Robert Herriot (editor)
Sun Microsystems
Sylvan Butler
Hewlett-Packard
Paul Moore
Microsoft.
Randy Turner
Sharp Labs
November 7, 1997

Internet Printing Protocol/1.0: Protocol Specification
draft-ietf-ipp-protocol-03~~2~~.txt


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), ds.internic.net (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 technology. The protocol is heavily influenced by the printing model introduced in the Document Printing Application (ISO/IEC 10175 DPA) standard [dpa]. 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~~
Requirements for an Internet Printing Protocol [ipp-req]
Internet Printing Protocol/1.0: Model and Semantics [ipp-mod]
Internet Printing Protocol/1.0: Protocol Specification (this document)

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 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 printer (server) side components.

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

Notice

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Table of Contents

# 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 [rfc2068] 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" [ipp-mod] defines the semantics of such a message body and the supported values. This document specifies the encoding of an IPP operation. The aforementioned document [ipp-mod] 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 [rfc2119].

# 3. Encoding of the Operation Layer

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

The encoding consists of octets as the most primitive type. There are several types built from octets, but three~~two~~ important types are integers, ~~and~~ character strings and octet strings, on which most other data types are built. Every character string in this encoding SHALL be a sequence of characters where the characters are associated with some charset and some natural language. ~~member of the UCS-2 coded character set~~. A character string MUST be in “network byte order” with the first character in the value (according to reading order) being the first character in the encoding. A character string whose associated charset is US-ASCII whose associated natural language is US English is henceforth called a US-ASCII-STRING. A character string whose associated charset and natural language are specified in a request or response as described in the model document is henceforth called a LOCALIZED-STRING. ~~and SHALL be encoded using UTF-8 which uses 1 to 3 octets per character~~. An octet string MUST be in “network byte order” with the first octet in the value (according to reading order) being the first octet in the encoding Every integer in this encoding SHALL be encoded as a signed integer using two’s-complement binary encoding with big-endian format (also known as “network order” and “most significant byte first”). The number of octets for an integer SHALL be 1, 2 or 4, depending on usage in the protocol. Such one-octet integers, henceforth called SIGNED-BYTE, are used for the version and tag fields. Such two-byte integers, henceforth called SIGNED-SHORT are used for the operation, status-code and length fields. Four byte integers, henceforth called SIGNED-INTEGER, are used for values fields.

The following two sections present the operation layer in two ways

- informally through pictures and description
- formally through Augmented Backus-Naur Form (ABNF), as specified by draft-ietf-drums-abnf-02.txt [abnf]

## 3.1 Picture of the Encoding

The encoding for an operation request or response consists of:

```
-----------------------------------------------
|                   version                   |   2 bytes  - required
-----------------------------------------------
|operation (request) or status-code (response)|   2 bytes  - required
-------------------------------------------------------------
|               xxx-attributes-tag            |   1 byte   |
-----------------------------------------------              |-0 or more
|             xxx-attribute-sequence          |   n bytes  |
-------------------------------------------------------------
|                  data-tag                   |   1 byte   - required
-----------------------------------------------
|                   data                      |   q bytes  - optional
-----------------------------------------------
```

The xxx-attributes-tag and xxx-attribute-sequence represents four different values of “xxx”, namely, operation, job, printer and unsupported-job. The xxx-attributes-tag and xxx-attribute-sequence may be omitted if the operation has no attributes or it may be repeated with the same or different values of “xxx” in ways that are specific to each operation. The data is omitted from some operations, but the data-tag is present even when the data is omitted. Note, the xxx-attributes-tags and data-tag are called ‘delimiter-tags’.

Note: the xxx-attribute-sequence, shown above may consist of 0 bytes, according to the rule below.

An xxx-attributes-sequence consists of zero or more compound-attributes.

```
-----------------------------------------------
|              compound-attribute             |   s bytes - 0 or more
-----------------------------------------------
```

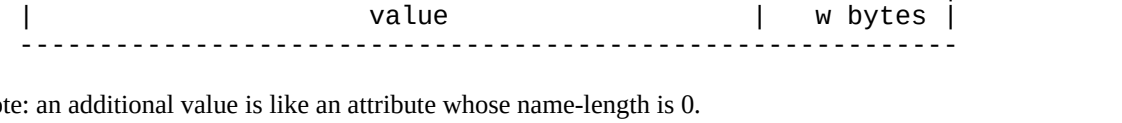
A compound-attribute consists of an attribute with a single value followed by zero or more additional values.

Note: a ‘compound-attribute’ represents a single attribute in the model document. The ‘additional value’ syntax is for attributes with 2 or more values.

Each attribute consists of:

