Internet Printing Protocol WG
INTERNET-DRAFT
draft-ietf-ipp-implementers-guide-v11-03~~2~~.txt
Obsoletes RFC 2639
[Target category: informational]
Expires: January 17, 2002

T. Hastings
C. Manros
P. Zehler
Xerox Corporation
C. Kugler
IBM Printing Systems Co
H. Holst
i-data Printing Systems
July 17~~January 25~~, 2001

Internet Printing Protocol/1.1: Implementer's Guide

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 technologies. This document contains information that supplements the IPP Model and Semantics [RFC2911] and the IPP Transport and Encoding [RFC2910] documents. It is intended to help implementers understand IPP/1.1, as well as IPP/1.0 [RFC2565, RFC2566], and some of the considerations that may assist them in the design of their client and/or IPP object implementations. For example, a typical order of processing requests is given, including error checking. Motivation for some of the specification decisions is also included.

This document obsoletes RFC 2639 which was the Implementer’s Guide for IPP/1.0.

~~The full set of IPP documents includes:~~

- ~~Design Goals for an Internet Printing Protocol [RFC2567]~~
- ~~Rationale for the Structure and Model and Protocol for the Internet Printing Protocol [RFC2568]~~
- ~~Internet Printing Protocol/1.1: Model and Semantics [RFC2911]~~
- ~~Internet Printing Protocol/1.1: Encoding and Transport [RFC2910]~~
- ~~Mapping between LPD and IPP Protocols [RFC2569]~~

~~The document, "Design Goals for an Internet Printing Protocol", 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 design goal document calls out a subset of end user requirements that are satisfied in IPP/1.1. Operator and administrator requirements are out of scope for version 1.1.~~

~~The document, "Rationale for the Structure and Model and Protocol for the Internet Printing Protocol", describes IPP from a high level view, defines a roadmap for the various documents that form the suite of IPP specifications, and gives background and rationale for the IETF working group's major decisions.~~

~~The document, "Internet Printing Protocol/1.1: Model and Semantics", describes a simplified model with abstract objects, their attributes, and their operations. The model introduces a Printer and a Job. The Job supports multiple documents per Job. The model document also addresses how security, internationalization, and directory issues are addressed.~~

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

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

TABLE OF CONTENTS

TABLES

1 Introduction

The IPP Implementer's Guide (IIG) (this document) contains information that supplements the IPP Model and Semantics [RFC2911] and the IPP Transport and Encoding [RFC2910] documents. This document is just one of a suite of documents that fully define IPP. The base set of IPP documents includes:

Design Goals for an Internet Printing Protocol [RFC2567]
Rationale for the Structure and Model and Protocol for the Internet Printing Protocol [RFC2568]
Internet Printing Protocol/1.1: Model and Semantics [RFC2911]
Internet Printing Protocol/1.1: Encoding and Transport [RFC2910]
Internet Printing Protocol/1.1: Implementer's Guide (this document)
Mapping between LPD and IPP Protocols [RFC2569]

See section 10 for a description of these base IPP documents. Anyone reading these documents for the first time is strongly encouraged to read the IPP documents in the above order.

As such the~~is~~ information in this document is not part of the formal specification~~s~~ of IPP/1.1. Instead information is presented to help implementers understand ~~the specification~~IPP/1.1, as well as IPP/1.0 [RFC2565, RFC2566], including some of the motivation for decisions taken by the committee in developing the specification. Some of the implementation considerations are intended to help implementers design their client and/or IPP object implementations. If there are any contradictions between this document and [RFC2911] or [RFC2910], those documents take precedence over this document.

Platform-specific implementation considerations will be included in this guide as they become known.

Note: In order to help the reader of the IIG and the IPP Model and Semantics document, the sections in this document parallel the corresponding sections in the Model document and are numbered the same for ease of cross reference. The sections that correspond to the IPP Transport and Encoding are correspondingly offset.

1.1 Conformance language

Usually, this document does not contain the terminology MUST, MUST NOT, MAY, NEED NOT, SHOULD, SHOULD NOT, REQUIRED, and OPTIONAL. However, when those terms do appear in this document, their intent is to repeat what the [RFC2911] and [RFC2910] documents require and allow, rather than specifying additional conformance requirements. These terms are defined in section 12 on conformance terminology in [RFC2911], most of which is taken from RFC 2119 [RFC2119].

Implementers should read section 12 (APPENDIX A) in [RFC2911] in order to understand these capitalized words. The words MUST, MUST NOT, and REQUIRED indicate what implementations are required to support in a client or IPP object in order to be conformant to [RFC2911] and [RFC2910]. MAY, NEED NOT, and OPTIONAL indicate was is merely allowed as an implementer option. The verbs SHOULD and SHOULD NOT indicate suggested behavior, but which is not required or disallowed, respectively, in order to conform to the specification.

1.2 Other terminology

This document uses other terms, such as "attributes", "operation", and "Printer" as defined in [RFC2911] section 12. In addition, the term "sender" refers to the client that sends a request or an IPP object that returns a response. The term "receiver" refers to the IPP object that receives a request and to a client that receives a response.

1.3 Issues Raised from Interoperability Testing Events

The IPP WG has conducted three open Interoperability Testing Events. The first one was held in September 1998, the second one was held in March 1999, and the third one was held in October 2000. See the summary reports in:

ftp://ftp.pwg.org/pub/pwg/ipp/new_TES/

The issues raised from the first Interoperability Testing Event are numbered 1.n in this document and have been incorporated into "IPP/1.0 Model and Semantics" [RFC2566] and the "IPP/1.0 Encoding and Transport" [RFC2565] documents. However, some of the discussion is left here in the Implementer's Guide to help understanding.

The issues raised from the second Interoperability Testing Event are numbered 2.n in this document have been incorporated into "IPP/1.1 Model and Semantics" [RFC2911] and the "IPP/1.1 Encoding and Transport" [RFC2910] documents. However, some of the discussion is left here in the Implementer's Guide to help understanding.

The issues raised from the third Interoperability Testing Event are numbered 3.n in this document and are described in:

ftp://ftp.pwg.org/pub/pwg/ipp/Issues/Issues-raised-at-Bake-Off3.pdf
ftp://ftp.pwg.org/pub/pwg/ipp/Issues/Issues-raised-at-Bake-Off3.doc
ftp://ftp.pwg.org/pub/pwg/ipp/Issues/Issues-raised-at-Bake-Off3.txt

2 IPP Objects

The term "client" in IPP is intended to mean any client that issues IPP operation requests and accepts IPP operation responses, whether it be a desktop or a server. In other words, the term "client" does not just mean end-user clients, such as those associated with desktops.

The term "IPP Printer" in IPP is intended to mean an object that accepts IPP operation requests and returns IPP operation responses, whether implemented in a server or a device. An IPP Printer object MAY, if implemented in a server, turn around and forward received jobs (and other requests) to other devices and print servers/services, either using IPP or some other protocol.

3 IPP Operations

This section corresponds to Section 3 "IPP Operations" in the IPP/1.1 Model and Semantics document [RFC2911].

3.1 Common Semantics

This section discusses semantics common to all operations.

3.1.1 Summary of Operation Attributes

Table 1 - Summary of Printer operation attributes that sender MUST supply

Operation Attributes	Printer Operations						
	Requests						Responses
	PJ, VJ (R)	PU (O)	CJ (O)	GPA (R)	GJ (R)	PP, RP, PP (O+)	All Operations
Operation parameters--REQUIRED to be supplied by the sender:							
operation-id	R	R	R	R	R	R	
status-code							R
request-id	R	R	R	R	R	R	R
version-number	R	R	R	R	R	R	R
Operation attributes--REQUIRED to be supplied by the sender:							
attributes-charset	R	R	R	R	R	R	R
attributes-natural-language	R	R	R	R	R	R	R
document-uri		R					
job-id*							
job-uri*							
last-document							
printer-uri	R	R	R	R	R	R	
Operation attributes--RECOMMENDED to be supplied by the sender:							
job-name	R	R	R				
requesting-user-name	R	R	R	R	R	R	

Legend:
PJ, VJ: Print-Job, Validate-Job
PU: Print-URI
CJ: Create-Job
GPA: Get-Printer-Attributes
GJ: Get-Jobs
PP, RP, PP: Pause-Printer, Resume-Printer, Purge-Printer

R	indicates a REQUIRED operation that MUST be supported by the IPP object (Printer or Job). For attributes, R indicates that the attribute MUST be supported by the IPP object that supports the associated operation.
O	indicates an OPTIONAL operation or attribute that MAY be supported by the IPP object (Printer or Job).
+	indicates that this is not an IPP/1.0 feature, but is only a part of IPP/1.1 and future versions of IPP.

Table 2 - Summary of Printer operation attributes that sender MAY supply

Operation Attributes	Printer Operations						
	Requests						Responses
	PJ, VJ (R)	PU (O)	CJ (O)	GPA (R)	GJ (R)	PP, RP, PP (O+)	All Operations
Operation attributes--OPTIONAL to be supplied by the sender:							
status-message							O
detailed-status-message							O
document-access-error							O**
compression	R~~O~~	R~~O~~					
document-format	R	R		R			
document-name	O	O					
document-natural-language	O	O					
ipp-attribute-fidelity	R	R	R				
job-impressions	O	O	O				
job-k-octets	O	O	O				
job-media-sheets	O	O	O				
limit					R		
message							
my-jobs					R		
requested-attributes				R	R		
which-jobs					R		

Legend:
PJ, VJ: Print-Job, Validate-Job
PU: Print-URI
CJ: Create-Job
GPA: Get-Printer-Attributes
GJ: Get-Jobs
PP, RP, PP: Pause-Printer, Resume-Printer, Purge-Printer
R indicates a REQUIRED operation that MUST be supported by the IPP object (Printer or Job). For attributes, R indicates that the attribute MUST be supported by the IPP object that supports the associated operation.
O indicates an OPTIONAL operation or attribute that MAY be supported by the IPP object (Printer or Job).
+ indicates that this is not an IPP/1.0 feature, but is only a part of IPP/1.1 and future versions of IPP.
* "job-id" is REQUIRED only if used together with "printer-uri" to identify the target job; otherwise, "job-uri" is REQUIRED.
** "document-access-error" applies to the Print-URI response only.

Table 3 - Summary of Job operation attributes that sender MUST supply

	Job Operations					
	Requests					Responses
Operation Attributes	SD (O)	SU (O)	CJ (R)	GJA (R)	HJ, RJ, RJ (O+)	All Operations
Operation parameters--REQUIRED to be supplied by the sender:						
operation-id	R	R	R	R	R	
status-code						R
request-id	R	R	R	R	R	R
version-number	R	R	R	R	R	R
Operation attributes--REQUIRED to be supplied by the sender:						
attributes-charset	R	R	R	R	R	R
attributes-natural-language	R	R	R	R	R	R
document-uri		R				
job-id*	R	R	R	R	R	
job-uri*	R	R	R	R	R	
last-document	R	R				
printer-uri	R	R	R	R	R	
Operation attributes--RECOMMENDED to be supplied by the sender:						
job-name						
requesting-user-name	R	R	R	R	R	

Legend:

SD: Send-Document

SU: Send-URI

CJ: Cancel-Job

GJA: Get-Job-Attributes

HJ, RJ, RJ: Hold-Job, Release-Job, Restart-Job

R indicates a REQUIRED operation that MUST be supported by the IPP object (Printer or Job). For attributes, R indicates that the attribute MUST be supported by the IPP object that supports the associated operation.

O indicates an OPTIONAL operation or attribute that MAY be supported by the IPP object (Printer or Job).

+ indicates that this is not an IPP/1.0 feature, but is only a part of IPP/1.1 and future versions of IPP.

* "job-id" is REQUIRED only if used together with "printer-uri" to identify the target job; otherwise, "job-uri" is REQUIRED.

Table 4 - Summary of Job operation attributes that sender MAY supply

	Job Operations						
	Requests						Responses
Operation Attributes	SD (O)	SU (O)	CJ (R)	GJA (R)	HJ, RJ, RJ (O+)	SD (O)	All Operations
Operation attributes--OPTIONAL to be supplied by the sender:							
status-message							O
detailed-status-message							O
document-access-error							O**
compression	R~~O~~	R~~O~~					
document-format	R	R					
document-name	O	O					
document-natural-language	O	O					
ipp-attribute-fidelity							
job-impressions							
job-k-octets							
job-media-sheets							
limit							
message			O		O	O	
job-hold-until					R		
my-jobs							
requested-attributes				R			
which-jobs							

Legend:

SD: Send-Document
SU: Send-URI
CJ: Cancel-Job
GJA: Get-Job-Attributes
HJ, RJ, RJ: Hold-Job, Release-Job, Restart-Job

R indicates a REQUIRED operation that MUST be supported by the IPP object (Printer or Job). For attributes, R indicates that the attribute MUST be supported by the IPP object that supports the associated operation.

O indicates an OPTIONAL operation or attribute that MAY be supported by the IPP object (Printer or Job).

+ indicates that this is not an IPP/1.0 feature, but is only a part of IPP/1.1 and future versions of IPP.

* "job-id" is REQUIRED only if used together with "printer-uri" to identify the target job; otherwise, "job-uri" is REQUIRED.

** "document-access-error" applies to the Send-URI operation only

Table 5 - Printer operation response attributes

	Printer Operations						
	Response						
Operation Attributes	PJ (R) SD (O)	VJ (R)	PU (O) SU (O)	CJ (O)	GPA (R)	GJ (R)	PP, RP, PP (O+)
job-uri	R		R	R			
job-id	R		R	R			
job-state	R		R	R			
job-state-reasons	R+		R+	R+			
number-of-intervening-jobs	O		O	O			
document-access-error+			O				

Legend:
PJ, SJ: Print-Job, Send-Document
VJ: Validate-Job
PU, SU: Print-URI, Send-URI
CJ: Create-Job
GPA: Get-Printer-Attributes
GJ: Get-Jobs
PP, RP, PP: Pause-Printer, Resume-Printer, Purge-Printer
R indicates a REQUIRED operation that MUST be supported by the IPP object (Printer or Job). For attributes, R indicates that the attribute MUST be supported by the IPP object that supports the associated operation.
O indicates an OPTIONAL operation or attribute that MAY be supported by the IPP object (Printer or Job).

3.1.2 Suggested Operation Processing Steps for IPP Objects

This section suggests the steps and error checks that an IPP object MAY perform when processing requests and returning responses. An IPP object MAY perform some or all of the error checks. However, some implementations MAY choose to be more forgiving than the error checks shown here, in order to be able to accept requests from non-conforming clients. Not performing all of these error checks is a so-called "forgiving" implementation. On the other hand, clients that successfully submit requests to IPP objects that do perform all the error checks will be more likely to be able to interoperate with other IPP object implementations. Thus an implementer of an IPP object needs to decide whether to be a "forgiving" or a "strict" implementation. Therefore, the error status codes returned may differ between implementations. Consequentially, client SHOULD NOT expect exactly the error code processing described in this section.

When an IPP object receives a request, the IPP object either accepts or rejects the request. In order to determine whether or not to accept or reject the request, the IPP object SHOULD execute the following steps. The order of the steps may be rearranged and/or combined, including making one or multiple passes over the request.

A client MUST supply requests that would pass all of the error checks indicated here in order to be a conforming client. Therefore, a client SHOULD supply requests that are conforming, in order to avoid being rejected by some IPP object implementations and/or risking different semantics by different implementations of forgiving implementations. For example, a forgiving implementation that accepts multiple occurrences of the same attribute, rather than rejecting the request might use the first occurrences, while another might use the last occurrence. Thus such a non-conforming client would get different results from the two forgiving implementations.

