---|---|---|
| | Client | Server | Server | Client | |
| Allow | not | not | not | not | |
| Content-Base | not | not | not | not | |
| Content-Encoding | opt | man | man | man | per RFC 2068 and IANA registry for content codings. |
| Content-Language | opt | man | man | man | see RFC 1766 [26]. |
| Content-Length | c-man | man | c-man | man | the length of the message-body per RFC 2068. Header MUST be present if Transfer-Encoding is absent.. |
| Content-Location | not | not | not | not | |
| Content-MD5 | opt | opt | opt | opt | per RFC 2068 |
| Content-Range | not | not | not | not | |
| Content-Type | man | man | man | man | “application/ipp” only |
| ETag | not | not | not | not | |
| Expires | not | not | not | not | |
| Last-Modified | not | not | not | not | |

## 5. Security Considerations

When utilizing HTTP 1.1 as a transport for IPP, all of the security considerations specified in RFC 2068 [27] apply. In addition, the IPP adds some additional application-specific security considerations, including denial-of-service attacks, mutual authentication, and privacy. The IPP model document addresses IPP-specific security considerations, while RFC 2068 addresses HTTP-related security considerations.

ISSUE: the security subgroup is free to add whatever is necessary to fill out the "security considerations" section of this document. However, the IPP model document should include the bulk of security discussions that are IPP-specific.

## 6. Appendix A: Requirements without HTTP/1.1 as a Transport Layer

The operation layer defined above assumed certain services would be provided by the HTTP transport layer. Without that layer, information, such as length, URI and Content-Encoding are absent.

This section specifies the modifications to the operation-layer encoding for raw TCP/IP. The following changes would have to made. All of these changes are upward compatible with the encoding defined in section 3 “Encoding of the Operation Layer”.

## 6.1 Additional Parameter-group for HTTP header information

First there is an additional header parameter-group which SHALL appear as the first parameter-group, preceding the parameter-group for actual operation parameters, and which SHALL contain the “target- URI” along with relevant HTTP header information, including those shown below. This header parameter-group SHALL be preceded by a name-length of -4 (0xFFFC) which functions like the other negative lengths defined in section 3 “Encoding of the Operation Layer”. This special number specifies that the first parameter group contains header type information, and distinguishes it from the protocol which has these parameters outside of the operation layer.

The following table shows the mapping of HTTP headers to parameters in the operation layer.

| **HTTP header or other concept** | **IPP parameter name** | **Syntax Type of Parameter** |
|---|---|---|
| URI | target-URI | uri |
| Connection | Close-Connection | Boolean |
| Accept-Charset | Accept-Charset | keyword |
| Accept-Encoding | Accept-Encoding | keyword |
| Accept-Language | Accept-Language | keyword |
| Content-Encoding | Content-Encoding | keyword |
| Content-Language | Content-Language | keyword |
| charset parameter | Content-Charset | keyword |
| miscellaneous security | if needed at this level | |

The first parameter in the header parameter-group for a request SHALL be the attribute “target-URI” which is the target object of the operation.

Except for Content-Encoding, the parameters SHALL take effect at the beginning of the next parameter-group and apply to the rest of the operation. If the parameter is Content-Encoding, then the encoding SHALL apply only to the ‘full-data’ or ‘data-segment’s as defined by the syntax below and the resulting decoded data SHALL have the syntax of “parameters data” as defined by the syntax below. That is, from a decoding point of view if Content-Encoding is specified, the operation’s data is decoded using the algorithm specified by Content-Encoding. The resulting octet stream is parsed as if it were a ‘parameters’ followed by ‘data’.

NOTE: This rule for Content-Encoding allows a client or server to encode operation parameters or to transmit them unencoded.

ISSUE: should the reason-phrase be in the initial parameter group instead of the second one for responses?

## 6.2 Chunking of Data

Second the “parameters” and “data” of the operation layer are separated by -3 (0xFFFD) instead of -1 (0xFFFF) to denote that the following data is chunked. A chunk of length 0 denotes the end of the data. The syntax for the chunked data is:

```
chunked-data = *data-chunk END-OF-DATA
data-chunk = data-segment-length data-segment
data-segment-length = SIGNED-SHORT   ; number of octets of the data in binary
data-segment = OCTET-STRING

END-OF-DATA = %x00 %x00
```

A data-segment contains fragments of the data. When all the data-segments are concatenated together, they form the complete data.