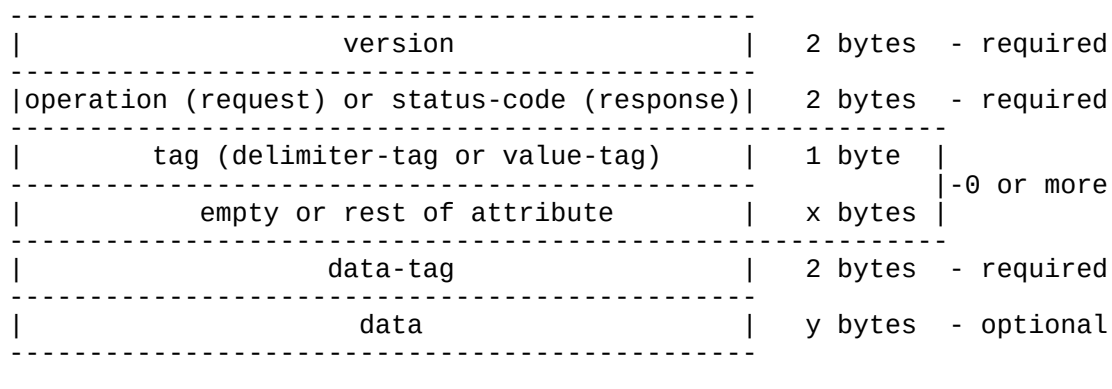
```
-----------------------------------------------
|                  value-tag                  |   1 byte
-----------------------------------------------
|               name-length  (value is u)     |   2 bytes
-----------------------------------------------
|                    name                     |   u bytes
-----------------------------------------------
|              value-length  (value is v)     |   2 bytes
-----------------------------------------------
|                    value                    |   v bytes
-----------------------------------------------
```

An additional value consists of:




Note: an additional value is like an attribute whose name-length is 0.

From the standpoint of a parsing loop, the encoding consists of:




The value of the tag determines whether the bytes following the tag are:

- attributes
- data
- the remainder of a single attribute where the tag specifies the type of the value.

## 3.2 Syntax of Encoding

The syntax below is ABNF [abnf] except ‘strings of literals’ SHALL be case sensitive. For example ‘a’ means lower case ‘a’ and not upper case ‘A’. In addition, SIGNED-BYTE and SIGNED-SHORT fields are represented as ‘%x’ values which show their range of values.

```
ipp-message = ipp-request / ipp-response
ipp-request = version operation
      *(xxx-attributes-tag  xxx-attribute-sequence) data-tag data
ipp-response = version status-code
      *(xxx-attributes-tag xxx-attribute-sequence)  data-tag  data
xxx-attribute-sequence = *compound-attribute
   ; where “xxx” in the three rules above stands for any of the following
   ; values:  “operation”,  “job”, “printer” or “unsupported-job”.


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-code = SIGNED-SHORT  ; mapping from model defined below

    compound-attribute = attribute *additional-values

    attribute = value-tag name-length name value-length value
    additional-values = value-tag 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 = %x00.00            ; name-length of 0
    operation-attributes-tag =  %x01                ; tag of 1
    job-attributes-tag          =  %x02             ; tag of 2
    printer-attributes-tag =  %x04                  ; tag of 4
    unsupported-job-attributes-tag =  %x05          ; tag of 5
    data-tag = %x03                                   ; tag of 3
    value-tag = %x10-FF

    SIGNED-BYTE = BYTE
    SIGNED-SHORT = 2BYTE
    DIGIT = %x30-39    ;  "0" to "9"
    LALPHA = %x61-7A   ;  "a" to "z"
    BYTE = %x00-FF
    OCTET-STRING = *BYTE
```

The syntax allows an xxx-attributes-tag to be present when the xxx-attribute-sequence that follows is empty. The syntax is defined this way to allow for the response of Get-Jobs where no attributes are returned for some job-objects. Although it is RECOMMENDED that the sender not send an xxx-attributes-tag if there are no attributes (except in the Get-Jobs response just mentioned), the receiver MUST be able to decode such syntax.

## 3.3 Version

The version SHALL consist of a major and minor version, each of which SHALL be represented by a SIGNED-BYTE. The protocol described in this document SHALL have a major version of 1 (0x01) and a minor version of 0 (0x00). The ABNF for these two bytes SHALL be %x01.00.

## 3.4 Mapping of Operations

Operations are defined as enums in the model document. An operations enum value SHALL be encoded as a SIGNED-SHORT

Note: the values 0x4000 to 0xFFFF are reserved for private extensions.

## 3.5 Mapping of Status-code

Status-codes are defined as enums in the model document. A status-code enum value SHALL be encoded as a SIGNED-SHORT

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

## 3.6 Tags

There are two kinds of tags:

- delimiter tags: delimit major sections of the protocol, namely attributes and data
- value tags: specify the type of each attribute value

### 3.6.1 Delimiter Tags

The following table specifies the values for the delimiter tags:

| Tag Value (Hex) | Delimiter |
|---|---|
| 0x00 | reserved |
| 0x01 | operation-attributes-tag |
| 0x02 | job-attributes-tag |
| 0x03 | data-tag |
| 0x04 | printer-attributes-tag |
| 0x05 | unsupported-job-attributes-tag |
| 0x06-0x0F | reserved for future delimiters |

When an xxx-attributes-tag occurs in the protocol, it SHALL mean that the zero or more following attributes up to the next delimiter tag are xxx attributes as defined in the model document, where xxx is operation, job, printer, unsupported-job.

Doing substitution for xxx in the above paragraph, this means the following. When an operation-attributes-tag occurs in the protocol, it SHALL mean that the zero or more following attributes up to the next delimiter tag are operation attributes as defined in the model document. When an job-attributes-tag occurs in the protocol, it SHALL mean that the zero or more following attributes up to the next delimiter tag are job attributes as defined in the model document. When an printer-attributes-tag occurs in the protocol, it SHALL mean that the zero or more following attributes up to the next delimiter tag are printer attributes as defined in the model document. When an unsupported-job-attributes-tag occurs in the protocol, it SHALL mean that the zero or more following attributes up to the next delimiter tag are unsupported-job attributes as defined in the model document.