In the following, processing continues step by step until a "RETURNS the xxx status code ..." statement is encountered. Error returns are indicated by the verb: "REJECTS". Since clients have difficulty getting the status code before sending all of the document data in a Print-Job request, clients SHOULD use the Validate-Job operation before sending large documents to be printed, in order to validate whether the IPP Printer will accept the job or not.

It is assumed that security authentication and authorization has already taken place at a lower layer.

3.1.2.1 Suggested Operation Processing Steps for all Operations

This section is intended to apply to all operations. The next section contains the additional steps for the Print-Job, Validate-Job, Print-URI, Create-Job, Send-Document, and Send-URI operations that create jobs, adds documents, and validates jobs.

```
IIG Sect #         Flow                    IPP error status codes
----------         ----                    ----------------------
                     |
                     v          err
3.1.2.1.1   <Validate version>  --> server-error-version-not-
                                    supported
                   ok|
                     v          err
3.1.2.1.2  <Validate operation> --> server-error-operation-not-
                                    supported
                   ok|
                     v          err
3.1.2.1.4.1- <Validate presence> --> client-error-bad-request
3.1.2.1.4.2    <of attributes>
                   ok|
                     v          err
3.1.2.1.4.3 <Validate presence> --> client-error-bad-request
            <of operation attr>
                   ok|
                     v          err
3.1.2.1.5  <Vali~~e~~date values of>  --> client-error-bad-request
            <operation attrs>       client-error-request-value-
                                    too-long
          <(length, tag, range,>
              <multi-value)>
                   ok|
                     v          err
3.1.2.1.5    <Validate values>  --> client-error-bad-request
          <with supported values>   client-error-charset-not-
                                    supported
                   ok|              client-error-attributes-or-
                                    values-
                     |                           not-supported
                     v          err
3.1.2.1.6 <Validate optionally> --> client-error-bad-request
             <operation attr>       client-error-natural-language-
                                    not-supported
                     |              client-error-request-value-
                                    too-long
                     |              client-error-attributes-or-
                                    values-not-supported
```

3.1.2.1.1 Validate version number

Every request and every response contains the "version-number" attribute. The value of this attribute is the major and minor version number of the syntax and semantics that the client and IPP object is using, respectively. The "version-number" attribute remains in a fixed position across all future versions so that all clients and IPP object that support future versions can determine which version is being used. The IPP object checks to see if the major version number supplied in the request is supported. If not, the Printer object REJECTS the request and RETURNS the 'server-error-version-not-supported' status code in the response. The IPP object returns in the "version-number" response attribute the major and minor version for the error response. Thus the client can learn at least one major and minor version that the IPP object supports. The IPP object is encouraged to return the closest version number to the one supplied by the client.

The checking of the minor version number is implementation dependent, however if the client-supplied minor version is explicitly supported, the IPP object MUST respond using that identical minor version number. If the major version number matches, but the minor version number does not, the Printer SHOULD accept and attempt to process the request, or MAY reject the request and return the 'server-error-version-not-supported' status code. In all cases, the Printer MUST return the nearest version number that it supports. For example, suppose that an IPP/1.2 Printer supports versions '1.1' and '1.2'. The following responses are conforming:

Table 6 - Examples of validating IPP version

Client supplies	Printer Accept Request?	Printer returns
1.0	yes (SHOULD)	1.1
1.0	no (SHOULD NOT)	1.1
1.1	yes (MUST)	1.1
1.2	yes (MUST)	1.2
1.3	yes (SHOULD)	1.2
1.3	no (SHOULD NOT)	1.2

It is advantageous for Printers to support both IPP/1.1 and IPP/1.0, so that they can interoperate with either client implementations. Some implementations may allow an Administrator to explicitly disable support for one or the other by setting the "ipp-versions-supported" Printer description attribute.

Likewise, it is advantageous for clients to support both versions to allow interoperability with new and legacy Printers.

3.1.2.1.2 Validate operation identifier

The Printer object checks to see if the "operation-id" attribute supplied by the client is supported as indicated in the Printer object's "operations-supported" attribute. If not, the Printer REJECTS the request and returns the 'server-error-operation-not-supported' status code in the response.

3.1.2.1.3 Validate the request identifier

The Printer object SHOULD NOT check to see if the "request-id" attribute supplied by the client is in range: between 1 and 2**31 - 1 (inclusive), but copies all 32 bits.

Note: The "version-number", "operation-id", and the "request-id" parameters are in fixed octet positions in the IPP/1.1 encoding. The "version-number" parameter will be the same fixed octet position in all versions of the protocol. These fields are validated before proceeding with the rest of the validation.

3.1.2.1.4 Validate attribute group and attribute presence and order

The order of the following validation steps depends on implementation.

3.1.2.1.4.1 Validate the presence and order of attribute groups

Client requests and IPP object responses contain attribute groups that Section 3 requires to be present and in a specified order. An IPP object verifies that the attribute groups are present and in the correct order in requests supplied by clients (attribute groups without an * in the following tables).

If an IPP object receives a request with (1) required attribute groups missing, or (2) the attributes groups are out of order, or (3) the groups are repeated, the IPP object REJECTS the request and RETURNS the 'client-error-bad-request' status code. For example, it is an error for the Job Template Attributes group to occur before the Operation Attributes group, for the Operation Attributes group to be omitted, or for an attribute group to occur more than once, except in the Get-Jobs response.

Since this kind of attribute group error is most likely to be an error detected by a client developer rather than by a customer, the IPP object NEED NOT return an indication of which attribute group was in error in either the Unsupported Attributes group or the Status Message. Also, the IPP object NEED NOT find all attribute group errors before returning this error.

3.1.2.1.4.2 Ignore unknown attribute groups in the expected position

Future attribute groups may be added to the specification at the end of requests just before the Document Content and at the end of response, except for the Get-Jobs response, where it maybe there or before the first job attributes returned. If an IPP object receives an unknown attribute group in these positions, it ignores the entire group, rather than returning an error, since that group may be a new group in a later minor version of the protocol that can be ignored. (If the new attribute group cannot be ignored without confusing the client, the major version number would have been increased in the protocol document and in the request). If the unknown group occurs in a different position, the IPP object REJECTS the request and RETURNS the 'client-error-bad-request' status code.

Clients also ignore unknown attribute groups returned in a response.

Note: By validating that requests are in the proper form, IPP objects force clients to use the proper form which, in turn, increases the chances that customers will be able to use such clients from multiple vendors with IPP objects from other vendors.

3.1.2.1.4.3 Validate the presence of a single occurrence of required Operation attributes

Client requests and IPP object responses contain Operation attributes that [RFC2911] Section 3 requires to be present. Attributes within a group may be in any order, except for the ordering of target, charset, and natural languages attributes. These attributes MUST be first, and MUST be supplied in the following order: charset, natural language, and then target. An IPP object verifies that the attributes that Section 4 requires to be supplied by the client have been supplied in the request (attributes without an * in the following tables). An asterisk (*) indicates groups and Operation attributes that the client may omit in a request or an IPP object may omit in a response.

If an IPP object receives a request with required attributes missing or repeated from a group or in the wrong position, the behavior of the IPP object is IMPLEMENTATION DEPENDENT. Some of the possible implementations are:

REJECTS the request and RETURNS the 'client-error-bad-request' status code

accepts the request and uses the first occurrence of the attribute no matter where it is

accepts the request and uses the last occurrence of the attribute no matter where it is

accept the request and assume some default value for the missing attribute

Therefore, client MUST send conforming requests, if they want to receive the same behavior from all IPP object implementations. For example, it is an error for the "attributes-charset" or "attributes-natural-language" attribute to be omitted in any operation request, or for an Operation attribute to be supplied in a Job Template group or a Job Template attribute to be supplied in an Operation Attribute group in a create request. It is also an error to supply the "attributes-charset" attribute twice.

Since these kinds of attribute errors are most likely to be detected by a client developer rather than by a customer, the IPP object NEED NOT return an indication of which attribute was in error in either the Unsupported Attributes group or the Status Message. Also, the IPP object NEED NOT find all attribute errors before returning this error.

The following tables list all the attributes for all the operations by attribute group in each request and each response. The order of the groups is the order that the client supplies the groups as specified in [RFC2911] Section 3. The order of the attributes within a group is arbitrary, except as noted for some of the special operation attributes (charset, natural language, and target). The tables below use the following notation:

R	indicates a REQUIRED attribute or operation that an IPP object MUST support
O	indicates an OPTIONAL attribute or operation that an IPP object NEED NOT support

* indicates that a client MAY omit the attribute in a request and that an IPP object MAY omit the attribute in a response. The absence of an * means that a client MUST supply the attribute in a request and an IPP object MUST supply the attribute in a response.

+ indicates that this is not a IPP/1.0 operation, but is only a part of IPP/1.1 and future versions of IPP.

Operation Requests

The tables below show the attributes in their proper attribute groups for operation requests:

Note: All operation requests contain "version-number", "operation-id", and "request-id" parameters.

```
Print-Job Request (R):
  Group 1: Operation Attributes (R)
       attributes-charset (R)
       attributes-natural-language (R)
       printer-uri (R)
       requesting-user-name (R*)
       job-name (R*)
       ipp-attribute-fidelity (R*)
       document-name (R*)
       document-format (R*)
       document-natural-language (O*)
       compression (R~~O~~*)
       job-k-octets (O*)
       job-impressions (O*)
       job-media-sheets (O*)
  Group 2: Job Template Attributes (R*)
       <Job Template attributes> (O*)
            (see [RFC2911] Section 4.2)
  Group 3: Document Content (R)
       <document content>

Validate-Job Request (R):
  Group 1: Operation Attributes (R)
       attributes-charset (R)
       attributes-natural-language (R)
       printer-uri (R)
       requesting-user-name (R*)
       job-name (R*)
       ipp-attribute-fidelity (R*)
       document-name (R*)
       document-format (R*)
       document-natural-language (O*)
       compression (R~~O~~*)
       job-k-octets (O*)
       job-impressions (O*)
```

```
          job-media-sheets (O*)
     Group 2: Job Template Attributes (R*)
          <Job Template attributes> (O*)
               (see [RFC2911] Section 4.2)

   Print-URI Request (O):
     Group 1: Operation Attributes (R)
          attributes-charset (R)
          attributes-natural-language (R)
          printer-uri (R)
          document-uri (R)
          requesting-user-name (R*)
          job-name (R*)
          ipp-attribute-fidelity (R*)
          document-name (R*)
          document-format (R*)
          document-natural-language (O*)
          compression (R~~O~~*)
          job-k-octets (O*)
          job-impressions (O*)
          job-media-sheets (O*)
     Group 2: Job Template Attributes (R*)
          <Job Template attributes> (O*) (see
               (see [RFC2911] Section 4.2)

   Create-Job Request (O):
     Group 1: Operation Attributes (R)
          attributes-charset (R)
          attributes-natural-language (R)
          printer-uri (R)
          requesting-user-name (R*)
          job-name (R*)
          ipp-attribute-fidelity (R*)
          job-k-octets (O*)
          job-impressions (O*)
          job-media-sheets (O*)
     Group 2: Job Template Attributes (R*)
          <Job Template attributes> (O*) (see
               (see [RFC2911] Section 4.2)

   Get-Printer-Attributes Request (R):
     Group 1: Operation Attributes (R)
          attributes-charset (R)
          attributes-natural-language (R)
          printer-uri (R)
          requesting-user-name (R*)
          requested-attributes (R*)
          document-format (R*)

   Get-Jobs Request (R):
     Group 1: Operation Attributes (R)
          attributes-charset (R)
          attributes-natural-language (R)
          printer-uri (R)
          requesting-user-name (R*)
          limit (R*)
          requested-attributes (R*)
          which-jobs (R*)
          my-jobs (R*)

   Send-Document Request (O):
     Group 1: Operation Attributes (R)
          attributes-charset (R)
          attributes-natural-language (R)
          (printer-uri & job-id) | job-uri (R)
          last-document (R)
          requesting-user-name (R*)
          document-name (R*)
          document-format (R*)
          document-natural-language (O*)
          compression (R~~O~~*)
     Group 2: Document Content (R*)
          <document content>

   Send-URI Request (O):
     Group 1: Operation Attributes (R)
          attributes-charset (R)
          attributes-natural-language (R)
          (printer-uri & job-id) | job-uri (R)
          last-document (R)
          document-uri (R)
          requesting-user-name (R*)
          document-name (R*)
          document-format (R*)
          document-natural-language (O*)
          compression (R~~O~~*)

   Cancel-Job Request (R):
   Release-Job Request (O+):
     Group 1: Operation Attributes (R)
          attributes-charset (R)
          attributes-natural-language (R)
          (printer-uri & job-id) | job-uri (R)
          requesting-user-name (R*)
          message (O*)

   Get-Job-Attributes Request (R):
     Group 1: Operation Attributes (R)
          attributes-charset (R)
          attributes-natural-language (R)
```

```
       (printer-uri & job-id) | job-uri (R)
       requesting-user-name (R*)
       requested-attributes (R*)

Pause-Printer Request (O+):
Resume-Printer Request (O+):
Purge-Printer Request (O+):
  Group 1: Operation Attributes (R)
       attributes-charset (R)
       attributes-natural-language (R)
       printer-uri (R)
       requesting-user-name (R*)

Hold-Job Request (O+):
Restart-Job Request (O+):
  Group 1: Operation Attributes (R)
       attributes-charset (R)
       attributes-natural-language (R)
       (printer-uri & job-id) | job-uri (R)
       requesting-user-name (R*)
       job-hold-until (R*)
       message (O*)
```

Operation Responses

The tables below show the response attributes in their proper attribute groups for responses.

Note: All operation responses contain "version-number", "status-code", and "request-id" parameters.

```
Print-Job Response (R):
Create-Job Response (O):
Send-Document Response (O):
  Group 1: Operation Attributes (R)
       attributes-charset (R)
       attributes-natural-language (R)
       status-message (O*)
       detailed-status-message (O*)
  Group 2: Unsupported Attributes (R*) (see Note 3)
       <unsupported attributes> (R*)
  Group 3: Job Object Attributes(R*) (see Note 2)
       job-uri (R)
       job-id (R)
       job-state (R)
       job-state-reasons (O* | R+)
       job-state-message (O*)
       number-of-intervening-jobs (O*)

Validate-Job Response (R):
```

```
Cancel-Job Response (R):
Hold-Job Response (O+):
Release-Job Response (O+):
Restart-Job Response (O+):
  Group 1: Operation Attributes (R)
       attributes-charset (R)
       attributes-natural-language (R)
       status-message (O*)
       detailed-status-message (O*)
  Group 2: Unsupported Attributes (R*) (see Note 3)
       <unsupported attributes> (R*)

Print-URI Response (O):
Send-URI Response (O):
  Group 1: Operation Attributes (R)
       attributes-charset (R)
       attributes-natural-language (R)
       status-message (O*)
       detailed-status-message (O*)
       document-access-error (O*)
  Group 2: Unsupported Attributes (R*) (see Note 3)
       <unsupported attributes> (R*)
  Group 3: Job Object Attributes(R*) (see Note 2)
       job-uri (R)
       job-id (R)
       job-state (R)
       job-state-reasons (O* | R+)
       job-state-message (O*)
       number-of-intervening-jobs (O*)

Get-Printer-Attributes Response (R):
  Group 1: Operation Attributes (R)
       attributes-charset (R)
       attributes-natural-language (R)
       status-message (O*)
       detailed-status-message (O*)
  Group 2: Unsupported Attributes (R*) (see Note 4)
       <unsupported attributes> (R*)
  Group 3: Printer Object Attributes(R*) (see Note 2)
       <requested attributes> (R*)

Get-Jobs Response (R):
  Group 1: Operation Attributes (R)
       attributes-charset (R)
       attributes-natural-language (R)
       status-message (O*)
       detailed-status-message (O*)
  Group 2: Unsupported Attributes (R*) (see Note 4)
       <unsupported attributes> (R*)
  Group 3: Job Object Attributes(R*) (see Note 2, 5)
```

```
            <requested attributes> (R*)

     Get-Job-Attributes Response (R):
       Group 1: Operation Attributes (R)
            attributes-charset (R)
            attributes-natural-language (R)
            status-message (O*)
            detailed-status-message (O*)
       Group 2: Unsupported Attributes (R*) (see Note 4)
            <unsupported attributes> (R*)
       Group 3: Job Object Attributes(R*) (see Note 2)
            <requested attributes> (R*)

     Pause-Printer Response (O+):
     Resume-Printer Response (O+):
     Purge-Printer Response (O+):
       Group 1: Operation Attributes (R)
            attributes-charset (R)
            attributes-natural-language (R)
            status-message (O*)
            detailed-status-message (O*)
       Group 2: Unsupported Attributes (R*) (see Note 4)
            <unsupported attributes> (R*)
```

Note 2 - the Job Object Attributes and Printer Object Attributes are returned only if the IPP object returns one of the success status codes.