### 6.3 Revised Syntax for the Operation Layer

The following is the revised syntax for the operation layer.

```
ipp-message = ipp-request / ipp-response
ipp-request = version operation parameters data
ipp-response = version status parameters data
version = major-version minor-version
major-version = SIGNED-BYTE  ; initially %d1
minor-version = SIGNED-BYTE  ; initially %d0
operation = SIGNED-SHORT  ; mapping from model defined below
status = SIGNED-SHORT     ; mapping from model defined below
parameters = (headers / parameter-group ) *( END-PARAMETER_GROUP parameter-group)
headers = START_HEADER  parameter-group
parameter-group = *parameter
parameter = single-parameter *(more-values)
single-parameter = name-length name value-length value
more-values = ZERO-NAME-LENGTH value-length value
name-length = SIGNED-SHORT    ; number of octets of ‘name’
name = LALPHA *( LALPHA / DIGIT / “-” / “_” )
value-length = SIGNED-SHORT  ; number of octets of ‘value’
value = OCTET-STRING
data = (END-PARAMETERS[ full-data ] ) / (END-PARAMETERS-CHUNKED chunked-data )
full-data = OCTET-STRING
chunked-data = *data-chunk %x00 %x00
data-chunk = data-segment-length data-segment
data-segment-length = SIGNED-SHORT   ; number of octets of the data in binary
data-segment = OCTET-STRING

ZERO-NAME-LENGTH = %x00 %x00           ; name-length of 0
END-PARAMETERS = %xFF %xFF             ; name-length of -1
END-PARAMETER_GROUP =  %xFF %xFE  ; name-length of -2
END-PARAMETERS-CHUNKED = %xFF %xFD       ; name-length of -3
START_HEADER = %xFF %xFC       ; name-length of -4
SIGNED-BYTE = %x0..%xFF
SIGNED-SHORT = %x0..%xFF %x0..%xFF
DIGIT = "0".."9"
LALPHA = "a".."z"
BYTE = %x0..%xFF
OCTET-STRING = *BYTE
```

## 7. 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), 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.

[20] Internet Printing Protocol: Requirements

[21] Internet Printing Protocol/1.0: Model and Semantics

[22] Internet Printing Protocol/1.0: Security

[23] Internet Printing Protocol/1.0: Protocol Specification (This document)

[24] Internet Printing Protocol/1.0: Directory Schema

[25] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119 , March 1997

[26] H. Alvestrand, " Tags for the Identification of Languages", RFC 1766, March 1995.

[27] R Fielding, et al, “Hypertext Transfer Protocol – HTTP/1.1” RFC 2068, January 1997

[28] Marcus Kuhn, “International Standard Date and Time Notation”, ISO 8601, http://www.ft.uni-erlangen.de/~mskuhn/iso-time.html

[29] D. Crocker et al., “Augmented BNF for Syntax Specifications: ABNF”, draft-ietf-drums-abnf-02.txt.

## 8. Author's Address

Robert Herriot (editor)
Sun Microsystems Inc.
901 San Antonio.Road, MPK-17
Palo Alto, CA 94303

Phone: 415-786-8995
Fax: 415-786-7077
Email: robert.herriot@eng.sun.com

Sylvan Butler
Hewlett-Packard
11311 Chinden Blvd.
Boise, ID 83714

Phone: 208-396-6000
Fax: 208-396-3457
Email: sbutler@boi.hp.com

Paul Moore
Microsoft
One Microsoft Way
Redmond, WA 98053

Phone: 425-936-0908
Fax: 425-93MS-FAX
Email: paulmo@microsoft.com

Randy Turner
Sharp Laboratories
5750 NW Pacific Rim Blvd
Camas, WA 98607

Phone: 360-817-8456
Fax: : 360-817-8436
Email: rturner@sharplabs.com

IPP Mailing List: ipp@pwg.org
IPP Mailing List Subscription: ipp-request@pwg.org
IPP Web Page: http://www.pwg.org/ipp/

## 9. Other Participants:

Chuck Adams - Tektronix
Ron Bergman - Data Products
Keith Carter - IBM
Angelo Caruso - Xerox
Jeff Copeland - QMS
Roger Debry - IBM
Lee Farrell - Canon
Sue Gleeson - Digital
Charles Gordon - Osicom
Brian Grimshaw - Apple
Jerry Hadsell - IBM
Richard Hart - Digital
Tom Hastings - Xerox
Stephen Holmstead
Zhi-Hong Huang - Zenographics
Scott Isaacson - Novell
Rich Lomicka - Digital
David Kellerman - Northlake Software
Robert Kline - TrueSpectra
Dave Kuntz - Hewlett-Packard
Takami Kurono - Brother
Rich Landau - Digital
Greg LeClair - Epson
Harry Lewis - IBM

Tony Liao - Vivid Image
David Manchala - Xerox
Carl-Uno Manros - Xerox
Jay Martin - Underscore
Larry Masinter - Xerox
Bob Pentecost - Hewlett-Packard
Patrick Powell - SDSU
Jeff Rackowitz - Intermec
Xavier Riley - Xerox
Gary Roberts - Ricoh
Stuart Rowley - Kyocera
Richard Schneider - Epson
Shigern Ueda - Canon
Bob Von Andel - Allegro Software
William Wagner - Digital Products
Jasper Wong - Xionics
Don Wright - Lexmark
Rick Yardumian - Xerox
Lloyd Young - Lexmark
Peter Zehler - Xerox
Frank Zhao - Panasonic
Steve Zilles - Adobe

NOTE: if I missed someone, please let me know.