The operation-attributes-tag and data-tag SHALL each occur exactly once in an operation. The operation-attributes-tag SHALL be the first tag delimiter, and the data-tag SHALL be the last tag delimiter.

Each of the other three ~~four~~ xxx-attributes-tags defined above is OPTIONAL in an operation and each SHALL occur at most once in an operation, except for job-attributes-tag in a Get-Jobs response which may occur zero or more times.

~~The data-tag SHALL occur exactly once in an operation.If an operation contains an operations-attribute-tag, it SHALL be the first tag delimiter. The data-tag SHALL be the last tag delimiter.~~

The order and presence of delimiter tags for each operation request and each operation response SHALL be that defined in the model document. For further details, see Section 3.8 Mapping of Attribute Names and Appendix B: Mapping of Each Operation in the Encoding.

### 3.6.2 Value Tags

The remaining tables show values for the value-tag, which is the first octet of an attribute. The value-tag specifies the type of the value of the attribute. If the value-tag specifies a type of compoundValue, it represents a compound value whose type is the that of the last member of the compound value. ~~The value of the value-tag of an attribute SHALL either be a type value specified in the model document or an "out-of-band" value, such as "unsupported" or "default". If the value of value-tag for an attribute is not "out-of-band" and differs from the value type specified by the model document, then a printer receiving such a request MAY reject the attribute or just the value. A client receiving such a response MAY ignore the attribute or just the value.~~

~~If ipp-attribute-fidelity is true and a printer rejects a value, it is the same as rejecting the attribute. If ipp-attribute-fidelity is false and a printer rejects a value, or it a client rejects a value, then it is as if the attribute didn't have that value. If after rejecting values, the attribute no longer has any values the attribute is rejected.~~

~~Note: the intent of the above rule is for servers to be able to understand text and name values when they don't support the naturalLanguage override for the value.~~

The following table specifies the "out-of-band" values for the value-tag.

| Tag Value (Hex) | Meaning |
|---|---|
| 0x10 | unsupported |
| 0x11 | reserved for future 'default' |
| 0x12 | unknown~~no-value~~ |
| 0x13 | compoundValue |
| 0x14-0x1F | reserved for future "out-of-band" values. |

The "unsupported" value SHALL be used in the attribute-sequence of an error response for those attributes which the printer does not support. The "default" value is reserved for future use of setting value back to their default value. The "unknown~~no-value~~" value is used for the value of a supported attribute when its value is temporarily unknown. ~~the "no-value" value in model, e.g. when a document-attribute is returned as a set of values and an attribute has no specified value for one or more of the documents~~. The "compoundValue" SHALL be used to form a single value from a collection of values, and its value is the number of members forming the compound value, excluding the compoundValue. For example, a text value with a naturalLanguage override consists of 3 "values": a compoundValue with value 2, a naturalLanguage value and a text value.

The following table specifies the integer values for the value-tag

| Tag Value (Hex) | Meaning |
|---|---|
| 0x20 | reserved |
| 0x21 | integer |
| 0x22 | boolean |
| 0x23 | enum |
| 0x24-0x2F | reserved for future integer types |

NOTE: 0x20 is reserved for "generic integer" if should ever be needed.

The following table specifies the octetString values for the value-tag

| Tag Value (Hex) | Meaning |
|---|---|
| 0x30 | octetString with an unspecified format |
| 0x31 | dateTime |
| 0x32 | resolution |

| Tag Value (Hex) | Meaning |
|---|---|
| 0x33 | rangeOfInteger |
| 0x34 | reserved for dictionary (in the future) |
| 0x35-0x3F | reserved for future octetString types |

The following table specifies the character-string values for the value-tag

| Tag Value (Hex) | Meaning |
|---|---|
| 0x40 | reserved |
| 0x41 | text |
| 0x42 | name |
| 0x43 | reserved |
| 0x44 | keyword |
| 0x45 | uri |
| 0x46 | uriScheme |
| 0x47 | charsSet |
| 0x48 | naturalLanguage |
| 0x49 | mimeMediaType |
| 0x4A-0x5F | reserved for future character string types |

NOTE: 0x40 is reserved for “generic character-string” if should ever be needed.

The values 0x60-0xFF are reserved for future types. There are no values allocated for private extensions. A new type must be registered via the type 2 process.

## 3.7 Name-Lengths

The name-length field SHALL consist of a SIGNED-SHORT. This field SHALL specify the number of octets in the name field 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 attribute. Within an attribute-sequence, if two attributes have the same name, the first occurrence SHALL be ignored. The zero-length name is the only mechanism for multi-valued attributes.

## 3.8 Mapping of Attribute Names

Some attributes are encoded in a special position. These attribute are:

- “printer-uri”: The target printer-uri of each operation in the IPP model document SHALL be specified outside of the operation layer as the request-URI on the Request-Line at the HTTP level.
- “job-uri”: The target job-uri of each operation in the IPP model document SHALL be specified outside of the operation layer as the request-URI on the Request-Line at the HTTP level.
- “document-content”: The attribute named “document-content” in the IPP model document SHALL become the “data” in the operation layer.
- “status-code”: The attribute named “status-code” in the IPP model document SHALL become the “status-code” field in the operation layer response.

