R. deBry
IBM Corporation
T. Hastings
Xerox Corporation
R. Herriot
Sun Microsystems
S. Isaacson
Novell, Inc.
P. Powell
San Diego State University
July 14, 1997

Internet Printing Protocol/1.0: Model and Semantics
draft-ietf-ipp-model-03.txt

Abstract

This document is one of a set of documents, which together describe all aspects of a new Internet Printing Protocol (IPP). IPP is an application level protocol that can be used for distributed printing using Internet tools and technology. The protocol is heavily influenced by the printing model introduced in the Document Printing Application (ISO/IEC 10175 DPA) standard. Although DPA specifies both end user and administrative features, IPP version 1.0 is focused only on end user functionality.

The full set of IPP documents includes:

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

The requirements document takes a broad look at distributed printing functionality, and it enumerates real-life scenarios that help to clarify the features that need to be included in a printing protocol for the Internet. It identifies requirements for three types of users: end users, operators, and administrators. The requirements document calls out a subset of end user requirements that MUST be satisfied in the first version of IPP. Operator and administrator requirements are out of scope for v1.0. The model and semantics document describes a simplified model with abstract objects, their attributes, and their operations. The model introduces a Printer object and a Job object. The Job object supports multiple documents per job. The security document covers potential threats and proposed counters to those threats. The protocol specification is formal document which incorporates the ideas in all the other documents into a concrete mapping using clearly defined data representations and transport protocol mappings that real implementers can use to develop interoperable client and server side components. Finally, the directory schema document shows a generic schema for directory service entries that represent instances of IPP Printers.

This document is the "Internet Printing Protocol/1.0: Model and Semantics" document.

Table of Contents

1.   Simplified Printing Model....................................7
2.   IPP Objects..9
2.1 Printer Object..10
2.2 Job Object..12
2.3 Document Object...13
2.4 Object Relationships..14
2.5 Object Identity...14
3.   IPP Operations..15
3.1 Operation Semantics...16
3.1.1   Get-Operations Operation...............................17
3.1.1.1   Get-Operations Request............................17
3.1.1.2   Get-Operations Response...........................17
3.1.2   Print-Job Operation....................................17
3.1.2.1   Print-Job Request.................................18
3.1.2.2   Print-Job Response................................20
3.1.3   Print-URI Operation....................................20
3.1.3.1   Print-URI Request.................................20
3.1.3.2   Print-URI Response................................21
3.1.4   Validate-Job Operation.................................21
3.1.4.1   Validate-Job Request..............................21
3.1.4.2   Validate-Job Response.............................21
3.1.5   Create-Job Operation...................................22
3.1.5.1   Create-Job Request................................22
3.1.5.2   Create Job Response...............................22
3.1.6   Send-Document Operation................................23
3.1.6.1   Send-Document Request.............................23
3.1.6.2   Send-Document Response............................23
3.1.7   Send-URI Operation.....................................23
3.1.7.1   Send-URI Request..................................24
3.1.7.2   Send-URI Response.................................24
3.1.8   Cancel Job Operation...................................24
3.1.8.1   Cancel-Job Request................................24
3.1.8.2   Cancel-Job Response...............................25
3.1.9   Get-Attributes Operation...............................25
3.1.9.1   Get-Attributes Request............................25
3.1.9.2   Get-Attributes Response...........................27
3.1.10   Get-Jobs Operation.....................................27
3.1.10.1   Get-Jobs Request..................................27
3.1.10.2   Get-Jobs Response.................................28
3.2 Operation Status Codes and Messages.........................28
4.   Object Attributes...29
4.1 Attribute Syntaxes..29

4.1.1 Attribute Extensibility.................................31
4.2 Job Template Attributes.....................................33
4.2.1 job-name (name)...37
4.2.2 job-sheets (type4 keyword)...............................38

```
4.2.3     notify-events (1setOf type2 keyword)......................38
4.2.4     notify-addresses (1setOf uri).............................38
4.2.5     job-priority (integer(1:100)).............................39
4.2.6     job-hold-until (type4 keyword)............................39
4.2.7     multiple-documenthandling (type2 keyword).................40
4.2.8     best-effort (boolean).....................................40
4.2.9     media (type4 keyword).....................................42
4.2.10    number-up (type3 keyword).................................42
4.2.11    sides (type2 keyword).....................................43
4.2.12    printer-resolution (type2 enum)...........................43
4.2.13    print-quality (type2 enum)................................44
4.2.14    copies (integer(1:2**31 - 1)).............................45
4.2.15    finishing (1setOf type2 enum).............................45
4.2.16    document-format (type2 keyword)...........................46
4.2.17    compression (type3 keyword)...............................46
4.2.18    job-k-octets (integer(0:2**31 - 1)).......................46
4.2.19    job-impressions (integer(0:2**31 - 1))....................46
4.2.20    job-media-sheets (integer(0:2**31 - 1))...................47
4.3  Job Description Attributes.....................................47
4.3.1     job-uri (uri).............................................48
4.3.2     job-uri-user (uri)........................................49
4.3.3     job-originating-user (name)...............................49
4.3.4     job-originating-host (name)...............................49
4.3.5     user-locale (locale)......................................49
4.3.6     job-state (type1 enum)....................................49
4.3.7     job-state-reasons (1setOf  type2 keyword).................51
4.3.8     job-state-message (text)..................................53
4.3.9     output-device-assigned (name).............................53
4.3.10    time-since-pending (milliseconds).........................53
4.3.11    time-since-processing (milliseconds)......................53
4.3.12    time-since-completed (milliseconds).......................53
4.3.13    number-of-intervening-jobs (integer(0:2**31 - 1)).........53
4.3.14    job-message-from-operator (text)..........................53
4.3.15    job-k-octets-processed (integer(0:2**31 - 1)).............54
4.3.16    job-impressions-completed  (integer(0:2**31 - 1)).........54
4.3.17    job-media-sheets-completed (integer(0:2**31 - 1)).........54
4.4  Document Attributes............................................54
4.4.1     document-name (name)......................................54
4.4.2     document-format (type2 enum)..............................55
4.4.3     document-uri (uri)........................................55
4.5  Printer Description Attributes.................................55
4.5.1     printer-uri (uri).........................................56
4.5.2     printer-uri-user (uri)....................................57
4.5.3     printer-name (name).......................................57
```