Note 3 - the Unsupported Attributes Group is present only if the client included some Operation and/or Job Template attributes or values that the Printer doesn't support whether a success or an error return.

Note 4 - the Unsupported Attributes Group is present only if the client included some Operation attributes that the Printer doesn't support whether a success or an error return.

Note 5: for the Get-Jobs operation the response contains a separate Job Object Attributes group 3 to N containing requested-attributes for each job object in the response.

3.1.2.1.5 Validate the values of the REQUIRED Operation attributes

An IPP object validates the values supplied by the client of the REQUIRED Operation attribute that the IPP object MUST support. The next section specifies the validation of the values of the OPTIONAL Operation attributes that IPP objects MAY support.

The IPP object performs the following syntactic validation checks of each Operation attribute value:

a) that the length of each Operation attribute value is correct for the attribute syntax tag supplied by the client according to [RFC2911] Section 4.1,

b) that the attribute syntax tag is correct for that Operation attribute according to [RFC2911] Section 3,

c) that the value is in the range specified for that Operation attribute according to [RFC2911] Section 3,

d) that multiple values are supplied by the client only for operation attributes that are multi-valued, i.e., that are 1setOf X according to [RFC2911] Section 3.

If any of these checks fail, the IPP object REJECTS the request and RETURNS the 'client-error-bad-request' or the 'client-error-request-value-too-long' status code. Since such an error is most likely to be an error detected by a client developer, rather than by an end-user, the IPP object NEED NOT return an indication of which attribute had the error in either the Unsupported Attributes Group or the Status Message. The description for each of these syntactic checks is explicitly expressed in the first IF statement in the following table.

In addition, the IPP object checks each Operation attribute value against some Printer object attribute or some hard-coded value if there is no "xxx-supported" Printer object attribute defined. If its value is not among those supported or is not in the range supported, then the IPP object REJECTS the request and RETURNS the error status code indicated in the table by the second IF statement. If the value of the Printer object's "xxx-supported" attribute is 'no-value' (because the system administrator hasn't configured a value), the check always fails.

----

attributes-charset (charset)

```
    IF NOT a single non-empty 'charset' value, REJECT/RETURN 'client-error-bad-request'.
    IF the value length is greater than 63 octets, REJECT/RETURN 'client-error-request-value-too-
        long'.
    IF NOT in the Printer object's "charset-supported" attribute, REJECT/RETURN "client-error-
        charset-not-supported".
```

attributes-natural-language(naturalLanguage)

```
    IF NOT a single non-empty 'naturalLanguage' value, REJECT/RETURN 'client-error-bad-
        request'.
    IF the value length is greater than 63 octets, REJECT/RETURN 'client-error-request-value-too-
        long'.
    ACCEPT the request even if not a member of the set in the Printer object's "generated-natural-
        language-supported" attribute.  If the supplied value is not a member of the Printer
        object's "generated-natural-language-supported" attribute, use the Printer object's
        "natural-language-configured" value.
```

requesting-user-name

IF NOT a single 'name' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.

IF the IPP object can obtain a better-authenticated name, use it instead.

job-name(name)

IF NOT a single 'name' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.

IF NOT supplied by the client, the Printer object creates a name from the document-name or document-uri.

document-name (name)

IF NOT a single 'name' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.

ipp-attribute-fidelity (boolean)

IF NEITHER a single 'true' NOR a single 'false' 'boolean' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is NOT equal to 1 octet, REJECT/RETURN 'client-error-request-value-too-long'

IF NOT supplied by the client, the IPP object assumes the value 'false'.

document-format (mimeMediaType)

IF NOT a single non-empty 'mimeMediaType' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.

IF NOT in the Printer object's "document-format-supported" attribute, REJECT/RETURN 'client-error-document-format-not-supported'

IF NOT supplied by the client, the IPP object assumes the value of the Printer object's "document-format-default" attribute.

document-uri (uri)

IF NOT a single non-empty 'uri' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is greater than 1023 octets, REJECT/RETURN 'client-error-request-value-too-long'.

IF the URI syntax is not valid, REJECT/RETURN 'client-error-bad-request'.

If the client-supplied URI scheme is not supported, i.e. the value is not in the Printer object's referenced-uri-scheme-supported" attribute, the Printer object MUST reject the request and return the 'client-error-uri-scheme-not-supported' status code. The Printer object MAY check to see if the document exists and is accessible. If the document is not found or is not accessible, REJECT/RETURN 'client-error-not found'.

last-document (boolean)

IF NEITHER a single 'true' NOR a single 'false' 'boolean' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is NOT equal to 1 octet, REJECT/RETURN 'client-error-request-value-too-long'

job-id (integer(1:MAX))

IF NOT an single 'integer' value equal to 4 octets AND in the range 1 to MAX, REJECT/RETURN 'client-error-bad-request'.

IF NOT a job-id of an existing Job object, REJECT/RETURN 'client-error-not-found' or 'client-error-gone' status code, if keep track of recently deleted jobs.

requested-attributes (1setOf keyword)

IF NOT one or more 'keyword' values, REJECT/RETURN 'client-error-bad-request'.

IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.

Ignore unsupported values, which are the keyword names of unsupported attributes. Don't bother to copy such requested (unsupported) attributes to the Unsupported Attribute response group since the response will not return them.

which-jobs (type2 keyword)

IF NOT a single 'keyword' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.

IF NEITHER 'completed' NOR 'not-completed', copy the attribute and the unsupported value to the Unsupported Attributes response group and REJECT/RETURN 'client-error-attributes-or-values-not-supported'.

Note: a Printer still supports the 'completed' value even if it keeps no completed/canceled/aborted jobs: by returning no jobs when so queried.

IF NOT supplied by the client, the IPP object assumes the 'not-completed' value.

my-jobs (boolean)

IF NEITHER a single 'true' NOR a single 'false' 'boolean' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is NOT equal to 1 octet, REJECT/RETURN 'client-error-request-value-too-long'

IF NOT supplied by the client, the IPP object assumes the 'false' value.

limit (integer(1:MAX))

IF NOT a single 'integer' value equal to 4 octets AND in the range 1 to MAX,
REJECT/RETURN 'client-error-bad-request'.
IF NOT supplied by the client, the IPP object returns all jobs, no matter how many.

3.1.2.1.6 Validate the values of the OPTIONAL Operation attributes

OPTIONAL Operation attributes are those that an IPP object MAY support. An IPP object validates the values of the OPTIONAL attributes supplied by the client. The IPP object performs the same syntactic validation checks for each OPTIONAL attribute value as in Section 3.1.2.1.5. As in Section 3.1.2.1.5, if any fail, the IPP object REJECTS the request and RETURNS the 'client-error-bad-request' or the 'client-error-request-value-too-long' status code.

In addition, the IPP object checks each Operation attribute value against some Printer attribute or some hard-coded value if there is no "xxx-supported" Printer attribute defined. If its value is not among those supported or is not in the range supported, then the IPP object REJECTS the request and RETURNS the error status code indicated in the table. If the value of the Printer object's "xxx-supported" attribute is 'no-value' (because the system administrator hasn't configured a value), the check always fails.

If the IPP object doesn't recognize/support an attribute, the IPP object treats the attribute as an unknown or unsupported attribute (see the last row in the table below).

----

document-natural-language (naturalLanguage)

IF NOT a single non-empty 'naturalLanguage' value, REJECT/RETURN 'client-error-bad-request'.
IF the value length is greater than 63 octets, REJECT/RETURN 'client-error-request-value-too-long'.
IF NOT a value that the Printer object supports in document formats, (no corresponding "xxx-supported" Printer attribute), REJECT/RETURN 'client-error-natural-language-not-supported'.

compression (type3 keyword)

IF NOT a single 'keyword' value, REJECT/RETURN 'client-error-bad-request'.
IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.
IF NOT in the Printer object's "compression-supported" attribute, ~~copy the attribute and the unsupported value to the Unsupported Attributes response group and~~ REJECT/RETURN 'client-error-~~attributes-or-values~~compression-not-supported'.

Note to IPP/1.0 implementers: Support for the "compression" attribute was optional in IPP/1.0 and was changed to REQUIRED in IPP/1.1. However, an IPP/1.0 object SHOULD at least check for the "compression" attribute being present and reject the create request, if they don't support "compression". Not checking is a bug, since the data will be unintelligible.

job-k-octets (integer(0:MAX))

IF NOT a single 'integer' value equal to 4 octets, REJECT/RETURN 'client-error-bad-request'.

IF NOT in the range of the Printer object's "job-k-octets-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group and REJECT/RETURN 'client-error-attributes-or-values-not-supported'.

job-impressions (integer(0:MAX))

IF NOT a single 'integer' value equal to 4 octets, REJECT/RETURN 'client-error-bad-request'.

IF NOT in the range of the Printer object's "job-impressions-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group and REJECT/RETURN 'client-error-attributes-or-values-not-supported'.

job-media-sheets (integer(0:MAX))

IF NOT a single 'integer' value equal to 4 octets, REJECT/RETURN 'client-error-bad-request'.

IF NOT in the range of the Printer object's "job-media-sheets-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group and REJECT/RETURN 'client-error-attributes-or-values-not-supported'.

message (text(127))

IF NOT a single 'text' value, REJECT/RETURN 'client-error-bad-request'.

IF the value length is greater than 127 octets, REJECT/RETURN 'client-error-request-value-too-long'.

unknown or unsupported attribute

IF the attribute syntax supplied by the client is supported but the length is not legal for that attribute syntax, REJECT/RETURN 'client-error-request-value-too-long'.

ELSE copy the attribute and value to the Unsupported Attributes response group and change the attribute value to the "out-of-band" 'unsupported' value, but otherwise ignore the attribute.

Note: Future Operation attributes may be added to the protocol specification that may occur anywhere in the specified group. When the operation is otherwise successful, the IPP object returns the 'successful-ok-ignored-or-substituted-attributes' status code. Ignoring unsupported Operation attributes in all operations is analogous to the handling of unsupported Job Template attributes in the create and Validate-Job operations when the client supplies the "ipp-attribute-fidelity" Operation attribute with the 'false' value. This last rule is so that we can add OPTIONAL Operation attributes to future versions of IPP so that older clients can inter-work with new IPP objects and newer clients can inter-work with older IPP objects. (If the new attribute cannot be ignored without performing unexpectedly, the major version number would have been increased in the protocol document and in the request). This rule for Operation attributes is independent of the value of the "ipp-attribute-fidelity" attribute. For example, if an IPP object doesn't support the OPTIONAL "job-k-octets" attribute', the IPP object treats "job-k-octets" as an unknown attribute and only checks the length for the 'integer' attribute syntax supplied by the client. If it is not four octets, the IPP object REJECTS the request and RETURNS the 'client-error-bad-request' status code, else the IPP object copies the attribute to the Unsupported Attribute response group, setting the value to the "out-of-band" 'unsupported' value, but otherwise ignores the attribute.

3.1.2.2 Suggested Additional Processing Steps for Operations that Create/Validate Jobs and Add Documents

This section in combination with the previous section recommends the processing steps for the Print-Job, Validate-Job, Print-URI, Create-Job, Send-Document, and Send-URI operations that IPP objects SHOULD use. These are the operations that create jobs, validate a Print-Job request, and add documents to a job.

```
IIG Sect #         Flow                 IPP error status codes
----------         ----                 ----------------------
                     |
                     v             No
3.1.2.2.1 <ipp-attribute-fidelity> ------------------+
               <supplied?>                           |
                  Yes|                               |
                     |  ipp-attribute-fidelity = no  |
                     |<------------------------------+
                     v          No
3.1.2.2.2       <Printer is>    --> server-error-not-accepting-jobs
             <accepting jobs?>
                  Yes|
                     v          err
3.1.2.3    <Validate values of> --> client-error-bad-request
        <Job template attributes>   client-error-request-value-too-
                                    long
         <(length, tag, range,>
              <multi-value)>
                   ok|
                     v          err
3.1.2.3  <Validate values with> --> client-error-bad-request
          <supported values>        client-error-attributes-or-
                     |              values-not-supported
                     v          err
3.1.2.3.1   <Any conflicting>   --> client-error-conflicting-
                                    attributes
       <Job Template attr values>   client-error-attributes-or-
                                    values-not-supported
                     v
```

3.1.2.2.1 Default "ipp-attribute-fidelity" if not supplied

The Printer object checks to see if the client supplied an "ipp-attribute-fidelity" Operation attribute. If the attribute is not supplied by the client, the IPP object assumes that the value is 'false'.

3.1.2.2.2 Check that the Printer object is accepting jobs

If the value of the Printer objects "printer-is-accepting-jobs" is 'false', the Printer object REJECTS the request and RETURNS the 'server-error-not-accepting-jobs' status code.

3.1.2.2.3 Validate the values of the Job Template attributes

An IPP object validates the values of all Job Template attribute supplied by the client. The IPP object performs the analogous syntactic validation checks of each Job Template attribute value that it performs for Operation attributes (see Section 3.1.2.1.5.):

a) that the length of each value is correct for the attribute syntax tag supplied by the client according to [RFC2911] Section 4.1.

b) that the attribute syntax tag is correct for that attribute according to [RFC2911] Sections 4.2 to 4.4.

c) that multiple values are supplied only for multi-valued attributes, i.e., that are 1setOf X according to [RFC2911] Sections 4.2 to 4.4.

As in Section 3.1.2.1.5, if any of these syntactic checks fail, the IPP object REJECTS the request and RETURNS the 'client-error-bad-request' or 'client-error-request-value-too-long' status code as appropriate, independent of the value of the "ipp-attribute-fidelity". Since such an error is most likely to be an error detected by a client developer, rather than by an end-user, the IPP object NEED NOT return an indication of which attribute had the error in either the Unsupported Attributes Group or the Status Message. The description for each of these syntactic checks is explicitly expressed in the first IF statement in the following table.