The model document arranges the remaining attributes into groups for each operation request and response. Each such group SHALL be represented in the protocol by an xxx-attribute-sequence preceded by the appropriate xxx-attributes-tag (See the table

below and Appendix B: Mapping of Each Operation in the Encoding). In addition, the order of these xxx-attributes-tags and xxx-attribute-sequences in the protocol SHALL be the same as in the model document, but the order of attributes within each xxx-attribute-sequence SHALL be unspecified. The table below maps the model document group name to xxx-attributes-sequence

| **Model Document Group** | ***xxx*-attributes-sequence** |
|---|---|
| Operation Attributes | operations-attributes-sequence |
| Job Template Attributes | job-attributes-sequence |
| Job Object Attributes | job-attributes-sequence |
| Unsupported Attributes | unsupported-job-attributes-sequence |
| Requested Attributes (Get-Attributes of job object) | job-attributes-sequence |
| Requested Attributes (Get-Attributes of printer object) | printer-attributes-sequence |
| Document Content | in a special position as described above |

ISSUE: coordinate this with the model document.

If an operation contains attributes from more than one job object (e.g. Get-Jobs response), the attributes from each job object SHALL be in a separate job-attribute-sequence, such that the attributes from the ith job object are in the ith job-attribute-sequence. See Section 11 “Appendix B: Mapping of Each Operation in the Encoding” for table showing the application of the rules above.

## 3.9 Value Lengths

Each attribute value SHALL be preceded by a SIGNED-SHORT which SHALL specify the number of octets in the value which follows this length, exclusive of the two bytes specifying the length.

For any of the types represented by binary signed integers, the sender MUST encode the value in exactly four octets..

For any of the types represented by character-strings, the sender MUST encode the value with all the characters of the string and without any padding characters.

If a value-tag contains an “out-of-band” value which is not compoundValue, such as “unsupported”, the value-length SHALL be 0 and the value empty — the value has no meaning when the value-tag has an “out-of-band” value. If a printer or client receives an operation with a nonzero value-length in this case, it SHALL ignore the value field.

## 3.10 Mapping of Attribute Values

~~The following SHALL be the mapping of attribute values to their IPP encoding in the value field.~~ The syntax types and most of the details of their representation are defined in the IPP model document. The table below augments the information in the model document, and defines the syntax types from the model document in terms of the 5 basic types defined in section 3 Encoding of the Operation Layer. The 5 types are US-ASCII-STRING, LOCALIZED-STRING, SIGNED-INTEGER, SIGNED-SHORT, SIGNED-BYTE, and OCTET-STRING.

| **Syntax of Attribute Value** | **Encoding** |
|---|---|
| text, name | LOCALIZED-STRING.<br>The override natural language mechanism is encoded by syntactically preceding the text or name value by two values: first a value of type compoundValue whose value is 2 and second a value of type naturalLanguage whose value is the language override. From a protocol syntax view, there are three separate values: the compoundValue, the naturalLanguage value and the text or name value, but from a semantic view, the Printer treats them as a single value where the naturalLanguage value overrides the language of |

| Syntax of Attribute Value | Encoding |
|---|---|
| | the immediately following text or name value in the attribute. The override applies to just the text or name within the compound value. Other text or name values needing an override must be overridden with additional compoundValues. |
| charset, naturalLanguage, mimeMediaType, keyword, uri, and uriScheme | US-ASCII-STRING |
| boolean | SIGNED-BYTE ~~one binary octet~~ where 0x00 is 'false' and 0x01 is 'true' |
| integer and enum | a SIGNED-INTEGER~~, defined previously as a signed integer using two's-complement binary encoding in four octets with big-endian format (also known as "network order" and "most significant byte first").~~ |
| compoundValue | a SIGNED-INTEGER with a special meaning. ~~has the same representation as an integer, but with a different meaning.~~ If the value of a compoundValue is n, then the n following values of the attribute form a single value whose type is that of the last member of the compound value. For example, if an attribute has 3 successive values: compoundValue of 2, naturalLanguage of 'fr-CA' and name of 'chien', then these three "values" form a single value which is a name of 'chien' in Canadian French.. |
| dateTime | OCTET-STRING consisting of eleven octets whose contents are defined by "DateAndTime" in RFC 1903 [rfc1903]. Although RFC 1903 also defines an eight octet format which omits the time zone, a value of this type in the IPP protocol MUST use the eleven octet format. [ transfer to model]. |
| resolution | OCTET_STRING consisting of nine octets ~~consisting~~ of 2 SIGNED-INTEGERs followed by a SIGNED-BYTE. ~~The values are the same as those specified in RFC 1759 (Printer MIB) [r1759].~~ The first SIGNED-INTEGER contains the value of cross feed direction resolution ~~prtMarkerAddressabilityXFeedDir~~. The second SIGNED-INTEGER contains the value of feed direction resolution~~prtMarkerAddressabilityFeedDir~~. The SIGNED-BYTE contains the unts value ~~of prtMarkerAddressabilityUnit. Note: the latter value is either 3 (tenThousandsOfInches) or 4 (micrometers) and the addressability is in 10,000 units of measure. Thus the SIGNED-INTEGERS represent integral values in either dots-per-inch or dots-per-centimeter.~~ |
| rangeOfInteger | Eight octets consisting of 2 SIGNED-INTEGERs. The first SIGNED-INTEGERs contains the lowerest bound ~~value of the range~~ and the second SIGNED-INTEGERs contains the upper ~~highest~~ bound~~value of the range~~ |
| 1setOf X | encoding according to the rules for an attribute with more than 1~~more~~ value. Each value X is encoded according to the rules for encoding its type. |
| octetString | OCTET-STRING |

The type of the value in the model document determines the encoding in the value and the value of the value-tag.