4.5.4 printer-location (text)..................................57
4.5.5 printer-description (text)...............................57
4.5.6 printer-more-info-site (uri).............................57
4.5.7 printer-driver-installer (uri)...........................57

```
4.5.8     printer-make-and-model (text)............................57
4.5.9     printer-more-info-manufacturer (uri).....................58
4.5.10    printer-state (type1 keyword)............................58
4.5.11    printer-state-reasons (1setOf type2 keyword).............59
4.5.12    printer-is-accepting-jobs (boolean)......................61
4.5.13    printer-state-message (text).............................61
4.5.14    queued-job-count (integer(0:2**31 - 1))..................61
4.5.15    printer-message-from-the-operator (text).................61
4.5.16    printer-locale (locale)..................................61
4.5.17    printer-locales-supported (1setOf locale)................61
4.5.18    color-supported (boolean)................................61
4.5.19    pdl-override (type2 keyword).............................62
4.5.20    Security Related Attributes..............................63
4.5.20.1       message-protection-supported (keyword)..............63
4.5.20.2       authentication-authorization-supported (keyword)....63
5.   Conformance...................................................64
5.1  Conditionally Mandatory.......................................64
5.2  Client Conformance Requirements...............................64
5.3  Printer Object Conformance Requirements.......................65
5.3.1     Objects..................................................65
5.3.2     Operations...............................................65
5.3.3     Attributes...............................................65
5.3.4     Printer extensions.......................................66
5.3.5     Attribute Syntaxes.......................................66
5.4  Security Conformance Requirements.............................66
6.   IANA Considerations (registered and private extensions).......66
6.1  Typed Extensions..............................................67
6.1.1     Type1....................................................67
6.1.2     Type2....................................................67
6.1.3     Type3....................................................68
6.1.4     Type4....................................................68
6.2  Registration of MIME types/sub-types for document-formats.....68
7.   Security Considerations.......................................69
8.   References....................................................69
9.   Author's Address..............................................70
10.  APPENDIX A: Terminology.......................................73
10.1 Conformance Terminology.......................................73
10.1.1    MUST.....................................................73
10.1.2    MUST NOT.................................................73
10.1.3    SHOULD...................................................73
10.1.4    SHOULD NOT...............................................73
10.1.5    MAY......................................................73
10.1.6    CONDITIONALLY MANDATORY..................................74
10.1.7    NEED NOT.................................................74
```

10.2 Model Terminology...74
10.2.1 Keyword..74
10.2.2 Parameters...74
10.2.2.1 Parameter Name...74

10.2.2.2   Parameter Value.................................75
10.2.2.3   Parameter Syntax................................75
10.2.3   Attributes...75
10.2.3.1   Attribute Name..................................75
10.2.3.2   Attribute Group Name............................76
10.2.3.3   Attribute Value.................................76
10.2.3.4   Attribute Syntax................................76
10.2.4   Supports...76
11. APPENDIX B: Status Codes...................................77
11.1 Status Codes (type2 keyword)................................78
11.1.1   Informational..78
11.1.2   Successful Status Codes................................78
11.1.2.1   successful-ok...................................78
11.1.3   Redirection Status Codes...............................78
11.1.4   Client Error Status Codes..............................79
11.1.4.1   client-error-bad-request........................79
11.1.4.2   client-error-forbidden..........................79
11.1.4.3   client-error-not-authenticated..................79
11.1.4.4   client-error-not-authorized.....................79
11.1.4.5   client-error-not-possible.......................80
11.1.4.6   client-error-timeout............................80
11.1.4.7   client-error-not-found..........................80
11.1.4.8   client-error-gone...............................80
11.1.4.9   client-error-request-entity-too-large...........81
11.1.4.10   client-error-request-URI-too-long...............81
11.1.4.11   client-error-unsupported-document-format........81
11.1.4.12   client-error-attribute- not-supported...........81
11.1.5   Server Error Status Codes..............................82
11.1.5.1   server-error-internal- error....................82
11.1.5.2   server-error-operation-not-supported............82
11.1.5.3   server-error-service-unavailable................82
11.1.5.4   server-error- version-not-supported.............82
11.1.5.5   server-error-device-error.......................83
11.1.5.6   server-error-temporary-error....................83
11.2 Status Keywords for IPP Operations..........................84
12. APPENDIX C: "document-format" enum values...................84
13. APPENDIX D: "media" keyword values.........................87

1. Simplified Printing Model

In order to a achieve its goal of realizing a workable printing protocol for the Internet, the Internet Printing Protocol (IPP) is based on a simplified printing model which abstracts the many (often complex) components of real world printing solutions. Many of these systems include features, interfaces, and relationships that are beyond the scope of IPP. IPP has to run in a distributed computing environment where requesters of print services (clients, applications, PC drivers, etc.) cooperate and interact with print service providers. Although the underlying configuration may be a complex n-tier client/server system, an important simplifying step in the IPP model is to expose only the key objects and interfaces required for printing. The IPP model encapsulates these important elements into three simple object types:

  Printer (Section 2.1)
  Job (Section 2.2)
  Document (Section 2.3)

Each object type has an associated set of operations (see section 3) and attributes (see section 4)

It is important, however, to understand that in real system implementations (which lie underneath the abstracted IPP model), there are other components of a print service which are not explicitly defined in the IPP model. The following figure illustrates where IPP fits with respect to these other components.

```
                             +--------------+
                             |  Application |
                   o         +. . . . . . . |
                  \|/        |   Spooler    |
                  / \        +. . . . . . . |   +---------+
                End-User     | Print Driver |---|  File   |
      +-----------+ +-----+  +------+-------+   +----+----+
      |  Browser  | | GUI |         |                |
      +-----+-----+ +--+--+         |                |
            |          |            |                |
            |      +---+------------+---+            |
N   D   S   |      |      IPP Client    |------------+
O   I   E   |      +---------+----------+
T   R   C   |                |
I   E   U   |
F   C   R   -------------- Transport ------------------
I   T   I
C   O   T                |         --+
A   R   Y       +--------+--------+  |
T   Y           |    IPP Server   |  |
I               +--------+--------+  |
O                        |           |
N               +-----------------+  | IPP Printer
                |  Print Service  |  |
                +-----------------+  |
                         |         --+
                +-----------------+
                | Output Device(s)|
                +-----------------+
```

IPP Printers encapsulate the functions normally associated with physical output devices along with the spooling, scheduling and multiple device management functions associated with a print server. Printers may be registered as entries in a directory where end users find and select them based on some sort of filtered and context based searching. The directory is used to store relatively static information about the Printer, allowing end users to search for and find Printers that match their search criteria (name, context, printer capabilities, etc.).