Each Job Template attribute MUST occur no more than once. If an IPP Printer receives a create request with multiple occurrences of a Job Template attribute, it MAY:

1. reject the operation and return the 'client-error-bad-request' error status code

2. accept the operation and use the first occurrence of the attribute

3. accept the operation and use the last occurrence of the attribute

depending on implementation. Therefore, clients MUST NOT supply multiple occurrences of the same Job Template attribute in the Job Attributes group in the request.

3.1.2.3 Algorithm for job validation

The process of validating a Job-Template attribute "xxx" against a Printer attribute "xxx-supported" can use the following validation algorithm (see section 3.2.1.2 in [RFC2911]).

To validate the value U of Job-Template attribute "xxx" against the value V of Printer "xxx-supported", perform the following algorithm:

1. If U is multi-valued, validate each value X of U by performing the algorithm in Table 7 with each value X. Each validation is separate from the standpoint of returning unsupported values. Example: If U is "finishings" that the client supplies with 'staple', 'bind' values, then X takes on the successive values: 'staple', then 'bind'

2. If V is multi-valued, validate X against each Z of V by performing the algorithm in Table 7 with each value Z. If a value Z validates, the validation for the attribute value X succeeds. If it fails, the algorithm is applied to the next value Z of V. If there are no more values Z of V, validation fails. Example" If V is "sides-supported" with values: 'one-sided', 'two-sided-long', and 'two-sided-short', then Z takes on the successive values: 'one-sided', 'two-sided-long', and 'two-sided-short'. If the client supplies "sides" with 'two-sided-long', the first comparison fails ('one-sided' is not equal to 'two-sided-long'), the second comparison succeeds ('two-sided-long' is equal to 'two-sided-long"), and the third comparison ('two-sided-short' with 'two-sided-long') is not even performed.

3. If both U and V are single-valued, let X be U and Z be V and use the validation rules in Table 7.

Table 7 - Rules for validating single values X against Z

Attribute syntax of X	attribute syntax of Z	validated if:
integer	rangeOfInteger	X is within the range of Z
uri	uriScheme	the uri scheme in X is equal to Z
any	boolean	the value of Z is TRUE
any	any	X and Z are of the same type and are equal.

If the value of the Printer object's "xxx-supported" attribute is 'no-value' (because the system administrator hasn't configured a value), the check always fails. If the check fails, the IPP object copies the attribute to the Unsupported Attributes response group with its unsupported value. If the attribute contains more than one value, each value is checked and each unsupported value is separately copied, while supported values are not copied. If an IPP object doesn't recognize/support a Job Template attribute, i.e., there is no corresponding Printer object "xxx-supported" attribute, the IPP object treats the attribute as an unknown or unsupported attribute (see the last row in the table below).

If some Job Template attributes are supported for some document formats and not for others or the values are different for different document formats, the IPP object SHOULD take that into account in this validation using the value of the "document-format" supplied by the client (or defaulted to the value of the Printer's "document-format-default" attribute, if not supplied by the client). For example, if "number-up" is supported for the 'text/plain' document format, but not for the 'application/postscript' document format, the check SHOULD (though it NEED NOT) depend on the value of the "document-format" operation attribute. See "document-format" in [RFC2911] section 3.2.1.1 and 3.2.5.1.

Note: whether the request is accepted or rejected is determined by the value of the "ipp-attribute-fidelity" attribute in a subsequent step, so that all Job Template attribute supplied are examined and all unsupported attributes and/or values are copied to the Unsupported Attributes response group.

----

job-priority (integer(1:100))

IF NOT a single 'integer' value with a length equal to 4 octets, REJECT/RETURN 'client-error-bad-request'.
IF NOT supplied by the client, use the value of the Printer object's "job-priority-default" attribute at job submission time.
IF NOT in the range 1 to 100, inclusive, copy the attribute and the unsupported value to the Unsupported Attributes response group.
Map the value to the nearest supported value in the range 1:100 as specified by the number of discrete values indicated by the value of the Printer's "job-priority-supported" attribute. See the formula in [RFC2911] Section 4.2.1.

job-hold-until (type3 keyword | name)

IF NOT a single 'keyword' or 'name' value, REJECT/RETURN 'client-error-bad-request'.
IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.
IF NOT supplied by the client, use the value of the Printer object's "job-hold-until" attribute at job submission time.
IF NOT in the Printer object's "job-hold-until-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group.

job-sheets (type3 keyword | name)

IF NOT a single 'keyword' or 'name' value, REJECT/RETURN 'client-error-bad-request'.
IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.
IF NOT in the Printer object's "job-sheets-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group.

multiple-document-handling (type2 keyword)

IF NOT a single 'keyword' value, REJECT/RETURN 'client-error-bad-request'.
IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.
IF NOT in the Printer object's "multiple-document-handling-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group.

copies (integer(1:MAX))

```
    IF NOT a single 'integer' value with a length equal to 4 octets,
    REJECT/RETURN 'client-error-bad-request'.
    IF NOT in range of the Printer object's "copies-supported" attribute
    copy the attribute and the unsupported value to the Unsupported Attributes response group.

finishings (1setOf type2 enum)

    IF NOT an 'enum' value(s) each with a length equal to 4 octets, REJECT/RETURN 'client-error-bad-request'.
    IF NOT in the Printer object's "finishings-supported" attribute, copy the attribute and the unsupported value(s), but not any supported values, to the Unsupported Attributes response group.

page-ranges (1setOf  rangeOfInteger(1:MAX))

    IF NOT a 'rangeOfInteger' value(s) each with a length equal to 8 octets, REJECT/RETURN 'client-error-bad-request'.
    IF first value is greater than second value in any range, the ranges are not in ascending order, or ranges overlap, REJECT/RETURN 'client-error-bad-request'.
    IF the value of the Printer object's "page-ranges-supported" attribute is 'false', copy the attribute to the Unsupported Attributes response group and set the value to the "out-of-band" 'unsupported' value.

sides (type2 keyword)

    IF NOT a single 'keyword' value, REJECT/RETURN 'client-error-bad-request'.
    IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.
    IF NOT in the Printer object's "sides-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group.

number-up (integer(1:MAX))

    IF NOT a single 'integer' value with a length equal to 4 octets,
    REJECT/RETURN 'client-error-bad-request'.
    IF NOT a value or in the range of one of the values of the Printer object's "number-up-supported" attribute, copy the attribute and value to the Unsupported Attribute response group.

orientation-requested (type2 enum)

    IF NOT a single 'enum' value with a length equal to 4 octets,
    REJECT/RETURN 'client-error-bad-request'.
    IF NOT in the Printer object's "orientation-requested-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group.

media (type3 keyword | name)
```

```
    IF NOT a single 'keyword' or 'name' value, REJECT/RETURN 'client-error-bad-request'.
    IF the value length is greater than 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.
    IF NOT in the Printer object's "media-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group.
```

printer-resolution (resolution)

```
    IF NOT a single 'resolution' value with a length equal to 9 octets,
    REJECT/RETURN 'client-error-bad-request'.
    IF NOT in the Printer object's "printer-resolution-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group.
```

print-quality (type2 enum)

```
    IF NOT a single 'enum' value with a length equal to 4 octets,
    REJECT/RETURN 'client-error-bad-request'.
    IF NOT in the Printer object's "print-quality-supported" attribute, copy the attribute and the unsupported value to the Unsupported Attributes response group.
```

unknown or unsupported attribute (i.e., there is no corresponding Printer object "xxx-supported" attribute)

```
    IF the attribute syntax supplied by the client is supported but the length is not legal for that attribute syntax,
    REJECT/RETURN 'client-error-bad-request' if the length of the attribute syntax is fixed or 'client-error-request-value-too-long' if the length of the attribute syntax is variable.
    ELSE copy the attribute and value to the Unsupported Attributes response group and change the attribute value to the "out-of-band" 'unsupported' value.  Any remaining Job Template Attributes are either unknown or unsupported Job Template attributes and are validated algorithmically according to their attribute syntax for proper length (see below).
```

If the attribute syntax is supported AND the length check fails, the IPP object REJECTS the request and RETURNS the 'client-error-bad-request' if the length of the attribute syntax is fixed or the 'client-error-request-value-too-long' status code if the length of the attribute syntax is variable. Otherwise, the IPP object copies the unsupported Job Template attribute to the Unsupported Attributes response group and changes the attribute value to the "out-of-band" 'unsupported' value. The following table shows the length checks for all attribute syntaxes. In the following table: "<=" means less than or equal, "=" means equal to:

Name	Octet length check for read-write attributes
'textWithLanguage	<= 1023 AND 'naturalLanguage' <= 63
'textWithoutLanguage'	<= 1023
'nameWithLanguage'	<= 255 AND 'naturalLanguage' <= 63
'nameWithoutLanguage'	<= 255

'keyword'	<= 255
'enum'	= 4
'uri'	<= 1023
'uriScheme'	<= 63
'charset'	<= 63
'naturalLanguage'	<= 63
'mimeMediaType'	<= 255
'octetString'	<= 1023
'boolean'	= 1
'integer'	= 4
'rangeOfInteger'	= 8
'dateTime'	= 11
'resolution'	= 9
'1setOf X'	

Note: It's possible for a Printer to receive a zero length keyword in a request. Since this is a keyword, its value needs to be compared with the supported values. Assuming that the printer doesn't have any values in its corresponding "xxx-supported" attribute that are keywords of zero length, the comparison will fail. Then the request will be accepted or rejected depending on the value of "ipp-attributes-fidelity" being 'false' or 'true', respectively. No special handling is required for

3.1.2.3.1 Check for conflicting Job Template attributes values

Once all the Operation and Job Template attributes have been checked individually, the Printer object SHOULD check for any conflicting values among all the supported values supplied by the client. For example, a Printer object might be able to staple and to print on transparencies, however due to physical stapling constraints, the Printer object might not be able to staple transparencies. The IPP object copies the supported attributes and their conflicting attribute values to the Unsupported Attributes response group. The Printer object only copies over those attributes that the Printer object either ignores or substitutes in order to resolve the conflict, and it returns the original values which were supplied by the client. For example suppose the client supplies "finishings" equals 'staple' and "media" equals 'transparency', but the Printer object does not support stapling transparencies. If the Printer chooses to ignore the stapling request in order to resolve the conflict, the Printer objects returns "finishings" equal to 'staple' in the Unsupported Attributes response group. If any attributes are multi-valued, only the conflicting values of the attributes are copied.

Note: The decisions made to resolve the conflict (if there is a choice) is implementation dependent.

3.1.2.3.2 Decide whether to REJECT the request

If there were any unsupported Job Template attributes or unsupported/conflicting Job Template attribute values and the client supplied the "ipp-attribute-fidelity" attribute with the 'true' value, the Printer object REJECTS the request and return the status code:

1. 'client-error-conflicting-attributes' status code, if there were any conflicts between attributes supplied by the client.

2. 'client-error-attributes-or-values-not-supported' status code, otherwise.

Note: Unsupported Operation attributes or values that are returned do not affect the status returned in this step. If the unsupported Operation attribute was a serious error, the above already rejected the request in a previous step. If control gets to this step with unsupported Operation attributes being returned, they are not serious errors.

In general, the final results of Job processing are unknown at Job submission time. The client has to rely on notifications or polling to find out what happens at Job processing time. However, there are cases in which some Printers can determine at Job submission time that Job processing is going to fail. As an optimization, we'd like to have the Printer reject the Job in these cases.

There are three types of "processing" errors that might be detectable at Job submission time:

1. 'client-error-document-format-not-supported' : For the Print-Job, Send-Document, Print-URI, and Send-URI operations, if all these conditions are true:

- the Printer supports auto-sensing,
- the request "document-format" operation attribute is 'application/octet-stream',
- the Printer receives document data before responding,
- the Printer auto-senses the document format before responding,
- the sensed document format is not supported by the Printer

then the Printer should respond with 'client-error-document-format-not-supported' status.

2. 'client-error-compression-error': For the Print-Job, Send-Document, Print-URI, and Send-URI operations, if all these conditions are true:

- the client supplies a supported value for the "compression" operation attribute in the request
- the Printer receives document data before responding,
- the Printer attempts to decompress the document data before responding,
- the document data cannot be decompressed using the algorithm specified by the "compression" operation attribute

then the Printer should respond with 'client-error-compression-error' status.

3. 'client-error-document-access-error': For the Print-URI, and Send-URI operations, if the Printer attempts and fails to pull the referenced document data before responding, it should respond with 'client-error-document-access-error' status.

Some Printers are not able to detect these errors until Job processing time. In that case, the errors are recorded in the corresponding job-state and job-state reason attributes. (There is no standard way for a client to determine whether a Printer can detect these errors at Job submission time.) For example, if auto-sensing happens AFTER the job is accepted (as opposed to auto-sensing at submit time before returning the response), the implementation aborts the job, puts the job in the 'aborted' state and sets the 'unsupported-document-format' value in the job's "job-state-reasons".

A client should always provide a valid "document-format" operation attribute whenever practical. In the absence of other information, a client itself may sniff the document data to determine document format.

Auto sensing at Job submission time may be more difficult for the Printer when combined with compression. For auto-sensed Jobs, a client may be better off deferring compression to the transfer protocol layer, e.g.; by using the HTTP Content-Encoding header.

3.1.2.3.3 For the Validate-Job operation, RETURN one of the success status codes

If the requested operation is the Validate-Job operation, the Printer object returns:

1. the "successful-ok" status code, if there are no unsupported or conflicting Job Template attributes or values.
2. the "successful-ok-conflicting-attributes, if there are any conflicting Job Template attribute or values.
3. the "successful-ok-ignored-or-substituted-attributes, if there are only unsupported Job Template attributes or values.

Note: Unsupported Operation attributes or values that are returned do not affect the status returned in this step. If the unsupported Operation attribute was a serious error, the above already rejected the request in a previous step. If control gets to this step with unsupported Operation attributes being returned, they are not serious errors.

3.1.2.3.4 Create the Job object with attributes to support

If "ipp-attribute-fidelity" is set to 'false' (or it was not supplied by the client), the Printer object:

1. creates a Job object, assigns a unique value to the job's "job-uri" and "job-id" attributes, and initializes all of the job's other supported Job Description attributes.
2. removes all unsupported attributes from the Job object.
3. for each unsupported value, removes either the unsupported value or substitutes the unsupported attribute value with some supported value. If an attribute has no values after removing unsupported values from it, the attribute is removed from the Job object (so that the normal default behavior at job processing time will take place for that attribute).
4. for each conflicting value, removes either the conflicting value or substitutes the conflicting attribute value with some other supported value. If an attribute has no values after removing conflicting values from it, the attribute is removed from the Job object (so that the normal default behavior at job processing time will take place for that attribute).

If there were no attributes or values flagged as unsupported, or the value of 'ipp-attribute-fidelity" was 'false', the Printer object is able to accept the create request and create a new Job object. If the "ipp-attribute-fidelity" attribute is set to 'true', the Job Template attributes that populate the new Job object are necessarily all the Job Template attributes supplied in the create request. If the "ipp-attribute-fidelity" attribute is set to 'false', the Job Template attributes that populate the new Job object are all the client supplied Job Template attributes that are supported or that have value substitution. Thus, some of the requested Job Template attributes will not appear in the Job object because the Printer object did not support those attributes. The attributes that populate the Job object are persistently stored with the Job object for that Job. A Get-Job-Attributes operation on that Job object will return only those attributes that are persistently stored with the Job object.

Note: All Job Template attributes that are persistently stored with the Job object are intended to be "override values"; that is, they that take precedence over whatever other embedded instructions might be in the document data itself. However, it is not possible for all Printer objects to realize the semantics of "override". End users may query the Printer's "pdl-override-supported" attribute to determine if the Printer either attempts or does not attempt to override document data instructions with IPP attributes.