### 3.11 Data

The data part SHALL include any data required by the operation

# 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.

It is REQUIRED that a printer support HTTP over port 80, though a printer may support HTTP over port 516 or some other port. In addition, a printer may have to support another port for secure connections.

Note: Consistent with RFC 2068 (HTTP/1.1), HTTP URI's for IPP implicitly reference port 80. If a URI references some other port, the port number must be explicitly specified in the URI.

Each HTTP operation shall use the POST method where the request-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 3.2 "Syntax of Encoding". A client implementation SHALL adhere to the rules for a client described in RFC 2068 [rfc2068]. A printer (server) implementation SHALL adhere the rules for an origin server described in RFC 2068.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.

- **must:** the client or server MUST send the header,
- **must-if:** the client or server MUST send the header when the condition described in the "values and conditions" column is met,
- **may:** the client or server MAY send the header
- **not:** 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.

- **must:** the client or server MUST support the header,
- **may:** the client or server MAY support the header
- **not:** 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 | must | not | must | not | "no-cache" only |
| Connection | must-if | must | must-if | must | "close" only. Both client and server SHOULD keep a connection for the |

| General-Header | Request | | Response | | Values and Conditions |
|---|---|---|---|---|---|
| | Client | Server | Server | Client | |
| | | | | | duration of a sequence of operations. The client and server MUST include this header for the last operation in such a sequence. |
| Date | may | may | must | may | per RFC 1123 [rfc1123] |
| Pragma‘ | must | not | must | not | “no-cache” only |
| Transfer-Encoding | must-if | must | must-if | must | “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 | may | must | “application/ipp” only. This value is the default if the client omits it |
| Accept-Charset | not | not | Charset information is within the application/ipp entity |
| Accept-Encoding | may | must | empty and per RFC 2068 [rfc2068] and IANA registry for content-codings |
| Accept-Language | not | not | . language information is within the application/ipp entity |
| Authorization | must-if | must | per RFC 2068. A client MUST send this header when it receives a 401 “Unauthorized” response and does not receive a “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 | must | must | 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 | must-if | 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 |
|---|---|---|---|

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

## 4.4 Entity Headers

The following is a table for the entity 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 | may | must | must | must | per RFC 2068 and IANA registry for content codings. |
| Content-Language | not | not | not | not | Application/ipp handles language |
| Content-Length | must-if | must | must-if | must | 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 | may | may | may | may | per RFC 2068 |
| Content-Range | not | not | not | not | |
| Content-Type | must | must | must | must | “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 of IPP, the security considerations outlined in RFC 2068 [rfc2068] apply. Specifically, IPP servers can generate a 401 “Unauthorized” response code to request client authentication and IPP clients should correctly respond with the proper “Authorization” header. Both Basic Authentication (RFC 2068) and Digest Authentication (RFC 2069) [rfc2069] flavors of authentication SHALL be supported. The server chooses which type(s) of authentication to accept. Digest Authentication is a more secure method, and is always preferred to Basic Authentication.

For secure communication (privacy in particular), IPP SHOULD be run using a secure communications channel. For this purpose it is the intention to define standardization of IPP in combination with Transport Layer Security (TLS), currently under development in the IETF, when the TLS specifications are agreed and on the IETF standards track.

As an intercept solution for secure communication, the Secure Socket Layer 3.0 (SSL3) could be used, but be warned that such implementations may not be able to interoperate with a future standardized IPP and TLS solution. Appendix C gives some hints to implementors wanting to use SSL3 as intercept solution.

It is possible to combine the techniques, HTTP 1.1 client authentication (either basic or digest) with a secure communications channel. Together the two are more secure than client authentication and they perform user authentication.

See further discussion of IPP security concepts in the model document [ipp-mod].

# 6. Copyright



# 7. References

[rfc822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", RFC 822, August 1982.

[rfc1123] Braden, S., "Requirements for Internet Hosts - Application and Support", RFC 1123, October, 1989,

[rfc1179] McLaughlin, L. III, (editor), "Line Printer Daemon Protocol" RFC 1179, August 1990.

[rfc1630] T. Berners-Lee, "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the Word-Wide Web", RFC 1630, June 1994.

[rfc1759] Smith, R., Wright, F., Hastings, T., Zilles, S., and Gyllenskog, J., "Printer MIB", RFC 1759, March 1995.

[rfc1738] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource Locators (URL)", RFC 1738, December, 1994.

[rfc1543] Postel, J., "Instructions to RFC Authors", RFC 1543, October 1993.

[rfc1766] H. Alvestrand, " Tags for the Identification of Languages", RFC 1766, March 1995.

[rfc1903} J. Case, et al. "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

[rfc2046] N. Freed & N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. November 1996. (Obsoletes RFC1521, RFC1522, RFC1590), RFC 2046.

[rfc2048] N. Freed, J. Klensin & J. Postel. Multipurpose Internet Mail Extension (MIME) Part Four: Registration Procedures. November 1996. (Format: TXT=45033 bytes) (Obsoletes RFC1521, RFC1522, RFC1590) (Also BCP0013), RFC 2048.

[rfc2068] R Fielding, et al, “Hypertext Transfer Protocol – HTTP/1.1” RFC 2068, January 1997

[rfc2069] J. Franks, et al, “An Extension to HTTP: Digest Access Authentication” RFC 2069, January 1997

[rfc2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119 , March 1997

[rfc2184] N. Freed, K. Moore, “MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations”, RFC 2184, August 1997,

[abnf] D. Crocker et al., “Augmented BNF for Syntax Specifications: ABNF”, draft-ietf-drums-abnf-04.txt.