IPP clients implement the IPP protocol on the client side and give end users or programs the ability to query an IPP Printer and submit and manage their print jobs. An IPP server is just that part of the IPP

Printer that implements the protocol. The rest of the IPP Printer implements the application semantics of the print service itself. The IPP Printer may be embedded in an output device or may be implemented on a host on the network that communicates with the output device.

All information about the Printer, both static and dynamic information, can be accessed directly from the Printer itself. The more dynamic information associated with a Printer includes state, currently loaded and ready media, number of jobs on the Printer, errors, warnings, etc. The Printer is optionally represented by an entry in a directory service. The more static information (name, URI, location, etc.) are stored with along with the entry in the directory service to enable filtered directory queries.

When a job is submitted to the Printer, the Printer SHALL create a Job object. The end user then interacts with this new Job to query its status and monitor the progress of the job. End users may also cancel the Job. The end user is able to register to receive certain events which are then routed using the notification service(s).

2. IPP Objects

The IPP model introduces objects of type Printer, Job, and Document. Each object type is defined as a set of possible attributes that may be supported by each instance of an object of that type. The attributes (and values) supported by each object instance describe the implementation (that is the realizable features, functions, and characteristics either in software or hardware) for that object instance. For example, the object type "Printer" is defined as set of attributes that each instance of a Printer object might potentially support. In the same manner, the object type "Job" is defined as a set of attributes that are potentially supported by each instance of a Job object.

Each attribute included in the set of attributes defining an object type labeled as:

"MANDATORY": each object instance MUST support the attribute.
"OPTIONAL": each object instance MAY support the attribute.
"CONDITIONALLY MANDATORY": whether or not the object instance supports the attribute is determined by a semantic condition. For example, if the implementation behind a given instance of a Printer object knows about and is able to support multiple levels of job priorities, that instance MUST support the "job-priority-supported" attribute. An administrator may set the values to be somewhat more restrictive that what a given implementation might allow, however the attribute MUST still be supported.

2.1 Printer Object

A major component of the IPP model is the Printer object. The capabilities and state of an IPP Printer are described by its attributes. Printer attributes are grouped as follows:

"job-template" attributes (section 4.2)
"printer-description" attributes (section 4.5)

Operations which are invoked on a printer include:

Get-Operations (Section 3.1.1)
Print-Job (section 3.1.2)
Print-URI (Section 3.1.3)
Validate-Job (Section 3.1.4)
Create-Job (section 3.1.5)
Get-Attributes (section 3.1.9)
Get-Jobs (section 3.1.10)

An instance of a Printer object implements the IPP protocol. Using the protocol, end users may query the attributes of the Printer, submit jobs to the Printer, determine subsequent states of submitted and queued jobs, and cancel their own print jobs. The actual implementation components behind the Printer object abstraction may take on different forms and different configurations, however, the details of the configuration of real components are transparent to the end user.

Since a Printer object is an abstraction of a generic document output device and print service provider, an IPP Printer object could be used to represent any real or virtual device with semantics consistent with the Printer object. For example, an instance of a Printer object could be used to front end a fax-out device, any kind of imager, or even a CD writer.

Some examples of configurations supporting a Printer object include:

1) An output device, with no spooling capabilities
2) An output device, with a built-in spooler
3) A print server supporting IPP with one or more associated output devices
   3a) The associated output devices might or might not be capable of spooling jobs
   3b) The associated output devices might or might not support IPP

See the following figures for some examples on how to view Printer
objects on top of several print system configurations. The embedded

case below represents configurations 1 and 2. The hosted and fan-out figures below represent configuration 3.

```
Legend:

##### indicates a Printer object which is
      either embedded in an output device or is
      hosted in a server.  The implementation
      might or might not be capable of queuing/spooling.

any   indicates any network protocol or direct
      connect, including IPP


embedded printer:
                                          output device
                                        +---------------+
 O   +--------+                         |  ###########  |
/|\  | client |------------IPP------------># Printer #  |
/ \  +--------+                         |  # Object  #  |
                                        |  ###########  |
                                        +---------------+


hosted printer:
                                        +---------------+
 O   +--------+        ###########      |               |
/|\  | client |--IPP--># Printer #-any->| output device |
/ \  +--------+        # Object  #      |               |
                       ###########      +---------------+



                                         +---------------+
fan out:                                 |               |
                                     +-->| output device |
                                 any/    |               |
 O   +--------+      ###########   /     +---------------+
/|\  | client |-IPP-># Printer #--*
/ \  +--------+      # Object  #   \     +---------------+
                     ########### any\    |               |
                                     +-->| output device |
                                         |               |
                                         +---------------+
```

2.2 Job Object

A Job object is used to model a job. A job can contain one or more
documents. The information required to create a Job object is sent in

a create request from the end user via an IPP client to a Printer. A create request can be either a Print-Job Request, a Print-URI Request, or a Create-Job Request. The Printer MUST perform validation checks to verify that the job may indeed be processed. A client MAY send a Validate-Job Request (with no document data) so that the Printer performs all validation checks without the overhead of transferring all of the document data. As an example of some of the validation checks that are performed, the create request may specify that the documents within the job are to be printed duplex (on both sides of the media). However, the Printer might not support such a feature. Once the Printer validates the submitted information, a Job object is created. The instance of the Job object is initialized with information from the create request. If a Create-Job operation is used to create the Job object, subsequent Send-Document operations are used to transfer the document data from the client to the IPP Printer.

This model specification defines rules for what is done when:

- optional attributes are missing
- there are conflicts between what is supported and what is requested
- there are conflicts between what the client requests via external attributes in the IPP operation and what the client requests in embedded instructions in the document page description language (PDL).

Job attributes are grouped as follows:

"job-template" attributes (optionally supplied by the client/end user, section 4.2)
"job-description" attributes (set by the Printer, section 4.3)

The following operations can be invoked on Jobs:

Send-Document (section 3.1.6)
Send-URI (Section 3.1.7)
Cancel Job (section 3.1.8)
Get-Attributes (section 3.1.9)

2.3 Document Object

A Document object consists of either printable data or a reference (URI) to printable data and a set of Document Attributes (see section 4.4). These Document Attributes only describe the data to be printed;

they do not include any specialized document processing instructions
that apply to only this one Document. All Job Template attributes,
that isthose attributes that describe desired job processing behavior,

are defined as part of the Job object, therefore, they apply equally to all Documents within a Job.