There are some cases, where a Printer supports a Job Template attribute and has an associated default value set for that attribute. In the case where a client does not supply the corresponding attribute, the Printer does not use its default values to populate Job attributes when creating the new Job object; only Job Template attributes actually in the create request are used to populate the Job object. The Printer's default values are only used later at Job processing time if no other IPP attribute or instruction embedded in the document data is present.

Note: If the default values associated with Job Template attributes that the client did not supply were to be used to populate the Job object, then these values would become "override values" rather than defaults. If the Printer supports the 'attempted' value of the "pdl-override-supported" attribute, then these override values could replace values specified within the document data. This is not the intent of the default value mechanism. A default value for an attribute is used only if the create request did not specify that attribute (or it was ignored when allowed by "ipp-attribute-fidelity" being 'false') and no value was provided within the content of the document data.

If the client does not supply a value for some Job Template attribute, and the Printer does not support that attribute, as far as IPP is concerned, the result of processing that Job (with respect to the missing attribute) is undefined.

3.1.2.3.5 Return one of the success status codes

Once the Job object has been created, the Printer object accepts the request and returns to the client:

1. the 'successful-ok' status code, if there are no unsupported or conflicting Job Template attributes or values.
2. the 'successful-ok-conflicting-attributes' status code, if there are any conflicting Job Template attribute or values.

3. the 'successful-ok-ignored-or-substituted-attributes' status code, if there are only unsupported Job Template attributes or values.

Note: Unsupported Operation attributes or values that are returned do not affect the status returned in this step. If the unsupported Operation attribute was a serious error, the above already rejected the request in a previous step. If control gets to this step with unsupported Operation attributes being returned, they are not serious errors.

The Printer object also returns Job status attributes that indicate the initial state of the Job ('pending', 'pending-held', 'processing', etc.), etc. See Print-Job Response, [RFC2911] section 3.2.1.2.

3.1.2.3.6 Accept appended Document Content

The Printer object accepts the appended Document Content data and either starts it printing, or spools it for later processing.

3.1.2.3.7 Scheduling and Starting to Process the Job

The Printer object uses its own configuration and implementation specific algorithms for scheduling the Job in the correct processing order. Once the Printer object begins processing the Job, the Printer changes the Job's state to 'processing'. If the Printer object supports PDL override (the "pdl-override-supported" attribute set to 'attempted'), the implementation does its best to see that IPP attributes take precedence over embedded instructions in the document data.

3.1.2.3.8 Completing the Job

The Printer object continues to process the Job until it can move the Job into the 'completed' state. If an Cancel-Job operation is received, the implementation eventually moves the Job into the 'canceled' state. If the system encounters errors during processing that do not allow it to progress the Job into a completed state, the implementation halts all processing, cleans up any resources, and moves the Job into the 'aborted' state.

3.1.2.3.9 Destroying the Job after completion

Once the Job moves to the 'completed', 'aborted', or 'canceled' state, it is an implementation decision as to when to destroy the Job object and release all associated resources. Once the Job has been destroyed, the Printer would return either the "client-error-not-found" or "client-error-gone" status codes for operations directed at that Job.

Note: the Printer object SHOULD NOT re-use a "job-uri" or "job-id" value for a sufficiently long time after a job has been destroyed, so that stale references kept by clients are less likely to access the wrong (newer) job.

3.1.2.3.10 Interaction with "ipp-attribute-fidelity"

Some Printer object implementations may support "ipp-attribute-fidelity" set to 'true' and "pdl-override-supported" set to 'attempted' and yet still not be able to realize exactly what the client specifies in the create request. This is due to legacy decisions and assumptions that have been made about the role of job instructions embedded within the document data and external job instructions that accompany the document data and how to handle conflicts between such instructions. The inability to be 100% precise about how a given implementation will behave is also compounded by the fact that the two special attributes, "ipp-attribute-fidelity" and "pdl-"override-supported", apply to the whole job rather than specific values for each attribute. For example, some implementations may be able to override almost all Job Template attributes except for "number-up". Character Sets, natural languages, and internationalization

This section discusses character set support, natural language support and internationalization.

3.1.2.3.11 Character set code conversion support

IPP clients and IPP objects are REQUIRED to support UTF-8. They MAY support additional charsets. It is RECOMMENDED that an IPP object also support US-ASCII, since many clients support US-ASCII, and indicate that UTF-8 and US-ASCII are supported by populating the Printer's "charset-supported" with 'utf-8' and 'us-ascii' values. An IPP object is required to code covert with as little loss as possible between the charsets that it supports, as indicated in the Printer's "charsets-supported" attribute.

How should the server handle the situation where the "attributes-charset" of the response itself is "us-ascii", but one or more attributes in that response is in the "utf-8" format?

Example: Consider a case where a client sends a Print-Job request with "utf-8" as the value of "attributes-charset" and with the "job-name" attribute supplied. Later another client submits a Get-Job-Attribute or Get-Jobs request. This second request contains the "attributes-charset" with value "us-ascii" and "requested-attributes" attribute with exactly one value "job-name".

According to the RFC2911 document (section 3.1.4.2), the value of the "attributes-charset" for the response of the second request must be "us-ascii" since that is the charset specified in the request. The "job-name" value, however, is in "utf-8" format. Should the request be rejected even though both "utf-8" and "us-ascii" charsets are supported by the server? or should the "job-name" value be converted to "us-ascii" and return "successful-ok-conflicting-attributes" (0x0002) as the status code?

Answer: An IPP object that supports both utf-8 (REQUIRED) and us-ascii, the second paragraph of section 3.1.4.2 applies so that the IPP object MUST accept the request, perform code set conversion between these two charsets with "the highest fidelity possible" and return 'successful-ok', rather than a warning 'successful-ok-conflicting-attributes, or an error. The printer will do the best it can to convert between each of the character sets that it supports--even if that means providing a string of question marks because none of the characters are representable in US ASCII. If it can't perform such conversion, it MUST NOT advertise us-ascii as a value of its "attributes-charset-supported" and MUST reject any request that requests 'us-ascii'.

One IPP object implementation strategy is to convert all request text and name values to a Unicode internal representation. This is 16-bit and virtually universal. Then convert to the specified operation attributes-charset on output.

Also it would be smarter for a client to ask for 'utf-8', rather than 'us-ascii' and throw away characters that it doesn't understand, rather than depending on the code conversion of the IPP object.

3.1.2.3.12 What charset to return when an unsupported charset is requested (Issue 1.19)?

Section 3.1.4.1 Request Operation attributes was clarified in November 1998 as follows:

All clients and IPP objects MUST support the 'utf-8' charset [RFC2044] and MAY support additional charsets provided that they are registered with IANA [IANA-CS]. If the Printer object does not support the client supplied charset value, the Printer object MUST reject the request, set the "attributes-charset" to 'utf-8' in the response, and return the 'client-error-charset-not-supported' status code and any 'text' or 'name' attributes using the 'utf-8' charset.

Since the client and IPP object MUST support UTF-8, returning any text or name attributes in UTF-8 when the client requests a charset that is not supported should allow the client to display the text or name.

Since such an error is a client error, rather than a user error, the client should check the status code first so that it can avoid displaying any other returned 'text' and 'name' attributes that are not in the charset requested.

Furthermore, [RFC2911] section 14.1.4.14 client-error-charset-not-supported (0x040D) was clarified in November 1998 as follows:

For any operation, if the IPP Printer does not support the charset supplied by the client in the "attributes-charset" operation attribute, the Printer MUST reject the operation and return this status and any 'text' or 'name' attributes using the 'utf-8' charset (see Section 3.1.4.1).

3.1.2.3.13 Natural Language Override (NLO)

The 'text' and 'name' attributes each have two forms. One has an implicit natural language, and the other has an explicit natural language. The 'textWithoutLanguage' and 'textWithLanguage' are the two 'text' forms. The 'nameWithoutLanguage" and 'nameWithLanguage are the two 'name' forms. If a receiver (IPP object or IPP client) supports an attribute with attribute syntax 'text', it MUST support both forms in a request and a response. A sender (IPP client or IPP object) MAY send either form for any such attribute. When a sender sends a WithoutLanguage form, the implicit natural language is specified in the "attributes-natural-language" operation attribute, which all senders MUST include in every request and response.

When a sender sends a WithLanguage form, it MAY be different from the implicit natural language supplied by the sender or it MAY be the same. The receiver MUST treat either form equivalently.

There is an implementation decision for senders, whether to always send the WithLanguage forms or use the WithoutLanguage form when the attribute's natural language is the same as the request or response. The former approach makes the sender implementation simpler. The latter approach is more efficient on the wire and allows inter-working with non-conforming receivers that fail to support the WithLanguage forms. As each approach have advantages, the choice is completely up to the implementer of the sender.

Furthermore, when a client receives a 'text' or 'name' job attribute that it had previously supplied, that client MUST NOT expect to see the attribute in the same form, i.e., in the same WithoutLanguage or WithLanguage form as the client supplied when it created the job. The IPP object is free to transform the attribute from the WithLanguage form to the WithoutLanguage form and vice versa, as long as the natural language is preserved. However, in order to meet this latter requirement, it is usually simpler for the IPP object implementation to store the natural language explicitly with the attribute value, i.e., to store using an internal representation that resembles the WithLanguage form.

The IPP Printer MUST copy the natural language of a job, i.e., the value of the "attributes-natural-language" operation attribute supplied by the client in the create operation, to the Job object as a Job Description attribute, so that a client is able to query it. In returning a Get-Job-Attributes response, the IPP object MAY return one of three natural language values in the response's "attributes-natural-language" operation attribute: (1) that requested by the requester, (2) the natural language of the job, or (3) the configured natural language of the IPP Printer, if the requested language is not supported by the IPP Printer.

This "attributes-natural-language" Job Description attribute is useful for an IPP object implementation that prints start sheets in the language of the user who submitted the job. This same Job Description attribute is useful to a multi-lingual operator who has to communicate with different job submitters in different natural languages. This same Job Description attribute is expected to be used in the future to generate notification messages in the natural language of the job submitter.

Early drafts of [RFC2911] contained a job-level natural language override (NLO) for the Get-Jobs response. A job-level (NLO) is an (unrequested) Job Attribute which then specified the implicit natural language for any other WithoutLanguage job attributes returned in the response for that job. Interoperability testing of early implementations showed that no one was implementing the job-level NLO in Get-Job responses. So the job-level NLO was eliminated from the Get-Jobs response. This simplification makes all requests and responses consistent in that the implicit natural language for any WithoutLanguage 'text' or 'name' form is always supplied in the request's or response's "attributes-natural-language" operation attribute.

3.1.3 Status codes returned by operation

This section corresponds to [RFC2911] section 3.1.6 "Operation Response Status Codes and Status Messages". This section lists all status codes once in the first operation (Print-Job). Then it lists the status codes that are different or specialized for subsequent operations under each operation.

3.1.3.1 Printer Operations

3.1.3.1.1 Print-Job

The Printer object MUST return one of the following "status-code" values for the indicated reason. Whether all of the document data has been accepted or not before returning the success or error response depends on implementation. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the Job object has been created and the "job-id", and "job-uri" assigned and returned in the response:

successful-ok: no request attributes were substituted or ignored.
successful-ok-ignored-or-substituted-attributes: some supplied (1) attributes were ignored or (2) unsupported attribute syntaxes or values were substituted with supported values or were ignored. Unsupported attributes, attribute syntax's, or values MUST be returned in the Unsupported Attributes group of the response.
successful-ok-conflicting-attributes: some supplied attribute values conflicted with the values of other supplied attributes and were either substituted or ignored. Attributes or values which conflict with other attributes and have been substituted or ignored MUST be returned in the Unsupported Attributes group of the response as supplied by the client.

[RFC2911] section 3.1.6 Operation Status Codes and Messages states:

> If the Printer object supports the "status-message" operation attribute, it SHOULD use the REQUIRED 'utf-8' charset to return a status message for the following error status codes (see section 13 in [RFC2911]): 'client-error-bad-request', 'client-error-charset-not-supported', 'server-error-internal-error', 'server-error-operation-not-supported', and 'server-error-version-not-supported'. In this case, it MUST set the value of the "attributes-charset" operation attribute to 'utf-8' in the error response.
>
> For the following error status codes, no job is created and no "job-id" or "job-uri" is returned:
>
> client-error-bad-request: The request syntax does not conform to the specification.
> client-error-forbidden: The request is being refused for authorization or authentication reasons. The implementation security policy is to not reveal whether the failure is one of authentication or authorization.
> client-error-not-authenticated: Either the request requires authentication information to be supplied or the authentication information is not sufficient for authorization.
> client-error-not-authorized: The requester is not authorized to perform the request on the target object.
> client-error-not-possible: The request cannot be carried out because of the state of the system. See also 'server-error-not-accepting-jobs' status code, which MUST take precedence if the Printer object's "printer-accepting-jobs" attribute is 'false'.
> client-error-timeout: not applicable.
> client-error-not-found: the target object does not exist.
> client-error-gone: the target object no longer exists and no forwarding address is known.

client-error-request-entity-too-large: the size of the request and/or print data exceeds the capacity of the IPP Printer to process it.

client-error-request-value-too-long: the size of request variable length attribute values, such as 'text' and 'name' attribute syntax's, exceed the maximum length specified in [RFC2911] for the attribute and MUST be returned in the Unsupported Attributes Group.

client-error-document-format-not-supported: the document format supplied is not supported. The "document-format" attribute with the unsupported value MUST be returned in the Unsupported Attributes Group. This error SHOULD take precedence over any other 'xxx-not-supported' error, except 'client-error-charset-not-supported'.

client-error-attributes-or-values-not-supported: one or more supplied attributes, attribute syntax's, or values are not supported and the client supplied the "ipp-attributes-fidelity" operation attribute with a 'true' value. They MUST be returned in the Unsupported Attributes Group as explained below.

client-error-uri-scheme-not-supported: not applicable.

client-error-charset-not-supported: the charset supplied in the "attributes-charset" operation attribute is not supported. The Printer's "configured-charset" MUST be returned in the response as the value of the "attributes-charset" operation attribute and used for any 'text' and 'name' attributes returned in the error response. This error SHOULD take precedence over any other error, unless the request syntax is so bad that the client's supplied "attributes-charset" cannot be determined.

client-error-conflicting-attributes: one or more supplied attribute values conflicted with each other and the client supplied the "ipp-attributes-fidelity" operation attribute with a 'true' value. They MUST be returned in the Unsupported Attributes Group as explained below.

server-error-internal-error: an unexpected condition prevents the request from being fulfilled.

server-error-operation-not-supported: not applicable (since Print-Job is REQUIRED).

server-error-service-unavailable: the service is temporarily overloaded.

server-error-version-not-supported: the version in the request is not supported. The "closest" version number supported MUST be returned in the response.

server-error-device-error: a device error occurred while receiving or spooling the request or document data or the IPP Printer object can only accept one job at a time.

server-error-temporary-error: a temporary error such as a buffer full write error, a memory overflow, or a disk full condition occurred while receiving the request and/or the document data.

server-error-not-accepting-jobs: the Printer object's "printer-is-not-accepting-jobs" attribute is 'false'.

server-error-busy: the Printer is too busy processing jobs to accept another job at this time.

server-error-job-canceled: the job has been canceled by an operator or the system while the client was transmitting the document data.