[char] N. Freed, J. Postel: IANA CharsSet Registration Procedures, Work in Progress (draft-freed-charset-reg-02.txt).

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

[iana] IANA Registry of Coded Character Sets: ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets

[ipp-req] Wright, F. D., “Requirements for an Internet Printing Protocol:”

[ipp-mod] Isaacson, S, deBry, R, Hastings, T, Herriot, R, Powell, P, “Internet Printing Protocol/1.0: Model and Semantics”

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

# 8. Author’s Address

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

Phone: 650-786-8995
Fax: 650-786-7077
Email: robert.herriot@eng.sun.com

Paul Moore
Microsoft
One Microsoft Way
Redmond, WA 98053

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

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

Phone: 208-396-6000
Fax: 208-396-3457
Email: sbutler@boi.hp.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
Ira McDonald, 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

# 10. Appendix A: Protocol Examples

## 10.1 Print-Job Request

The following is an example of a Print-Job request with job-name, copies, and sides specified.

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x0100 | 1.0 | version |
| 0x0002 | PrintJob | operation |
| 0x01 | start operation-attributes | operation-attributes-tag |
| 0x47 | charset type | value-tag |
| 0x0012 | | name-length |
| attributes-charset | attributes-charset | name |
| 0x0008 | | value-length |
| US-ASCII | US-ASCII | value |
| 0x48 | natural-language type | value-tag |
| 0x001B | | name-length |
| attributes-natural-language | attributes-natural-language | name |
| 0x0005 | | value-length |
| en-US | en-US | value |
| 0x42 | name type | value-tag |
| 0x0008 | | name-length |
| job-name | job-name | name |

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x0006 | | value-length |
| foobar | foobar | value |
| 0x02 | start job-attributes | job-attributes-tag |
| 0x21 | integer type | value-tag |
| 0x0005 | | name-length |
| copies | copies | name |
| 0x0004 | | value-length |
| 0x00000014 | 20 | value |
| 0x44 | keyword type | value-tag |
| 0x0005 | | name-length |
| sides | sides | name |
| 0x0013 | | value-length |
| two-sided-long-edge | two-sided-long-edge | value |
| 0x03 | start-data | data-tag |
| %!PS... | <PostScript> | data |

## 10.2 Print-Job Response (successful)

Here is an example of a Print-Job response which is successful:

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x0100 | 1.0 | version |
| 0x0000 | OK (successful) | status-code |
| 0x01 | start operation-attributes | operation-attributes-tag |
| 0x47 | charset type | value-tag |
| 0x0012 | | name-length |
| attributes-charset | attributes-charset | name |
| 0x0008 | | value-length |
| US-ASCII | US-ASCII | value |
| 0x48 | natural-language type | value-tag |
| 0x001B | | name-length |
| attributes-natural-language | attributes-natural-language | name |
| 0x0005 | | value-length |
| en-US | en-US | value |
| 0x41 | text type | value-tag |
| 0x000E | | name-length |
| status-message | status-message | name |
| 0x0002 | | value-length |
| OK | OK | value |
| 0x02 | start job-attributes | job-attributes-tag |
| 0x21 | integer | value-tag |
| 0x0007 | | name-length |
| job-id | job-id | name |
| 0x0004 | | value-length |
| 147 | 147 | value |
| 0x45 | uri type | value-tag |
| 0x0008 | | name-length |
| job-uri | job-uri | name |
| 0x000E | | value-length |
| http://foo/123 | http://foo/123 | value |

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x25 | name type | value-tag |
| 0x0008 | | name-length |
| job-state | job-state | name |
| 0x0001 | | value-length |
| 0x03 | pending | value |
| 0x03 | start-data | data-tag |

## 10.3 Print-Job Response (failure)

Here is an example of a Print-Job response which fails because the printer does not support sides and because the value 20 for copies is not supported:

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x0100 | 1.0 | version |
| 0x0400 | client-error-bad-request | status-code |
| 0x01 | start operation-attributes | operation-attribute tag |
| 0x47 | charset type | value-tag |
| 0x0012 | | name-length |
| attributes-charset | attributes-charset | name |
| 0x0008 | | value-length |
| US-ASCII | US-ASCII | value |
| 0x48 | natural-language type | value-tag |
| 0x001B | | name-length |
| attributes-natural-language | attributes-natural-language | name |
| 0x0005 | | value-length |
| en-US | en-US | value |
| 0x41 | text type | value-tag |
| 0x000E | | name-length |
| status-message | status-message | name |
| 0x000D | | value-length |
| bad-request | bad-request | value |
| 0x04 | start unsupported-job-attributes | unsupported-job-attributes-tag |
| 0x21 | integer type | value-tag |
| 0x0005 | | name-length |
| copies | copies | name |
| 0x0004 | | value-length |
| 0x00000014 | 20 | value |
| 0x10 | unsupported (type) | value-tag |
| 0x0005 | | name-length |
| sides | sides | name |
| 0x0000 | | value-length |
| 0x03 | start-data | data-tag |

## 10.4 Print-URI Request

The following is an example of Print-URI request with copies and job-name parameters.