Currently there are no operations defined for Document objects.

2.4 Object Relationships

Instances of objects within the system have relationships that MUST be maintained persistently along with the persistent storage of the object attributes. An instance of a Printer object usually represents one or more output devices. A Printer object may represent a logical device which "processes" jobs but never actually uses a physical output device to put marks on paper (for example a Web page publisher or an interface into an online document archive or repository). A Printer can contain zero or more Job objects. An instance of a Job object is contained in exactly one Printer object (the same document data could be sent to a the same or a different Printer, but the corresponding Job object would be an identical, but different Job object). A Job object contains one or more Documents. If the Document is simply a reference to some print data stream, the reference may be used in multiple Documents in the same Job or even in different Jobs. If the Document is not just a reference, but an actual stream of print data, the stream is contained in only one Document, although there can be copies of the same document data in other Documents in the same or different Jobs.

2.5 Object Identity

All instances of Printer and Job objects have a URI so that they can persistently and unambiguously referenced. The IPP model requires that these values be URIs as defined by RFC 1738 [11] and RFC 1808. In addition to an identifier attribute, instances of Printer and Job objects may have a name. An object name need not be unique across all instances of all objects. The Printer name is chosen and set by an administrator. If not supplied by the client, the Printer creates the Job name. In all cases, the name only has local meaning, and it is not constrained to be unique.

To summarize, each instance of Printer and Job objects will have two identifying attributes:

- "xxx-uri": The unique identifier for this object instance
- "xxx-name": The non unique name for this object instance

Document objects sent to an IPP Printer only have names, no
identifiers. The "document-name" attribute is used to store the name

of the Document. This name is just of interest within the context of a Job; it need not be unique.

If Documents are printed by reference, the corresponding document object contains a "document-uri" attribute, but the value of this attribute is a reference to the document data to be printed; it is not the unique identifier of the Document object itself.

3. IPP Operations

Jobs and Printers each have a set of associated operations. End users or programs invoke these operations using an IPP client. The operations are:

For a Printer object:
- Get-Operations (section 3.1.1) Print-Job (section 3.1.2) Print-URI (section 3.1.3)
- Validate-Job (section 3.1.4)
- Create-Job (section 3.1.5)
- Get-Jobs (section 3.1.8)
- Get-Attributes (section 3.1.9)

For a Job object:
- Send-Document (section 3.1.6)
- Send-URI (section 3.1.7)
- Cancel-Job (section 3.1.8)
- Get-Attributes (section 3.1.9)

When a client communicates with a remote IPP object, it sends an operation request to the URI for that object. Each request carries along with it the input parameters and data required to perform the specified operation. Each request requires a response from the object indicating success or failure of the operation including output parameters, status codes, and/or status messages. The representation and encoding of the IPP protocol are contained in "Internet Printing Protocol: Protocol Specification."[23]

It is assumed that URIs for IPP Printers are available to end users or programs that wish to invoke Printer operations. Although NOT MANDATORY, it is RECOMMENDED that Printers be registered in a directory service which end users and programs can interrogate. "Internet Printing Protocol: Directory Schema"[24] defines the attributes to be associated with a Printer entry in a directory service.

3.1 Operation Semantics

In this section, the IPP operations are described in terms of their contents and semantics including both the request and the response for each operation.

In order to create a new Job object, a client uses one of three operations:

- The Print-Job operation: This operation is used if the client wants to create a Job with only a single Document and the document data is included in the request. In this case, the client "pushes" the document data to the Printer.

- The Print-URI operation: This operation is used if the client wants to create a Job with only a single Document and only a URI reference to the document data (not the document data itself) is included in the request. In this case, the Printer "pulls" the document data from the location identified by the URI.

- The Create-Job operation: This operation is used if the client wants to create a Job with one or more Documents. This operation is followed by an arbitrary number of Send-Document or Send-URI operations (each creating another Document for this Job). The Send-Document operation includes the document data with the operation request (client "pushes" the document data to the printer), and the Send-URI operation includes only a reference (a URI) to the document data (the Printer "pulls" the document data from the referenced location).

A Create-Job operation followed by a only one Send-Document operation is semantically equivalent to a Print-Job operation, however, for performance reasons, the client SHOULD use the Print-Job operation for all single Document Jobs. Throughout this model specification, the term "create request" is used to refer to any of these three operation requests.

Every operation response returns a MANDATORY status code and an OPTIONAL status message (see Section 10). In most cases, if the status code indicates an error (the code belongs to either the "client-error" or the "server-error" group), there are additional output parameters returned that are not returned in the successful case.

In many of these operations, a client supplies a list of attributes to

be returned in the response. A Printer may be configured, for
security reasons, not to return all attributes that a client requests.
It may even return none of the requested attributes. In such cases,

the status returned is the same as if the Printer had returned all requested attributes. The client cannot tell by such a response whether the requested attribute was present or absent on the Printer.

3.1.1 Get-Operations Operation

Since some of the IPP operations defined in this specification are OPTIONAL and therefore some implementations may choose to not implement support them, this operation is a simple, MANDATORY operation that all implementations MUST support. The client uses this operation to query a specific implementation for a list of supported operations.

3.1.1.1 Get-Operations Request

The Get-Operations Request has no input parameters.

3.1.1.2 Get-Operations Response

The Printer returns the following output parameters as part of the Get-Operations Response:

Supported Operations ("supported-operations"):
A list of the operations that this implementation supports (including all MANDATORY operations). The values are taken from the following set: 'Get-Operations', 'Print-Job', 'Print-URI', 'Validate-Job', 'Create-Job', 'Get-Jobs', 'Get-Attributes', 'Send-Document', 'Send-URI', and 'Cancel-Job'.

Note: Since this list contains all MANDATORY operations ('Get-Operations', 'Print-Job', 'Validate-Job', 'Get-Jobs', 'Get-Attributes', and 'Cancel-Job'), this output parameter will never be empty.

3.1.2 Print-Job Operation

When an end user desires to submit a print job with only one Document, the client sends a Print-Job Request to a Printer and receives a Print-Job Response from that Printer. The information in a Print-Job Request (along with any default information associated with the Printer) is sufficient for the Printer to create a Job object and then process that Job. A Print-Job operation differs from a Print-URI operation in that a Print-Job operation contains the document data to be printed and a Print-URI operation only contains a reference to the document data.