3.1.3.1.2 Print-URI

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to Print-URI with the following specializations and differences. See Section 14 for a more complete description of each status code.

client-error-uri-scheme-not-supported: the URI scheme supplied in the "document-uri" operation attribute is not supported and is returned in the Unsupported Attributes group.
server-error-operation-not-supported: the Print-URI operation is not supported.

3.1.3.1.3 Validate-Job

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to Validate-Job. See Section 13 in [RFC2911] for a more complete description of each status code.

3.1.3.1.4 Create-Job

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to Create-Job with the following specializations and differences. See Section 13 in [RFC2911] for a more complete description of each status code.

server-error-operation-not-supported: the Create-Job operation is not supported.
client-error-multiple-document-jobs-not-supported: while the Create-Job and Send-Document operations are supported, this implementation doesn't support more than one document with data.

3.1.3.1.5 Get-Printer-Attributes

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to the Get-Printer-Attributes operation with the following specialization's and differences. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the requested attributes are returned in Group 3 in the response:

successful-ok: no operation attributes or values were substituted or ignored (same as Print-Job) and no requested attributes were unsupported.
successful-ok-ignored-or-substituted-attributes: The "requested-attributes" operation attribute MAY, but NEED NOT, be returned with the unsupported values.
successful-ok-conflicting-attributes: same as Print-Job.

For the error status codes, Group 3 is returned containing no attributes or is not returned at all:

client-error-not-possible: Same as Print-Job, in addition the Printer object is not accepting any requests.
client-error-request-entity-too-large: same as Print-job, except that no print data is involved.
client-error-attributes-or-values-not-supported: not applicable, since unsupported operation attributes and/or values MUST be ignored and an appropriate success code returned (see above).
client-error-conflicting-attributes: same as Print-Job, except that "ipp-attribute-fidelity" is not involved.
server-error-operation-not-supported: not applicable (since Get-Printer-Attributes is REQUIRED).
server-error-device-error: same as Print-Job, except that no document data is involved.

server-error-temporary-error: same as Print-Job, except that no document data is involved.
server-error-not-accepting-jobs: not applicable.
server-error-busy: same as Print-Job, except the IPP object is too busy to accept even query requests.
server-error-job-canceled: not applicable.

3.1.3.1.6 Get-Jobs

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to the Get-Jobs operation with the following specialization's and differences. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the requested attributes are returned in Group 3 in the response:

successful-ok: same as Get-Printer-Attributes (see section 3.1.3.1.5).
successful-ok-ignored-or-substituted-attributes: same as Get-Printer-Attributes (see section 3.1.3.1.5).
successful-ok-conflicting-attributes: same as Get-Printer-Attributes (see section 3.1.3.1.5).

For any error status codes, Group 3 is returned containing no attributes or is not returned at all. The following brief error status code descriptions contain unique information for use with Get-Jobs operation. See section 14 for the other error status codes that apply uniformly to all operations:

client-error-not-possible: Same as Print-Job, in addition the Printer object is not accepting any requests.
client-error-request-entity-too-large: same as Print-job, except that no print data is involved.
client-error-document-format-not-supported: not applicable.
client-error-attributes-or-values-not-supported: not applicable, since unsupported operation attributes and/or values MUST be ignored and an appropriate success code returned (see above).
client-error-conflicting-attributes: same as Print-Job, except that "ipp-attribute-fidelity" is not involved.
server-error-operation-not-supported: not applicable (since Get-Jobs is REQUIRED).
server-error-device-error: same as Print-Job, except that no document data is involved.
server-error-temporary-error: same as Print-Job, except that no document data is involved.
server-error-not-accepting-jobs: not applicable.
server-error-job-canceled: not applicable.

3.1.3.1.7 Pause-Printer

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to Pause-Printer with the following specializations and differences. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the Printer object is being stopped from scheduling jobs on all its devices.

successful-ok: no request attributes were substituted or ignored (same as Print-Job).
successful-ok-ignored-or-substituted-attributes: same as Print-Job.
successful-ok-conflicting-attributes: same as Print-Job.

For any of the error status codes, the Printer object has not been stopped from scheduling jobs on all its devices.

client-error-not-possible: not applicable.
client-error-not-found: the target Printer object does not exist.
client-error-gone: the target Printer object no longer exists and no forwarding address is known.
client-error-request-entity-too-large: same as Print-Job, except no document data is involved.
client-error-document-format-not-supported: not applicable.
client-error-conflicting-attributes: same as Print-Job, except that the Printer's "printer-is-accepting-jobs" attribute is not involved.
server-error-operation-not-supported: the Pause-Printer operation is not supported.
server-error-device-error: not applicable.
server-error-temporary-error: same as Print-Job, except no document data is involved.
server-error-not-accepting-jobs: not applicable.
server-error-job-canceled: not applicable.

3.1.3.1.8 Resume-Printer

All of the Print-Job status code descriptions in Section 3.1.3.1.1 Print-Job Response with the specialization's described for Pause-Printer are applicable to Resume-Printer. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the Printer object resumes scheduling jobs on all its devices.

successful-ok: no request attributes were substituted or ignored (same as Print-Job).
successful-ok-ignored-or-substituted-attributes: same as Print-Job.
successful-ok-conflicting-attributes: same as Print-Job.

For any of the error status codes, the Printer object does not resume scheduling jobs.

server-error-operation-not-supported: the Resume-Printer operation is not supported.

3.1.3.1.8.1 What about Printers unable to change state due to an error condition?

If, in case, the IPP printer is unable to change its state due to some problem with the actual printer device (say, it is shut down or there is a media-jam as indicated in [RFC2911]), what should be the result of the "Resume-Printer" operation? Should it still change the 'printer-state-reasons' and return success or should it fail ?

The Resume-Printer operation must clear the 'paused' or 'moving-to-paused' 'printer-state-message'. The operation must return a 'successful-ok' status code.

3.1.3.1.8.2 How is "printer-state" handled on Resume-Printer?

If the Resume-Printer operation succeeds, what should be the value of "printer-state" and who should take care of the "printer-state" attribute value later on ?

The Resume-Printer operation may change the "printer-state-reasons" value.

The "printer-state" will change to one of three states:

1. 'idle' - no additional jobs and no error conditions present
2. 'processing' - job available and no error conditions present
3. current state (i.e. no change) an error condition is present (e.g. media jam)

In the third case the "printer-state-reason" will be cleared by automata when it detects the error condition no longer exists. The "printer-state" will move to 'idle' or 'processing' when conditions permit. (i.e. no more error conditions)

3.1.3.1.9 Purge-Printer

All of the Print-Job status code descriptions in Section 3.1.3.1.1 Print-Job Response with the specialization's described for Pause-Printer are applicable to Purge-Printer. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the Printer object purges all it's jobs.

successful-ok: no request attributes were substituted or ignored (same as Print-Job).
successful-ok-ignored-or-substituted-attributes: same as Print-Job.
successful-ok-conflicting-attributes: same as Print-Job.

For any of the error status codes, the Printer object does not purge any jobs.

server-error-operation-not-supported: the Purge-Printer operation is not supported.

3.1.3.2 Job Operations

3.1.3.2.1 Send-Document

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to the Get-Printer-Attributes operation with the following specialization's and differences. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the document has been added to the specified Job object and the job's "number-of-documents" attribute has been incremented:

successful-ok: no request attributes were substituted or ignored (same as Print-Job).
successful-ok-ignored-or-substituted-attributes: same as Print-Job.
successful-ok-conflicting-attributes: same as Print-Job.

For the error status codes, no document has been added to the Job object and the job's "number-of-documents" attribute has not been incremented:

client-error-not-possible: Same as Print-Job, except that the Printer's "printer-is-accepting-jobs" attribute is not involved, so that the client is able to finish submitting a job that was created with a Create-Job operation after this attribute has been set to 'true'. Another condition is that the state of the job precludes Send-Document, i.e., the job has already been closed out by the client. However, if the IPP Printer closed out the job due to timeout, the 'client-error-timeout' error status SHOULD be returned instead.
client-error-timeout: This request was sent after the Printer closed the job, because it has not received a Send-Document or Send-URI operation within the Printer's "multiple-operation-time-out" period .
client-error-request-entity-too-large: same as Print-Job.
client-error-conflicting-attributes: same as Print-Job, except that "ipp-attributes-fidelity" operation attribute is not involved..
server-error-operation-not-supported: the Send-Document request is not supported.
server-error-not-accepting-jobs: not applicable.
server-error-job-canceled: the job has been canceled by an operator or the system while the client was transmitting the data.

3.1.3.2.2 Send-URI

All of the Print-Job status code descriptions in Section 3.1.3.1.1 Print-Job Response with the specialization's described for Send-Document are applicable to Send-URI. See Section 13 in [RFC2911] for a more complete description of each status code.

client-error-uri-scheme-not-supported: the URI scheme supplied in the "document-uri" operation attribute is not supported and the "document-uri" attribute MUST be returned in the Unsupported Attributes group.
server-error-operation-not-supported: the Send-URI operation is not supported.

3.1.3.2.3 Cancel-Job

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to Cancel-Job with the following specializations and differences. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the Job object is being canceled or has been canceled:

successful-ok: no request attributes were substituted or ignored (same as Print-Job).
successful-ok-ignored-or-substituted-attributes: same as Print-Job.

successful-ok-conflicting-attributes: same as Print-Job.

For any of the error status codes, the Job object has not been canceled or was previously canceled.

client-error-not-possible: The request cannot be carried out because of the state of the Job object ('completed', 'canceled', or 'aborted') or the state of the system.
client-error-not-found: the target Printer and/or Job object does not exist.
client-error-gone: the target Printer and/or Job object no longer exists and no forwarding address is known.
client-error-request-entity-too-large: same as Print-Job, except no document data is involved.
client-error-document-format-not-supported: not applicable.
client-error-attributes-or-values-not-supported: not applicable, since unsupported operation attributes and values MUST be ignored.
client-error-conflicting-attributes: same as Print-Job, except that the Printer's "printer-is-accepting-jobs" attribute is not involved.
server-error-operation-not-supported: not applicable (Cancel-Job is REQUIRED).
server-error-device-error: same as Print-Job, except no document data is involved.
server-error-temporary-error: same as Print-Job, except no document data is involved.
server-error-not-accepting-jobs: not applicable..
server-error-job-canceled: not applicable.

3.1.3.2.4 Get-Job-Attributes

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to Get-Job-Attributes with the following specializations and differences. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the requested attributes are returned in Group 3 in the response:

successful-ok: same as Get-Printer-Attributes (see section 3.1.3.1.5).
successful-ok-ignored-or-substituted-attributes: same as Get-Printer-Attributes (see section 3.1.3.1.5).
successful-ok-conflicting-attributes: same as Get-Printer-Attributes (see section 3.1.3.1.5).

For the error status codes, Group 3 is returned containing no attributes or is not returned at all.

client-error-not-possible: Same as Print-Job, in addition the Printer object is not accepting any requests.
client-error-document-format-not-supported: not applicable.
client-error-attributes-or-values-not-supported: not applicable.
client-error-uri-scheme-not-supported: not applicable.
client-error-attributes-or-values-not-supported: not applicable, since unsupported operation attributes and/or values MUST be ignored and an appropriate success code returned (see above).
client-error-conflicting-attributes: not applicable
server-error-operation-not-supported: not applicable (since Get-Job-Attributes is REQUIRED).

server-error-device-error: same as Print-Job, except no document data is involved.
server-error-temporary-error: sane as Print-Job, except no document data is involved..
server-error-not-accepting-jobs: not applicable.
server-error-job-canceled: not applicable.

3.1.3.2.5 Hold-Job

All of the Print-Job status codes described in Section 3.1.3.1.1 Print-Job Response are applicable to Hold-Job with the following specializations and differences. See Section 13 in [RFC2911] for a more complete description of each status code.

For the following success status codes, the Job object is being held or has been held:

successful-ok: no request attributes were substituted or ignored (same as Print-Job).
successful-ok-ignored-or-substituted-attributes: same as Print-Job.
successful-ok-conflicting-attributes: same as Print-Job.

For any of the error status codes, the Job object has not been held or was previously held.

client-error-not-possible: The request cannot be carried out because of the state of the Job object ('completed', 'canceled', or 'aborted') or the state of the system.
client-error-not-found: the target Printer and/or Job object does not exist.
client-error-gone: the target Printer and/or Job object no longer exists and no forwarding address is known.
client-error-request-entity-too-large: same as Print-Job, except no document data is involved.
client-error-document-format-not-supported: not applicable.
client-error-conflicting-attributes: same as Print-Job, except that the Printer's "printer-is-accepting-jobs" attribute is not involved.
server-error-operation-not-supported: the Hold-Job operation is not supported.
server-error-device-error: not applicable.
server-error-temporary-error: same as Print-Job, except no document data is involved.
server-error-not-accepting-jobs: not applicable.
server-error-job-canceled: not applicable.

3.1.3.2.6 Release-Job

All of the Print-Job status code descriptions in Section 3.1.3.1.1 Print-Job Response with the specialization's described for Hold-Job are applicable to Release-Job. See Section 13 in [RFC2911] for a more complete description of each status code.

server-error-operation-not-supported: the Release-Job operation is not supported.

3.1.3.2.7 Restart-Job

All of the Print-Job status code descriptions in Section 3.1.3.1.1 Print-Job Response with the specialization's described for Hold-Job are applicable to Restart-Job. See Section 13 in [RFC2911] for a more complete description of each status code.

server-error-operation-not-supported: the Restart-Job operation is not supported.

3.1.3.2.7.1 Can documents be added to a restarted job?

Assume I give a Create-Job request along with a set of 5 documents . All the documents get printed and the job state is moved to completed . I issue a Restart-Job request on the job. Now the issue is that, if I try to add new documents to the restarted job, will the IPP Server permit me to do so or return "client-error-not-possible " and again print those 5 jobs?

A job can not move to the 'completed' state until all the documents have been processed. The 'last-document' flag indicates when the last document for a job is being sent from the client. This is the semantic equivalent of closing a job. No documents may be added once a job is closed. Section 3.3.7 of the IPP/1.1 model states "The job is moved to the 'pending' job state and restarts the beginning on the same IPP Printer object with the same attribute values." 'number-of-documents' is a job attribute.

3.1.4 Returning unsupported attributes in Get-Xxxx responses (Issue 1.18)

In the Get-Printer-Attributes, Get-Jobs, or Get-Job-Attributes responses, the client cannot depend on getting unsupported attributes returned in the Unsupported Attributes group that the client requested, but are not supported by the IPP object. However, such unsupported requested attributes will not be returned in the Job Attributes or Printer Attributes group (since they are unsupported). Furthermore, the IPP object is REQUIRED to return the 'successful-ok-ignored-or-substituted-attributes' status code, so that the client knows that not all that was requested has been returned.

3.1.5 Sending empty attribute groups

The [RFC2911] and [RFC2910] specifications RECOMMEND that a sender not send an empty attribute group in a request or a response. However, they REQUIRE a receiver to accept an empty attribute group as equivalent to the omission of that group. So a client SHOULD omit the Job Template Attributes group entirely in a create operation that is not supplying any Job Template attributes. Similarly, an IPP object SHOULD omit an empty Unsupported Attributes group if there are no unsupported attributes to be returned in a response.

The [RFC2910] specification REQUIRES a receiver to be able to receive either an empty attribute group or an omitted attribute group and treat them equivalently. The term "receiver" means an IPP object for a request and a client for a response. The term "sender' means a client for a request and an IPP object for a response.

There is an exception to the rule for Get-Jobs when there are no attributes to be returned. [RFC2910] contains the following paragraph:

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.