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x0100 | 1.0 | version |
| 0x0003 | Print-URI | operation |

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x01 | start operation-attributes | operation-attributes-tag |
| 0x47 | charset type | value-tag |
| 0x0012 | | name-length |
| attributes-charset | attributes-charset | name |
| 0x0008 | | value-length |
| US-ASCII | US-ASCII | value |
| 0x48 | natural-language type | value-tag |
| 0x001B | | name-length |
| attributes-natural-language | attributes-natural-language | name |
| 0x0005 | | value-length |
| en-US | en-US | value |
| 0x45 | uri type | value-tag |
| 0x000A | | name-length |
| document-uri | document-uri | name |
| 0x11 | | value-length |
| ftp://foo.com/foo | ftp://foo.com/foo | value |
| 0x42 | name type | value-tag |
| 0x0008 | | name-length |
| job-name | job-name | name |
| 0x0006 | | value-length |
| foobar | foobar | value |
| 0x02 | start job-attributes | job-attributes-tag |
| 0x21 | integer type | value-tag |
| 0x0005 | | name-length |
| copies | copies | name |
| 0x0004 | | value-length |
| 0x00000001 | 1 | value |
| 0x03 | start-data | data-tag |
| %!PS... | <PostScript> | data |

## 10.5 Create-Job Request

The following is an example of Create-Job request with no parameters and no attributes

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x0100 | 1.0 | version |
| 0x0005 | Create-Job | operation |
| 0x01 | start operation-attributes | operation-attributes-tag |
| 0x47 | charset type | value-tag |
| 0x0012 | | name-length |
| attributes-charset | attributes-charset | name |
| 0x0008 | | value-length |
| US-ASCII | US-ASCII | value |
| 0x48 | natural-language type | value-tag |
| 0x001B | | name-length |
| attributes-natural-language | attributes-natural-language | name |
| 0x0005 | | value-length |
| en-US | en-US | value |
| 0x03 | start-data | data-tag |

## 10.6 Get-Jobs Request

The following is an example of Get-Jobs request with parameters but no attributes.

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x0100 | 1.0 | version |
| 0x000A | Get-Jobs | operation |
| 0x01 | start operation-attributes | operation-attributes-tag |
| 0x47 | charset type | value-tag |
| 0x0012 | | name-length |
| attributes-charset | attributes-charset | name |
| 0x0008 | | value-length |
| US-ASCII | US-ASCII | value |
| 0x48 | natural-language type | value-tag |
| 0x001B | | name-length |
| attributes-natural-language | attributes-natural-language | name |
| 0x0005 | | value-length |
| en-US | en-US | value |
| 0x21 | integer type | value-tag |
| 0x0005 | | name-length |
| limit | limit | name |
| 0x0004 | | value-length |
| 0x00000032 | 50 | value |
| 0x44 | keyword type | value-tag |
| 0x0014 | | name-length |
| requested-attributes | requested-attributes | name |
| 0x0006 | | value-length |
| job-id | job-id | value |
| 0x44 | keyword type | value-tag |
| 0x0000 | additional value | name-length |
| 0x0008 | | value-length |
| job-name | job-name | value |
| 0x03 | start-data | data-tag |

## 10.7 Get-Jobs Response

The following is an of Get-Jobs response from previous request with 3 jobs. The Printer returns no information about the second job.

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x0100 | 1.0 | version |
| 0x0000 | OK (successful) | status-code |
| 0x01 | start operation-attributes | operation-attribute-tag |
| 0x47 | charset type | value-tag |
| 0x0012 | | name-length |
| attributes-charset | attributes-charset | name |
| 0x0008 | | value-length |
| ISO-8859-1 | ISO-8859-1 | value |
| 0x48 | natural-language type | value-tag |
| 0x001B | | name-length |
| attributes-natural-language | attributes-natural-language | name |
| 0x0005 | | value-length |
| en-US | en-US | value |

| Octets | Symbolic Value | Protocol field |
|---|---|---|
| 0x41 | text type | value-tag |
| 0x000E | | name-length |
| status-message | status-message | name |
| 0x0002 | | value-length |
| OK | OK | value |
| 0x02 | start job-attributes (1st object) | job-attributes-tag |
| 0x48 | natural-language type | value-tag |
| 0x001B | | name-length |
| attributes-natural-language | attributes-natural-language | name |
| 0x0005 | | value-length |
| fr-CA | fr-CA | value |
| 0x21 | integer type | value-tag |
| 0x0006 | | name-length |
| job-id | job-id | name |
| 0x0004 | | value-length |
| 147 | 147 | value |
| 0x42 | name type | value-tag |
| 0x0008 | | name-length |
| job-name | job-name | name |
| 0x0003 | | name-length |
| fou | fou | name |
| 0x02 | start job-attributes (2nd object) | job-attributes-tag |
| 0x02 | start job-attributes (3rd object) | job-attributes-tag |
| 0x21 | integer type | value-tag |
| 0x0006 | | name-length |
| job-id | job-id | name |
| 0x0004 | | value-length |
| 148 | 148 | value |
| 0x13 | compoundValue | value-tag |
| 0x0008 | | name-length |
| job-name | job-name | name |
| 0x0004 | | value-length |
| 0x0002 | 2 | value (number of values) |
| 0x48 | naturalLanguage | value-tag |
| 0x0000 | multi-value marker | name-length |
| 0x0005 | | value-length |
| de-CH | de-CH | value |
| 0x42 | name type | value-tag |
| 0x0000 | multi-value marker | name-length |
| 0x0003 | | name-length |
| isch guet | isch guet | name |
| 0x03 | start-data | data-tag |

# 11. Appendix B: Mapping of Each Operation in the Encoding

The next three tables show the results of applying the rules above to the operations defined in the IPP model document. There is no information in these tables that cannot be derived from the rules presented in Section 3.8 “Mapping of Attribute Names”.