3.1.2.1 Print-Job Request

The following input parameters are part of the Print-Job Request:

Job Template Attributes:
An optional set of Job Template attributes as defined in section 4.2. If the client supplies no Job Template attributes in the Create-Job Request, the Printer uses its default value attributes when processing the job. Since a Print-Job operation is used for a Job with only one Document, the Document attributes "document-name" and "document-format" are also supplied by the client. "document-name" is MANDATORY; "document-format" is OPTIONAL.

Document Content:
The client supplies the document data to be processed.

The simplest Print-Job Request consists of just the Document Content and nothing else. In this case, the Printer creates a new Job object with no associated Job Template attributes and the job contains a single Document.

When a Printer receives a Print-Job Request, the Printer either accepts or rejects the request. The Printer accepts the Print-Job Request and creates a Job object if it is able to accept all Job Template attributes in the request. The Printer rejects the request and does not create a Job object if the Printer rejects any Job Template attribute in the request. There are six cases to consider when accepting or rejecting Job Template attributes:

1. The client supplies a Job Template attribute named "xxx" and the value supplied by the client is among the values supported by the Printer (i.e., is among the values of the Printer's "xxx-supported" attribute): The "xxx" Job Template attribute is accepted. The Printer creates the Job object and associates the "xxx" attribute with the new Job object using the value supplied by the client.

2. The client supplies a Job Template attribute but the attribute is syntactically bad: The Printer SHALL reject the job and return name of the badly formed attribute (if known) in the "unsupported-attributes" response parameter.

3. The client supplies a Job Template attribute and the attribute value is not among the values supported by the Printer: This case depending on the value of the "best-effort' attribute (see

Section 4.2.8), tIf the client supplies a "best-effort" of
'false' (or supplies no "best-effort" attribute and the Printer's
default behavior attribute is set to 'false') the Printer SHALL

reject the Job and return the 'client-error-attribute-unsupported" error code and the unsupported attribute in the "unsupported-attributes " output parameter. If the client supplies a "best-effort" of 'true' (or supplies no "best-effort" attribute and the Printer's default behavior attribute is set to 'true') the Printer SHALL accept the Job and substitute supported values for all unsupported values supplied by the client. In this case, if everything else is ok, the Printer returns a "successful-ok" status code. The client must query the newly create Job object to find out if any of the requested values have been modified.

4. The client supplies a Job Template attribute and the Printer does not support the attribute: The Printer rejects the attribute. The Printer returns the 'client-error-attribute-unsupported' error code and the rejected attribute in the "unsupported-attributes" output parameter.

5. The client does not supply a Job Template attribute, but the Printer supports the attribute: The attribute is accepted and when the Printer creates the Job object, the Printer SHALL NOT associate the attribute with the new Job object using Printer's default value attribute. When the Printer processes that Job, the Printer uses the behavior implied by the default value Printer attribute as set at the time of Job processing (not Job creation). In other words, these rules allow for a Job object to be created without implementing some of the Job Template attributes. As the Printer processes the Job, if the Printer supports a corresponding default value attribute for the missing Job Template attribute, the Printer uses the default value.

6. The client does not supply an attribute, and the Printer does not support the attribute: The Printer accepts the Job. However, as far as IPP is concerned, the result of processing that Job (with respect to the missing attributes) is undefined. In many cases though, this case represents a legacy environment, and even without IPP attributes, the job will be processed successfully. For example, the document data might have been generated by a device-specific printer driver that formats the job and emits a data stream (a stream of PDL) that is finely tuned for the intended output device and its internal interpreter. But, both the output device and the printer driver are unaware of the IPP protocol, and an intermediate process (which is IPP aware) is able to submit the job and make sure that it is delivered to the

      output device, yet not be aware of all the Job Template
      attributes that might possibly be supported.

3.1.2.2 Print-Job Response

The Printer SHALL return to the client the following output parameters as part of the Print-Job Response:

Job Identifier ("job-uri"):
A URI which the client SHALL use for all other operations on this Job

Job Status Attributes:
This includes the following Job attributes: "job-name", "job-state", and "job-state-reasons". The value of each attribute is taken from a "snapshot" of the new Job object sometime after the time the Printer receives the print request until just prior to returning the response to the client. Since the "job-state-message" attribute is OPTIONAL, it MAY be included in the response, but it is NOT REQUIRED.

Note: Since any printer state information which affects a job's state is reflected in the "job-state" and "job-state-reasons" attributes, it is sufficient to return only these attributes and no specific printer status attributes.

Unsupported Attributes:
If there is an error, this output parameter contains a set of attributes that are unsupported. This output parameter is not used if the status code indicates that there were no errors.

The simplest response consists of the just the job identifier ("job-uri") and Job Status attributes output parameters with a status code of "successful-ok".

3.1.3 Print-URI Operation

This operation is identical to the Print-Job operation (section 3.1.2) except that a client supplies a reference (a URI) to the document data to be printed rather than the document data itself. It is up to the IPP server to interpret the URI and subsequently "pull" the document from the source referenced by the URI string.

3.1.3.1 Print-URI Request

The following elements are part of the Print-URI Request:

  Job Template Attributes:
     (see section 3.1.2.1)

  Document URI ("document-uri"):
     The client supplies a URI reference to the document data rather
     than the document data itself.

3.1.3.2 Print-URI Response

The Printer SHALL return to the client the following output parameters as part of the Print-URI Response:

  Job Identifier ("job-uri"):
     (see section 3.1.2.2)

  Job Status:
     (see section 3.1.2.2)

  Unsupported Attributes:
     (see section 3.1.2.2)

3.1.4 Validate-Job Operation

This operation is identical to the Print-Job operation (section 3.1.2) except that a client supplies no document data or any reference to document data and the Printer allocates no resources (i.e., a new Job object) to process the job. The VALIDATE request is only used to verify capabilities of a printer object against whatever input parameters are supplied in the Validate-Job request.

3.1.4.1 Validate-Job Request

The following elements are part of the Validate-Job Request:

  Job Template Attributes:
     (see section 3.1.2.1)

3.1.4.2 Validate-Job Response

The Printer SHALL return to the client the following output parameters as part of the Validate-Job Response:

  Unsupported Attributes:
     (see section 3.1.2.2)

Note: In this case, no "job-uri" or Job Status output parameters are returned.

3.1.5 Create-Job Operation

This operation is similar to the Print-Job operation (section 3.1.2) except that a client supplies no document data or any reference to document data in the Create-Job request. This operation is followed by one or more Send-Document or Send-URI operations. It is possible for a given implementation to only support either Send-Document or Send-URI but not both. In that case, a client SHOULD NOT use an unsupported operation. If a Printer supports the Create-Job operation, it MUST also support one of the Send-Document or Send-URI operations or both.