3.2 Printer Operations

3.2.1 Print-Job operation

3.2.1.1 Flow controlling the data portion of a Print-Job request (Issue 1.22)

A paused printer, or one that is stopped due to paper out or jam or spool space full or buffer space full, may flow control the data of a Print-Job operation (at the TCP/IP layer), so that the client is not able to send all the document data. Consequently, the Printer will not return a response until the condition is changed.

The Printer should not return a Print-Job response with an error code in any of these conditions, since either the printer will be resumed and/or the condition will be freed either by human intervention or as jobs print.

In writing test scripts to test IPP Printers, the script must also be written not to expect a response, if the printer has been paused, until the printer is resumed, in order to work with all possible implementations.

3.2.1.2 Returning job-state in Print-Job response (Issue 1.30)

An IPP client submits a small job via Print-Job. By the time the IPP printer/print server is putting together a response to the operation, the job has finished printing and been removed as an object from the print system. What should the job-state be in the response?

The Model suggests that the Printer return a response before it even accepts the document content. The Job Object Attributes are returned only if the IPP object returns one of the success status codes. Then the job-state would always be "pending" or "pending-held".

This issue comes up for the implementation of an IPP Printer object as a server that forwards jobs to devices that do not provide job status back to the server. If the server is reasonably certain that the job completed successfully, then it should return the job-state as 'completed'. Also the server can keep the job in its "job history" long after the job is no longer in the device. Then a user could query the server and see that the job was in the 'completed' state and completed as specified by the jobs "time-at-completed" time, which would be the same as the server submitted the job to the device.

An alternative is for the server to respond to the client before or while sending the job to the device, instead of waiting until the server has finished sending the job to the device. In this case, the server can return the job's state as 'pending' with the 'job-outgoing' value in the job's "job-state-reasons" attribute.

If the server doesn't know for sure whether the job completed successfully (or at all), it could return the (out-of-band) 'unknown' value.

On the other hand, if the server is able to query the device and/or setup some sort of event notification that the device initiates when the job makes state transitions, then the server can return the current job state in the Print-Job response and in subsequent queries because the server knows what the job state is in the device (or can query the device).

All of these alternatives depend on implementation of the server and the device.

3.2.2 Get-Printer-Attributes operation

If a Printer supports the "printer-make-and-model" attribute and returns the .INF file model name of the printer in that attribute, the Microsoft client will automatically install the correct driver (if available).

Clients which poll periodically for printer status or queued-job-count should use the "requested-attributes" operation attribute to limit the scope of the query in order to save Printer and network resources.

3.2.3 Get-Jobs operation

3.2.3.1 Get-Jobs, my-jobs='true', and 'requesting-user-name' (Issue 1.39)?

In [RFC2911] section 3.2.6.1 'Get-Jobs Request', if the attribute 'my-jobs' is present and set to TRUE, MUST the 'requesting-user-name' attribute be there too, and if it's not present what should the IPP printer do?

[RFC2911] Section 8.3 describes the various cases of "requesting-user-name" being present or not for any operation. If the client does not supply a value for "requesting-user-name", the printer MUST assume that the client is supplying some anonymous name, such as "anonymous".

3.2.3.2 Why is there a "limit" attribute in the Get-Jobs operation?

When using the Get-Jobs operation a client implementer might choose to limit the number of jobs that the client shows on the first screenful. For example, if its UI can only display 50 jobs, it can defend itself against a printer that would otherwise return 500 jobs, perhaps taking a long time on a slow dial-up line. The client can then go and ask for a larger number of jobs in the background, while showing the user the first 50 jobs. Since the job history is returned in reverse order, namely the most recently completed jobs are returned first, the user is most likely interested in the first jobs that are returned. Limiting the number of jobs may be especially useful for a client that is requesting 'completed' jobs from a printer that keeps a long job history. Clients that don't mind sometimes getting very large responses, can omit the "limit" attribute in their Get-Jobs requests.

3.2.4 Create-Job operation

A Printer may respond to a Create-Job operation with "job-state" 'pending' or 'pending-held' and " job-state-reason" 'job-data-insufficient' to indicate that operation has been accepted by the Printer, but the Printer is expecting additional document data before it can move the job into the 'processing' state. Alternatively, it may respond with "job-state" 'processing' and "job-state-reason" 'job-incoming' to indicate that the Create-Job operation has been accepted by the Printer, but the Printer is expecting additional Send-Document and/or Send-URI operations and/or is accessing/accepting document data. The second alternative is for non-spooling Printers that don't implement the 'pending' state.

Should the server wait for the "last-document" operation attribute set to 'true' before starting to "process" the job?

It depends on implementation. Some servers spool the entire job, including all document data, before starting to process, so such an implementation would wait for the "last-document" before starting to process the job. If the time-out occurs without the "last-document", then the server takes one of the indicated actions in section 3.3.1 in the [RFC2911] document. Other servers will start to process document data as soon as they have some. These are the so-called "non-spooling" printers. Currently, there isn't a way for a client to determine whether the Printer will spool all the data or will start to process (and print) as soon as it has some data.

3.3 Job Operations

3.3.1 Validate-Job

The Validate-Job operation has been designed so that its implementation may be a part of the Print-Job operation. Therefore, requiring Validate-Job is not a burden on implementers. Also it is useful for client's to be able to count on its presence in all conformance implementations, so that the client can determine before sending a long document, whether the job will be accepted by the IPP Printer or not.

3.3.2 Restart-Job

The Restart-Job operation allows the reprocessing of a completed job. Some jobs store the document data on the printer. Jobs created using the Print-Job operation are an example. It is required that the printer retains the job data after the job has moved to a 'completed state' in order for the Restart-Job operation to succeed.

Some jobs contain only a reference to the job data. A job created using the Print-URI is an example of such a job. When the Restart-Job operation is issued the job is reprocessed. The job data MUST be retrieved again to print the job.

It is possible that a job fails while attempting to access the print data. When such a job is the target of a Restart-Job the Printer SHALL attempt to retrieve the job data again.

4 Object Attributes

4.1 Attribute Syntax's

4.1.1 The 'none' value for empty sets (Issue 1.37)

[RFC2911] states that the 'none' value should be used as the value of a 1setOf when the set is empty. In most cases, sets that are potentially empty contain keywords so the keyword 'none' is used, but for the 3 finishings attributes, the values are enums and thus the empty set is represented by the enum 3. Currently there are no other attributes with 1setOf values, which can be empty and can contain values that are not keywords. This exception requires special code and is a potential place for bugs. It would have been better if we had chosen an out-of-band value, either "no-value" or some new value, such as 'none'. Since we didn't, implementations have to deal with the different representations of 'none', depending on the attribute syntax.

4.1.2 Multi-valued attributes (Issue 1.31)

What is the attribute syntax for a multi-valued attribute? Since some attributes support values in more than one data type, such as "media", "job-hold-until", and "job-sheets", IPP semantics associate the attribute syntax with each value, not with the attribute as a whole. The protocol associates the attribute syntax tag with each value. Don't be fooled, just because the attribute syntax tag comes before the attribute keyword. All attribute values after the first have a zero length attribute keyword as the indication of a subsequent value of the same attribute.

4.1.3 Case Sensitivity in URIs (issue 1.6)

IPP client and server implementations must be aware of the diverse uppercase/lowercase nature of URIs. RFC 2396 defines URL schemes and Host names as case insensitive but reminds us that the rest of the URL may well demonstrate case sensitivity. When creating URL's for fields where the choice is completely arbitrary, it is probably best to select lower case. However, this cannot be guaranteed and implementations MUST NOT rely on any fields being case-sensitive or case-insensitive in the URL beyond the URL scheme and host name fields.

The reason that the IPP specification does not make any restrictions on URIs, is so that implementations of IPP may use off-the-shelf components that conform to the standards that define URIs, such as RFC 2396 and the HTTP/1.1 specifications [RFC2616]. See these specifications for rules of matching, comparison, and case-sensitivity.

It is also recommended that System Administrators and implementations avoid creating URLs for different printers that differ only in their case. For example, don't have Printer1 and printer1 as two different IPP Printers.

Example of equivalent URI's

http://abc.com:80/~smith/home.html

http://ABC.com/%7Esmith/home.html

http:/ABC.com:/%7esmith/home.html

Example of equivalent URI's using the IPP scheme

ipp://abc.com:631/~smith/home.html

ipp://ABC.com/%7Esmith/home.html

http:/ABC.com:631/%7esmith/home.html

The HTTP/1.1 specification [RFC2616] contains more details on comparing URLs.

4.1.4 Maximum length for xxxWithLanguage and xxxWithoutLanguage

The 'textWithLanguage' and 'nameWithLanguage' are compound syntaxes that have two components. The first component is the 'language' component that can contain up to 63 octets. The second component is the 'text' or 'name' component. The maximum length of these are 1023 octets and 255 octets respectively. The definition of attributes with either syntax may further restrict the length. (e.g. printer-name (name(127)))

The length of the 'language' component has no effect on the allowable length of 'text' in 'textWithLanguage' or the length of 'name' in 'nameWithLanguage'

4.2 Job Template Attributes

4.2.1 multiple-document-handling(type2 keyword)

4.2.1.1 Support of multiple document jobs

IPP/1.0 is silent on which of the four effects an implementation would perform if it supports Create-Job, but does not support "multiple-document-handling" or multiple documents per job. IPP/1.1 was changed so that a Printer could support Create-Job without having to support multiple document jobs. The "multiple-document-jobs-supported" (boolean) Printer description attribute was added to IPP/1.1 along with the 'server-error-multiple-document-jobs-not-supported' status code for a Printer to indicate whether or not it supports multiple document jobs, when it supports the Create-Job operation. Also IPP/1.1 was clarified that the Printer MUST support the "multiple-document-handling" (type2 keyword) Job Template attribute with at least one value if the Printer supports multiple documents per job.

4.3 Job Description Attributes

4.3.1 Getting the date and time of day

The "date-time-at-creation", "date-time-at-processing", and "date-time-at-completed" attributes are returned as dateTime syntax. These attributes are OPTIONAL for a Printer to support. However, there are various ways for a Printer to get the date and time of day. Some suggestions:

1. A Printer can get time from an NTP timeserver if there's one reachable on the network . See RFC 1305. Also DHCP option 32 in RFC 2132 returns the IP address of the NTP server.

2. Get the date and time at startup from a human operator

3. Have an operator set the date and time using a web administrative interface

4. Get the date and time from incoming HTTP requests, though the problems of spoofing need to be considered. Perhaps comparing several HTTP requests could reduce the chances of spoofing.

5. Internal date time clock battery driven.

6. Query "http://tycho.usno.navy.mil/cgi-bin/timer.pl"

4.4 Printer Description Attributes

4.4.1 queued-job-count (integer(0:MAX))

4.4.1.1 Why is "queued-job-count" RECOMMENDED (Issue 1.14)?

The reason that "queued-job-count" is RECOMMENDED, is that some clients look at that attribute alone when summarizing the status of a list of printers, instead of doing a Get-Jobs to determine the number of jobs in the queue. Implementations that fail to support the "queued-job-count" will cause that client to display 0 jobs when there are actually queued jobs.

We would have made it a REQUIRED Printer attribute, but some implementations had already been completed before the issue was raised, so making it a SHOULD was a compromise.

4.4.1.2 Is "queued-job-count" a good measure of how busy a printer is (Issue 1.15)?

The "queued-job-count" is not a good measure of how busy the printer is when there are held jobs. A future registration could be to add a "held-job-count" (or an "active-job-count") Printer Description attribute if experience shows that such an attribute (combination) is needed to quickly indicate how busy a printer really is.

4.4.2 printer-current-time (dateTime)

A Printer implementation MAY support this attribute by obtaining the date and time by any number of implementation-dependent means at startup or subsequently. Examples include:

1. an internal date time clock,
2. from the operator at startup using the console,
3. from an operator using an administrative web page,
4. from HTTP headers supplied in client requests,
5. use HTTP to query "http://tycho.usno.navy.mil/cgi-bin/timer.pl"
6. from the network, using NTP [RFC1305] or DHCP option 32 [RFC2132] that returns the IP address of the NTP server.

If an implementation supports this attribute by obtaining the current time from the network (at startup or later), but the time is not available, then the implementation MUST return the value of this attribute using the out-of-band 'no-value' meaning not configured. See the beginning of section 4.1.

Since the new "date-and-time-at-xxx" Job Description attributes refer to the "printer-current-time", they will be covered also.

4.4.3 Printer-uri

Must the operational attribute for printer-uri match one of the values in "printer-uri-supported"?

A forgiving printer implementation would not reject the operation. But the implementation has its rights to reject a printer or job operation if the operational attribute printer-uri is not a value of the printer-uri-supported. The printer might not be improperly configured. The request obviously reached the printer. The printer could treat the printer-uri as the logical equivalent of a value in the printer-uri-supported. It would be implementation dependent for which value, and associated security policy, would apply. This does also apply to a job object specified with a printer-uri and job-id, or with a job-uri. See section 4.1.3 for how to compare URI's.

4.5 Empty Jobs

The IPP object model does not prohibit a job that contains no documents. Such a job may be created in a number of ways including a 'create-job' followed by an 'add-document' that contains no data and has the 'last-document' flag set.

An empty job is processed just as any other job. The operation that "closes" an empty job is not rejected because the job is empty. If no other conditions exist, other than the job is empty, the response to the operation will indicate success. After the job is scheduled and processed, the job state SHALL be 'completed'.

There will be some variation in the value(s) of the "job-state-reasons" attribute. It is required that if no conditions, other than the job being empty, exist the "job-state-reasons" SHALL include the 'completed-successfully'. If other conditions existed, the 'completed-with-warnings' or 'completed-with-errors' values may be used.

5 Directory Considerations

5.1 General Directory Schema Considerations

The [RFC2911] document lists RECOMMENDED and OPTIONAL Printer object attributes for directory schemas. See [RFC2911] APPENDIX E: Generic Directory Schema.

The SLP printer template is defined in the "Definition of the Printer Abstract Service Type v2.0" document [svrloc-printer]. The LDAP printer template is defined in the "Internet Printing Protocol (IPP): LDAP Schema for Printer Services" document [ldap-printer]. Both documents systematically add "printer-" to any attribute that doesn't already start with "printer-" in order to keep the printer directory attributes distinct from other directory attributes. Also, instead of using "printer-uri-supported", "uri-authentication-supported", and "uri-security-supported", they use a "printer-xri-supported" attribute with special syntax to contain all of the same information in a single attribute.

5.2 IPP Printer with a DNS name

If the IPP printer has a DNS name should there be at least two values for the printer-uri-supported attribute. One URL with the fully qualified DNS name the other with the IP address in the URL?

The printer may contain one or the other or both. It's up to the administrator to configure this attribute.

6 Security Considerations

The security considerations given in [RFC2911] Section 8 "Security Considerations" all apply to this document. In addition, the following sub-sections describes security consideration that have arisen as a result of implementation testing.~~This section corresponds to the RFC2911 Section 8 "Security Considerations.~~

6.1 Querying jobs with IPP that were submitted using other job submission protocols (Issue 1.32)

The following clarification was added to [RFC2911] section 8.5:

8.5 Queries on jobs submitted using non-IPP protocols

If the device that an IPP Printer is representing is able to accept jobs using other job submission protocols in addition to IPP, it is RECOMMEND that such an implementation at least allow such "foreign" jobs to be queried using Get-Jobs returning "job-id" and "job-uri" as 'unknown'. Such an implementation NEED NOT support all of the same IPP job attributes as for IPP jobs. The IPP object returns the 'unknown' out-of-band value for any requested attribute of a foreign job that is supported for IPP jobs, but not for foreign jobs.