The following table shows the mapping of all IPP model-document request attributes to an appropriate xxx-attribute-sequence or special position in the protocol.

The table below shows the attributes for operations sent to a Printer URI.

| **Operation** | **operation attributes** | **job attributes** | **special position** |
|---|---|---|---|
| Print-Job | attributes-charset<br>attributes-natural-language<br>job-name<br>document-name<br>ipp-attribute-fidelity<br>~~document-charset~~<br>document-natural-language | job-template attributes | document-content |
| Create-Job or Validate-Job | attributes-charset<br>attributes-natural-language<br>job-name<br>ipp-attribute-fidelity | job-template attributes | |
| Print-URI | attributes-charset<br>attributes-natural-language<br>job-name<br>ipp-attribute-fidelity<br>document-uri<br>~~document-charset~~<br>document-natural-language | job-template attributes | |
| Send-Document | attributes-charset<br>attributes-natural-language<br>job-id<br>last-document<br>document-name<br>~~document-charset~~<br>document-natural-language | | document-content |
| Send-URI | attributes-charset<br>attributes-natural-language<br>job-id<br>last-document<br>document-name<br>document-uri<br>~~document-charset~~<br>document-natural-language | | |
| Cancel-Job | attributes-charset<br>attributes-natural-language<br>job-id<br>message | | |
| Get-Attributes<br>(for a Printer) | attributes-charset<br>attributes-natural-language<br>requested-attributes<br>document-format | | |
| Get-Attributes<br>(for a Job) | attributes-charset<br>attributes-natural-language<br>job-id<br>requested-attributes | | |
| Get-Jobs | attributes-charset<br>attributes-natural-language<br>limit<br>requested-attributes<br>which-jobs | | |

The table below shows the attributes for operations sent to a Job URI.

| Operation | operation attributes | job attributes | special position |
|---|---|---|---|
| Send-Document | attributes-charset<br>attributes-natural-language<br>last-document<br>document-name<br>~~document-charset~~<br>document-natural-language | | document-content |
| Send-URI | attributes-charset<br>attributes-natural-language<br>last-document<br>document-name<br>document-uri<br>~~document-charset~~<br>document-natural-language | | |
| Cancel-Job | attributes-charset<br>attributes-natural-language<br>message | | |
| Get-Attributes (for a Job) | attributes-charset<br>attributes-natural-language<br>requested-attributes | | |

The following two tables shows the mapping of all IPP model-document response attributes to an appropriate xxx-attribute-sequence or special position in the protocol.

| Operation | operation attributes | job-attributes | unsupported-job-attributes | special position |
|---|---|---|---|---|
| Print-Job, Print-URI, Create-Job, Send-Document or Send-URI | attributes-charset<br>attributes-natural-language<br>status-message | job-id<br>job-uri<br>job-state<br>job-state-reasons<br>job-state-message<br>number-of-intervening-jobs | unsupported attributes | status-code |
| Validate-Job | attributes-charset<br>attributes-natural-language<br>status-message | | unsupported attributes | status-code |

Note: the unsupported-job-attributes are present only if the client included some job attributes that the Printer doesn't support.

Note: the job-attributes are present only if the server returns the status code of successful-ok or successful-ok-ignored-or-substituted-attributes.

| Operation | operation attributes | job-attributes | printer-attributes | special position |
|---|---|---|---|---|
| Cancel-Job | attributes-charset<br>attributes-natural-language<br>status-message | | | status-code |
| Get-Attributes (of a job) | attributes-charset<br>attributes-natural- | requested attributes | | status-code |

| Operation | operation attributes | job-attributes | printer-attributes | special position |
|---|---|---|---|---|
| | language<br>status-message | | | |
| Get-Attributes<br>(of a printer) | attributes-charset<br>attributes-natural-language<br>status-message | | requested attributes | status-code |
| Get-Jobs | attributes-charset<br>attributes-natural-language<br>status-message | requested attributes<br>(see the Note below) | | status-code |

Note for Get-Jobs: there is a separate job-attribute-sequence containing requested-attributes for each job object in the response

# 12. Appendix C: Hints to implementors using IPP with SSL3

WARNING: Clients and IPP objects using intermediate secure connection protocol solutions such as IPP in combination with Secure Socket Layer Version 3 (SSL3), which are developed in advance of IPP and TLS standardization, might not be interoperable with IPP and TLS standards-conforming clients and IPP objects.

An assumption is that the URI for a secure IPP Printer object has been found by means outside the IPP printing protocol, via a directory service, web site or other means.

IPP provides a transparent connection to SSL by calling the corresponding URL (a https URI connects by default to port 443). However, the following functions can be provided to ease the integration of IPP with SSL during implementation.

connect (URI), returns a status.

"connect" makes an https call and returns the immediate status of the connection as returned by SSL to the user. The status values are explained in section 5.4.2 of the SSL document [ssl].

A session-id may also be retained to later resume a session. The SSL handshake protocol may also require the cipher specifications supported by the client, key length of the ciphers, compression methods, certificates, etc. These should be sent to the server and hence should be available to the IPP client (although as part of administration features).

disconnect (session)

to disconnect a particular session.

The session-id available from the "connect" could be used.

resume (session)

to reconnect using a previous session-id.

The availability of this information as administration features are left for implementors, and need not be standardized at this time