3.1.5.1 Create-Job Request

The following elements are part of the Create-Job Request:

  Job Template Attributes:
     (see section 3.1.2.1)

3.1.5.2 Create Job Response

The Printer returns to the client the following output parameters as part of the Create-Job Response:

  Job Identifier ("job-uri"):
     (see section 3.1.2.2)

  Job Status:
     (see section 3.1.2.2)

  Unsupported Attributes:
     (see section 3.1.2.2)

3.1.6 Send-Document Operation

Once a Job object has been created using a Create-Job operation (returning a "job-uri"), a client directs a Send-Document operation to the newly create Job object. The operation adds a new Document to the Job object. An entire document MUST be sent in a single Send-Document operation.

3.1.6.1 Send-Document Request

The client submits the request to a Job URI.

The following abstract data types are part of the Send-Document Request:

Document Attributes:
A set of Document Description attributes (section 4.4).

Last Document Flag ("last-document"):
This is a boolean flag that is set to 'true' if this is the last Document for the Job.

Document Content:
The client supplies the document data.

3.1.6.2 Send-Document Response

The following output parameters are part of the Send-Document Response:

Job Status:
(see section 3.1.2.2)

Unsupported Attributes:
(see section 3.1.2.2)

3.1.7 Send-URI Operation

This operation is identical to the Send-Document operation (see section 3.1.6) except that a client supplies a reference (a URI) to the document data to be printed rather than the document data itself. It is up to the IPP server to interpret the URI and subsequently "pull" the document from the source referenced by the URI string.

3.1.7.1 Send-URI Request

The client submits the request to a Job URI.

The following abstract data types are part of the Send-URI Request:

Document Attributes:
(see section 3.1.6.1)

Last Document Flag ("last-document"):
(see section 3.1.6.1)

Document Reference ("document-uri"):
The client supplies a URI reference to the document data.

3.1.7.2 Send-URI Response

The following output parameters are part of the Send-URI Response:

Job Status:
(see section 3.1.6.2)

Unsupported Attributes:
(see section 3.1.6.2)

3.1.8 Cancel Job Operation

This operation allows a user to cancel one specific Print Job any time after the print job has been established on the Printer. Some pages may be printed before a job is terminated if printing has already started when the Cancel Job operation is received. Only the end user who is also the job originator ("job-originating-user" Job attribute) can cancel the job using IPP 1.0.

3.1.8.1 Cancel-Job Request

The client submits the request to a Job URI.

The following abstract data types are part of the Cancel Job Request:

Message ("message"):
Optional message to the operator

3.1.8.2 Cancel-Job Response

There are no output parameters other Cancel Job Response other than the Status Code and optional Status Message.

3.1.9 Get-Attributes Operation

The Get-Attributes operation allows a client to obtain information from a Printer or Job object. The client supplies as an operation parameter the set of attribute names and/or attribute group names that the requester is interested in. The Printer returns a corresponding attribute set in the response with the appropriate attribute values filled in for each requested attribute (either explicitly named in the request or implicitly included by naming an attribute group).

3.1.9.1 Get-Attributes Request

When querying a Printer object, the client submits the Get-Attributes request to a Printer URI. When querying a Job object, the client submits the Get-Attributes request to a Job URI.The following input parameters are part of the Get-Attributes Request whether or not the request is to a Printer URI or a Job URI:

Requested Attributes ("requested-attributes") :
An optional set of attribute names (without values) or attribute group names in whose values the requester is interested. If the client omits this input parameter, the Printer SHALL respond as if this input parameter had been supplied with a value of 'all'.

Attributes may be requested by name or by group name. For Jobs, the attribute groups include:

- 'job-template': all of the Job Template attributes that apply to a Job object (the first column of the table in Section 4.2).
- 'job-description': the Job Description attributes in Section 4.3.

For Printers, the attribute groups include:

- 'job-template': all of the Job Template attributes that apply to a Printer object (the last two columns of the table in

Section 4.2).
- 'printer-description': the attributes specified in Section 4.5.

There are also special groups:

- 'none': no attributes of the specified object. 'none' is primarily useful in Get-Jobs, but can be used as a "ping" with the Get-Attributes operation.
- 'all': all supported attributes

It is NOT REQUIRED that a Printer support ALL attributes belonging to a group, however it is MANDATORY that a Printer implementation understand these group names.

Document Attributes ("document-name", "document-format", and "document-uri", see Section 4.4) may also be requested either individually or as a group named 'document-attributes'. If any or all of these attributes are requested, in the response, they are treated as multi-valued attributes (one value for each Document in the Job). If there are 5 Documents in the Job, then each returned Document attribute MUST have 5 values: the set of first values from each attribute correspond to the first Document, the set of second values correspond to the second Document, and so on. In order to maintain the same number of values in each returned attribute, if there is no value for the "document-format" attribute for one of the Documents, the value 'other' is returned in its place, and if there is no value for the "document-uri" attribute for one of the Documents, the special URI 'none' is returned in its place. For example if all Document attributes are requested, and there are 3 Documents is a Job, the returned attributes might look like:

```
document-name = 'Document 1', 'Document 2', Document 3'
document-format = 'langPS', 'other', 'langPCL'
document-uri = ' ftp://some.domain.com/file1.ps', 'none', 'none'
```

The following input parameters are part of the Get-Attributes Request only when querying a Printer:

Document Format ("document-format") :

This input parameter conditions the Printer attributes and values that might depend on the document format. The Printer SHALL return only (1) those attributes that are supported and (2) the attribute values that are supported for the specified document format. By specifying the document format, the client can eliminate the attributes and values that are not supported for a specific document format. For example, a Printer might have multiple interpreters to support both 'langPS' (for PostScript) and 'langPCL' (for PCL) documents. However, for only one of those interpreters might the Printer be able to support "number-up" with values of 'one', 'two', and 'four'. For the other interpreter it might be able to only support "number-up" with a value of 'one'.

If the client omits this input parameter, the Printer SHALL respond as if the input parameter had been set to the value of the Printer's default value "document-format" attribute were supplied. It is recommended that the client always supply a value for document-format, since the Printer's default value for document-format may be 'langAutomatic', in which case the returned attributes and values are for the union of the document formats that the Printer can automatically sense.