It is further RECOMMENDED, that the IPP Printer generate "job-id" and "job-uri" values for such "foreign jobs", if possible, so that they may be targets of other IPP operations, such as Get-Job-Attributes and Cancel-Job. Such an implementation also needs to deal with the problem of authentication of such foreign jobs. One approach would be to treat all such foreign jobs as belonging to users other than the user of the IPP client. Another approach would be for the foreign job to belong to 'anonymous'. Only if the IPP client has been authenticated as an operator or administrator of the IPP Printer object, could the foreign jobs be queried by an IPP request. Alternatively, if the security policy were to allow users to query other users' jobs, then the foreign jobs would also be visible to an end-user IPP client using Get-Jobs and Get-Job-Attributes.

Thus IPP MAY be implemented as a "universal" protocol that provides access to jobs submitted with any job submission protocol. As IPP becomes widely implemented, providing a more universal access makes sense.

7 Encoding and Transport

This section discusses various aspects of IPP/1.1 Encoding and Transport [RFC2910].

A server is not required to send a response until after it has received the client's entire request. Hence, a client must not expect a response until after it has sent the entire request. However, we recommend that the server return a response as soon as possible if an error is detected while the client is still sending the data, rather than waiting until all of the data is received. Therefore, we also recommend that a client listen for an error response that an IPP server MAY send before it receives all the data. In this case a client, if chunking the data, can send a premature zero-length chunk to end the request before sending all the data (and so the client can keep the connection open for other requests, rather than closing it). If the request is blocked for some reason, a client MAY determine the reason by opening another connection to query the server using Get-Printer-Attributes.

IPP, by design, uses TCP's built-in flow control mechanisms [RFC 793] to throttle clients when Printers are busy. Therefore, it is perfectly normal for an IPP client transmitting a Job to be blocked for a really long time. Accordingly, socket timeouts must be avoided. Some socket implementations have a timeout option, which specifies how long a write operation on a socket can be blocked before it times out and the blocking ends. A client should set this option for infinite timeout when transmitting Job submissions.

Some IPP client applications might be able to perform other useful work while a Job transmission is blocked. For example, the client may have other jobs that it could transmit to other Printers simultaneously. A client may have a GUI, which must remain responsive to the user while the Job transmission is blocked. These clients should be designed to spawn a thread to handle the Job transmission at its own pace, leaving the main application free to do other work. Alternatively, single-threaded applications could use non-blocking I/O.

Some Printer conditions, such as jam or lack of paper, could cause a client to be blocked indefinitely. Clients may open additional connections to the Printer to Get-Printer-Attributes, determine the state of the device, alert a user if the printer is stopped, and let a user decide whether to abort the job transmission or not.

In the following sections, there are 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.

7.1 General Headers

The following is a table for the general headers.

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 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] from RFC 2616 [RFC2616]
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	

199 7.2 Request Headers

200 The following is a table for the request headers.

Request-Header	**Client**	**Server**	**Request Values and Conditions**

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 2616 [RFC2616] and IANA registry for content-codings
Accept-Language	not	not	language information is within the application/ipp entity
Authorization	must-if	must	per RFC 2616. 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 2616. Because RFC recommends sending this header only with the user's approval, it is not very useful
Host	must	must	per RFC 2616
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 2616. A client MUST send this header when it receives a 401 "Unauthorized" response and a "Proxy-Authenticate" header.
Range	not	not	
Referrer	not	not	
User-Agent	not	not	

201 7.3 Response Headers

202 The following is a table for the request headers.

Response-Header	Server	Client	Response Values and Conditions
Accept-Ranges	not	not	
Age	not	not	
Location	must-if	may	per RFC 2616. When URI needs redirection.
Proxy-Authenticate	not	must	per RFC 2616
Public	may	may	per RFC 2616
Retry-After	may	may	per RFC 2616
Server	not	not	
Vary	not	not	
Warning	may	may	per RFC 2616
WWW-Authenticate	must-if	must	per RFC 2616. When a server needs to authenticate a client.

7.4 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 2616 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 2616. Header MUST be present if Transfer-Encoding is absent..
Content-Location	not	not	not	not	
Content-MD5	may	may	may	may	per RFC 2616
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	

7.5 Optional support for HTTP/1.0

IPP implementations consist of an HTTP layer and an IPP layer. In the following discussion, the term "client" refers to the HTTP client layer and the term "server" refers to the HTTP server layer. The Encoding and Transport document [RFC2910] requires that HTTP 1.1 MUST be supported by all clients and all servers. However, a client and/or a server implementation may choose to also support HTTP 1.0.

This option means that a server may choose to communicate with a (non-conforming) client that only supports HTTP 1.0. In such cases the server should not use any HTTP 1.1 specific parameters or features and should respond using HTTP version number 1.0.

This option also means that a client may choose to communicate with a (non-conforming) server that only supports HTTP 1.0. In such cases, if the server responds with an HTTP 'unsupported version number' to an HTTP 1.1 request, the client should retry using HTTP version number 1.0.

7.6 HTTP/1.1 Chunking

7.6.1 Disabling IPP Server Response Chunking

Clients MUST anticipate that the HTTP/1.1 server may chunk responses and MUST accept them in responses. However, a (non-conforming) HTTP client that is unable to accept chunked responses may attempt to request an HTTP 1.1 server not to use chunking in its response to an operation by using the following HTTP header:

```
TE: identity
```

This mechanism should not be used by a server to disable a client from chunking a request, since chunking of document data is an important feature for clients to send long documents.

7.6.2 Warning About the Support of Chunked Requests

This section describes some problems with the use of chunked requests and HTTP/1.1 servers.

The HTTP/1.1 standard [RFC2616] requires that conforming servers support chunked requests for any method. However, in spite of this requirement, some HTTP/1.1 implementations support chunked responses in the GET method, but do not support chunked POST method requests. Some HTTP/1.1 implementations that support CGI scripts [CGI] and/or servlets [Servlet] require that the client supply a Content-Length. These implementations might reject a chunked POST method and return a 411 status code (Length Required), might attempt to buffer the request and run out of room returning a 413 status code (Request Entity Too Large), or might successfully accept the chunked request.

Because of this lack of conformance of HTTP servers to the HTTP/1.1 standard, the IPP standard [RFC2910] REQUIRES that a conforming IPP Printer object implementation support chunked requests and that conforming clients accept chunked responses. Therefore, IPP object implementers are warned to seek HTTP server implementations that support chunked POST requests in order to conform to the IPP standard and/or use implementation techniques that support chunked POST requests.

8 References

[CGI]
CGI/1.1 (http://www.ietf.org/internet-drafts/draft-coar-cgi-v11-00.txt).

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

[ldap-printer]
Fleming, P., Jones, K., Lewis, H., McDonald, I., "Internet Printing Protocol (IPP): LDAP Schema for Printer Services", <draft-ietf-ipp-ldap-printer-schema-04~~1~~.txt>, work in progress, ~~April 27~~December 20, 2000.

[RFC793]
J. Postel, "Transmission Control Protocol", RFC 793.

[RFC1123]
Braden, S., "Requirements for Internet Hosts - Application and Support", RFC 1123, October, 1989.

[RFC2026]
S. Bradner, "The Internet Standards Process -- Revision 3", RFC 2026, October 1996.

[RFC2119]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119 , March 1997.

[RFC2396]
Berners-Lee, T., Fielding, R., Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[RFC2565]
R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell, "Internet Printing Protocol/1.0: Model and Semantics", RFC 2566, April 1999.

[RFC2566]
Herriot, R., Butler, S., Moore, P., Turner, R., "Internet Printing Protocol/1.0: Encoding and Transport", RFC 2565, April 1999.

[RFC2567]
Wright, D., "Design Goals for an Internet Printing Protocol", ~~draft-ietf-ipp-req-03.txt~~RFC 2567, ~~November, 1998~~April 1999.

[RFC2568]
Zilles, S., "Rationale for the Structure and Model and Protocol for the Internet Printing Protocol", RFC 2568, April 1999.

[RFC2569]
Herriot, R., Hastings, T., Jacobs, N., Martin, J., "Mapping between LPD and IPP Protocols", RFC 2569, April 1999.

[RFC2616]

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, June 1999.

[RFC2910]
Herriot, R., Butler, S., Moore, P., Turner, R., "Internet Printing Protocol/1.0: Encoding and Transport", RFC 2910, September, 2000.

[RFC2911]
R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell, "Internet Printing Protocol/1.0: Model and Semantics", RFC 2911, September, 2000.

[Servlet]
Servlet Specification Version 2.1 (http://java.sun.com/products/servlet/2.1/index.html).

[svrloc-printer]
St. Pierre, P., Isaacson, S., McDonald, I., "Definition of the Printer Abstract Service Type v2.0", http://www.isi.edu/in-notes/iana/assignments/svrloc-templates/printer.2.0.en (IANA Registered, May 27, 2000)~~<draft-ietf-svrloc-printer-scheme-06.txt>, work in progress, March 8, 2000~~.

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

9 Authors' Address

Thomas N. Hastings
Xerox Corporation
701 Aviation Blvd.
El Segundo, CA 90245
hastings@cp10.es.xerox.com

Carl-Uno Manros
Xerox Corporation
701 Aviation Blvd.
El Segundo, CA 90245
cmanros@cp10.es.xerox.com

Carl Kugler
Mail Stop 003G
IBM Printing Systems Co
6300 Diagonal Hwy
Boulder CO 80301
Kugler@us.ibm.com

Henrik Holst

i-data Printing Systems
Vadstrupvej 35-43
2880 Bagsvaerd, Denmark
hh@I-data.com

Peter Zehler
Xerox Corporation
800 Philips Road
Webster, NY 14580
peter.zehler@usa.xerox.com

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

To subscribe to the ipp mailing list, send the following email:

1) send it to majordomo@pwg.org
2) leave the subject line blank
3) put the following two lines in the message body:

```
        subscribe ipp
        end
```

Implementers of this specification document are encouraged to join the IPP Mailing List in order to participate in any discussions of clarification issues and review of registration proposals for additional attributes and values. In order to reduce spam the mailing list rejects mail from non-subscribers, so you must subscribe to the mailing list in order to send a question or comment to the mailing list.

Other Participants:

Chuck Adams - Tektronix	Shivaun Albright - HP
Stefan Andersson - Axis	Jeff Barnett - IBM
Ron Bergman - Hitachi Koki Imaging Systems	Dennis Carney - IBM
Keith Carter - IBM	Angelo Caruso - Xerox
Rajesh Chawla - TR Computing Solutions	Nancy Chen - Okidata
Josh Cohen - Microsoft	Jeff Copeland - QMS
Andy Davidson - Tektronix	Roger deBry - IBM
Maulik Desai - Auco	Mabry Dozier - QMS
Lee Farrell - Canon Information Systems	Satoshi Fujitami - Ricoh
Steve Gebert - IBM	Sue Gleeson - Digital
Charles Gordon - Osicom	Brian Grimshaw - Apple
Jerry Hadsell - IBM	Richard Hart - Digital
Tom Hastings - Xerox	Henrik Holst - I-data
Stephen Holmstead	Zhi-Hong Huang - Zenographics
Scott Isaacson - Novell	Babek Jahromi - Microsoft
Swen Johnson - Xerox	David Kellerman - Northlake Software
Robert Kline - TrueSpectra	Charles Kong - Panasonic
Carl Kugler - IBM	Dave Kuntz - Hewlett-Packard
Takami Kurono - Brother	Rick Landau - Digital
Scott Lawrence - Agranot Systems	Greg LeClair - Epson
Dwight Lewis - Lexmark	Harry Lewis - IBM
Tony Liao - Vivid Image	Roy Lomicka - Digital
Pete Loya - HP	Ray Lutz - Cognisys
Mike MacKay - Novell, Inc.	David Manchala - Xerox
Carl-Uno Manros - Xerox	Jay Martin - Underscore
Stan McConnell - Xerox	Larry Masinter - Xerox
Sandra Matts - Hewlett Packard	Peter Michalek - Shinesoft
Ira McDonald - High North Inc.	Mike Moldovan - G3 Nova
Tetsuya Morita - Ricoh	Yuichi Niwa - Ricoh
Pat Nogay - IBM	Ron Norton - Printronics
Hugo Parra, Novell	Bob Pentecost - Hewlett-Packard
Patrick Powell - Astart Technologies	Jeff Rackowitz - Intermec
Eric Random - Peerless	Rob Rhoads - Intel
Xavier Riley - Xerox	Gary Roberts - Ricoh
David Roach - Unisys	Stuart Rowley - Kyocera
Yuji Sasaki - Japan Computer Industry	Richard Schneider - Epson
Kris Schoff - HP	Katsuaki Sekiguchi - Canon
Bob Setterbo - Adobe	Gail Songer - Peerless
Hideki Tanaka - Canon	Devon Taylor - Novell, Inc.
Mike Timperman - Lexmark	Atsushi Uchino - Epson
Shigeru Ueda - Canon	Bob Von Andel - Allegro Software
William Wagner - NetSilicon/DPI	Jim Walker - DAZEL
Chris Wellens - Interworking Labs	Trevor Wells - Hewlett Packard

Craig Whittle - Sharp Labs	Rob Whittle - Novell, Inc.
Jasper Wong - Xionics	Don Wright - Lexmark
Michael Wu - Heidelberg Digital	Rick Yardumian - Xerox
Michael Yeung - Toshiba	Lloyd Young - Lexmark
Atsushi Yuki - Kyocera	Peter Zehler - Xerox
William Zhang- Canon Information Systems	Frank Zhao - Panasonic
Steve Zilles - Adobe	Rob Zirnstein - Canon Information Systems

10 Description of the Base IPP Documents

In addition to this document, the base set of IPP documents includes:

Design Goals for an Internet Printing Protocol [RFC2567]
Rationale for the Structure and Model and Protocol for the Internet Printing Protocol [RFC2568]
Internet Printing Protocol/1.1: Model and Semantics [RFC2911]
Internet Printing Protocol/1.1: Encoding and Transport [RFC2910]
Mapping between LPD and IPP Protocols [RFC2569]

The "Design Goals for an Internet Printing Protocol" document takes a broad look at distributed printing functionality, and it enumerates real-life scenarios that help to clarify the features that need to be included in a printing protocol for the Internet. It identifies requirements for three types of users: end users, operators, and administrators. It calls out a subset of end user requirements that are satisfied in IPP/1.0 [RFC2566, RFC2565]. A few OPTIONAL operator operations have been added to IPP/1.1 [RFC2911, RFC2910].

The "Rationale for the Structure and Model and Protocol for the Internet Printing Protocol" document describes IPP from a high level view, defines a roadmap for the various documents that form the suite of IPP specification documents, and gives background and rationale for the IETF IPP working group's major decisions.

The "Internet Printing Protocol/1.1: Model and Semantics" document describes a simplified model with abstract objects, their attributes, and their operations. The model introduces a Printer and a Job. The Job supports multiple documents per Job. The model document also addresses how security, internationalization, and directory issues are addressed.

The "Internet Printing Protocol/1.1: Encoding and Transport" document is a formal mapping of the abstract operations and attributes defined in the model document onto HTTP/1.1 [RFC2616]. It also defines the encoding rules for a new Internet MIME media type called "application/ipp". This document also defines the rules for transporting a message body over HTTP whose Content-Type is “application/ipp”. This document defines the ‘ipp’ scheme for identifying IPP printers and jobs.

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

11 Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.