If this input parameter is sent in a Get-Attribute Request sent to a Job URI, the request is rejected with a status code of 'client-error-not-possible'.

3.1.9.2 Get-Attributes Response

The Printer returns the following response parameters as part of the Get-Attributes Response:

Requested Attributes:
This is a the set of requested attributes and their current values. The Printer ignores (does not respond with) any requested attribute which is not supported.

3.1.10 Get-Jobs Operation

The Get-Jobs operation allows a client to retrieve list of Jobs belonging to the target Printer object. The client may also supply a list of Job attribute names or attribute group names. These Job attributes will be returned for each Job that is returned.

This operation is like Get-Attributes, except that Get-Jobs operation returns attributes from more than one object.

3.1.10.1 Get-Jobs Request

The client submits the Get-Jobs request to a Printer URI.

The following input parameters are part of the Get-Jobs Request:

Limit ("limit"):
This is an integer value that indicates a limit to the number of Jobs returned. The limit is a "stateless limit" in that if the limit is n then only the first n jobs are returned in the Get-Jobs Response; there is no mechanism to allow for the "next" n jobs. The limit applies across all Job States requested. For example, if the limit if 50, and there are 75 jobs in the 'completed' state and 25 in the 'pending state' and the client requests first 'completed jobs' and then 'pending' jobs, only the oldest 50 'completed' jobs are returned. The other 25 'completed' jobs are not returned and neither are any of the 'pending' jobs returned.

Requested Job Attributes ("requested-attributes"):
An optional set of Job attribute names or attribute groups names in whose values the requester is interested. This set of attributes is returned for each Job that is returned.. The attribute group names are the same as for the Get-Attributes operation for the Job object. If the client omits this input parameter, the Printer SHALL respond as if this input parameter had been supplied with a value of " 'job-uri'.

3.1.10.2 Get-Jobs Response

The Printer SHALL return the following output parameters as part of the Get-Jobs Response:

Req   uested Attributes:
The result includes zero or more Job objects each with zero or more attributes. Jobs are returned in the following order: First all active Jobs (Jobs in the 'pending', 'processing', 'pending-held', and 'processing-stopped' states) are returned oldest to newest (with respect to expected completion time) followed by all completed Jobs (Jobs in the 'completed', 'aborted', or 'canceled' states) newest to oldest (with respect to actual completion time). Jobs that are in the 'pending-held' state SHALL appear in their position as if they were 'pending' (otherwise, a user might be confused by Jobs that move from 'pending-held' to 'pending' as seeming to jump ahead in the queue).

3.2 Operation Status Codes and Messages

An operation status code provides information on the processing of a
request. A message provides a short textual description of the status

of the operation. The status code is intended for use by automata and a status message is intended for the human user. An IPP application (i.e. a browser, GUI, print driver or gateway) is not required to examine or display the message. Status codes and suggested corresponding status messages are described in section 11..

4. Object Attributes

This section describes the attributes with their corresponding syntaxes and values that are part of the IPP model. The sections below show the objects and their associated attributes which are included within the scope of this protocol. Many of these attributes are derived from other relevant specifications:

- ISO/IEC 10175 DPA (Final, June 1996) [5]
- RFC 1759 Printer MIB (Proposed Standard, May 1995) [1]
- Internet-Draft: Printer MIB (Draft Standard in progress, July 1997) [29]
- Internet-Draft: Job Monitoring MIB (I-D in progress, June 1997) [27]

Each attribute is uniquely identified in this document using a "keyword" (see section 10.2.1). The keyword is included in the section header describing that attribute. Not only are attributes uniquely identified with keywords, some attributes are defined to have a syntax which is a set of keywords.

4.1 Attribute Syntaxes

The following table shows the basic syntax types that a client and server SHALL be able to handle.

text: a sequence of characters, length: 0 to 4095, UTF8 characters. This syntax type is used for free form human readable text intended for human consumption.

name: a sequence of characters, length: 1 to 255, UTF8 characters. This syntax type is used for referencing some object or entity via a user-friendly string, such as a Printer name, a document name, a user name, or a host name.

fileName: a sequence of characters, length: 1 to 1024, UTF8 characters. This syntax type is used for referencing some file.

     The limit is the same as in POSIX and Microsoft NT.

keyword: a sequence of characters, length: 1 to 255, containing only the characters ASCII lowercase letters ("a" - "z"), ASCII digits ("0" - "9"), hyphen ("-"), and underscore ("_"). The first character MUST be an ASCII lowercase letter. This syntax type is used for enumerating semantic identifiers of entities in the abstract protocol (specified in this document). These entities can be attribute names or values of attributes. When a keyword is used to represent an attribute (its name), it MUST be unique within the full scope of IPP objects and attributes. When a keyword is used to represent a value of an attribute, it MUST be unique just within the scope of that attribute. That is, the same keyword can not be used for two different values within the same attribute to mean two different semantic ideas. However, the same keyword can be used across two or more attributes, representing different semantic ideas for each attribute.

enum: an enumerated integer value that is in the range from -2**31 to 2**31 - 1.   Each value has an associated keyword name. Each attribute (whose syntax is enum) enumerates the values that are defined for the attribute. The enum type is used for attributes for which there are enum values assigned by other standards, such as SNMP MIBs. A number of attribute enum values in this specification are also used for corresponding attributes in the IETF Printer MIB [1] and the Job Monitoring MIB [27]. Enums are not used for attributes to which the system administrator may assign values. Values in the range 2**30 to 2**31 - 1 are reserved for private or experimental use. Implementers are warned that use of such values may conflict with other implementations. Implementers are encouraged to request registration of enum values following the procedures in Section 6.

uri: a sequence of characters as defined in rfc1738 and rfc1808. This syntax type is used for carrying Universal Resource Identifiers.

uriScheme: a sequence of characters representing the URI Scheme. These include 'http' for HTTP schemed URIs (e.g., http://...), and 'ftp' for FTP schemed URIs (e.g., ftp://...).

locale: a standard identifier for human language and optionally a country. The values for this syntax type are taken from RFC 1766 [26]. RFC 1766 does not have provision for expressing the coded character set component of a locale. The coded character set used in the IPP protocol SHALL be UTF-8 [28].

ISSUE: The term 'locale' usually includes country, language, and coded character set. But our data type and RFC 1766 do not include coded character set. Should we change the name from 'locale' to 'human-language' and the corresponding attributes from "user-locale" to "user-language", and "printer-locale" to "printer-language" (though printer language is what IANA registers for our document formats)?

octetString: a sequence of octets. This syntax type is used for opaque data, such as the document-content.

boolean: two values of 'true' and 'false'. This syntax type is like a keywordSet, but there are only two values. Note: An application might use a checkbox for an attribute with this syntax type.

integer: an integer value that is in the range from -2**31 to 2**31 - 1. Each attribute specifies the range constraint explicitly if the range is different from the full range of possible integer values (e.g., 0 - 100 for the "job-priority" attribute).

dateTime: a standard, fixed length representation of date and time (to the nearest second) as defined in RFC 1123 [27]. For example, Sun, 06 Nov 1994 08:49:37 GMT. This is a fixed-length subset of that defined by RFC 1123 (an update to RFC 822). All values MUST be represented in Greenwich Mean Time (GMT). This is indicated by the inclusion of "GMT" as the three-letter abbreviation for time.

seconds: a non-negative integer with implicit units of seconds. This is used for relative time.

milliseconds: a non-negative integer with implicit units of milliseconds. This is used for relative time.

1setOf X: 1 or more values of type X. This syntax type is used for multi-valued attributes, whose value is a set of values. Note: The syntax type is called "1setOf" to indicate that set of values SHALL NOT be empty (a set of size 0).

rangeOf X: a range of value of type X. This syntax type is used for ordered values (numeric, lexical, etc.) such as integers.

4.1.1 Attribute Extensibility

This document uses prefixes to the "keyword" and "enum" basic syntax type in order to communicate extra information to the reader through its name. This extra information need not be represented in an implementation because it is unimportant to a client or Printer. The table below describes the prefixes and their meaning.

"type1": The IPP standard must be revised to add a new keyword or a new enum. No private keywords or enums are allowed.

"type2": Implementers can, at any time, add new keyword or enum values by proposing them to the IPP working group for registration (or an IANA-appointed registry advisor after the IPP working group is no longer certified) where they are reviewed for approval. IANA keeps the registry.

"type3": Implementers can, at any time, add new keyword and enum values by submitting a registration request directly to IANA, no IPP working group or IANA-appointed registry advisor review is required.

"type4": Anyone (system administrators, system integrators, site managers, etc.) can, at any time, add new installation-defined values (keywords or new enum values) to a local system. Care SHOULD be taken by the implementers to see that keywords do not conflict with other keywords defined by the standard or as defined by the implementing product. There is no registration or approval procedure for type 4 keywords.

Each of the four types above assert some sort of registry or review process in order to be valid extensions. "type1" extensions are only valid if the specification is updated, "type2" extensions are only valid if the IPP working group or an IANA approved review process approves them, "type3" extensions are only valid if IANA registers the value with no review process required, and "type4" extensions are always valid (there is no review or registration process required). Any typeN value MAY be registered using a process for some typeM where M is less than N, however such registration is NOT REQUIRED. For example, a type4 value MAY be registered in a type 1 manner (by being included in a future version of an IPP specification) however it is NOT REQUIRED.

This specification defines keyword and enum values for all of the above types, including type4 keywords.

For private (unregistered) keyword extensions, implementers SHOULD use keywords with a suitable distinguishing prefix, such as "xxx-" where xxx is the (lowercase) company name registered with IANA for use in domain names [30].

Note: RFC 1035 [30] indicates that while upper and lower case letters are allowed in domain names, no significance is attached to the case.

That is, two names with the same spelling but different case are to be treated as if identical. Also, he labels in a domain name must follow the rules for ARPANET host names: They must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphen. Labels must be 63 characters or less.

ISSUE: Since "." is allowed in fully qualified domain names we must state that "." gets mapped to "_" for keywords or we must allow for "." in keywords. Also, why do we not allow uppercase in keywords? Should we make keyword be much more broad (all printable ASCII) now that we have binary delimeters?

For private (unregistered) enum extension, implementers SHOULD values in the reserved integer range (see "enum").

4.2 Job Template Attributes

Job Template attributes describe job processing behavior. Take for example, a generic Job Template attribute called "xxx":

1. "xxx" is optionally supplied by the client in a create request. If "xxx" is supplied, the client is specifying that the Printer SHALL apply a specific job processing behavior to this job while processing the Job. When "xxx" is not supplied by the client, the the Printer applies thedefault job processing behavior.

2. "xxx-supported" is a Printer attribute that describes which behaviors are supported by a Printer. "xxx-supported" is a CONDITIONALLY MANDATORY attribute, that is the "xxx-supported" attribute is MANDATORY if the Printer is capable of realizing one or more of the behaviors associated with the attribute and its values. A client can query the Printer and find out what behaviors are supported by inspecting the values of the "xxx-supported" attribute.

3. The Printer also supports a default value attribute named "xxx". This default value attribute describes what will be done when no other job processing information is supplied by the client (either explicitly as an IPP attribute in the create request or implicitly as an embedded instruction within the job data). Along with the supported attribute, the default value attribute is also CONDITIONALLY MANDATORY. However, if the Printer

supports either the "xxx" default value attribute or the "xxx-supported" attribute, the Printer MUST support the other and vice versa.

4. If a client application wishes to present an end user with a list of supported and default values from which to choose, the client program SHOULD query the Printer for the supported and default value attributes. The values that the client then supplies in the create request will all fall within the supported values of the Printer. When querying the Printer, the client MAY enumerate each attribute by name in the Get-Attributes Request, or the client MAY just name the "printer-job-template" group in order to get the complete set of supported and default value attributes which are supported.

The "job-priority" attribute is an example of a Job Template attribute. It is an integer in the range from 1 to 100. A client can query the Printer for the "job-priority-supported" attribute and the "job-priority" default value attribute. The supported attribute contains a range of supported priority values. The default value attribute contains the job priority value that will be used for a new job if the client does not supply one in the create request. If the client does supply the "job-priority" attribute, the Printer validates the value to make sure that it falls within the range of supported values. If the client-supplied value is supported, the Job object is created and the "job-priority" attribute is populated with that value. The Job object, when queried, returns the value supplied by the client. If the client does not supply a "job-priority" value in the create request, the Job object is created, but no "job-priority" attribute is associated with the Job. The client queries the Printer's default value "job-priority" value to find out at what priority the job will be processed.

The table below summarizes the names, relationships, and conformance requirements for all Job Template attributes. The following general rules apply to implementation requirements:

1. In a create request, all Job Template attributes are OPTIONAL.

2. In a Printer Object, all supported attributes ("xxx-supported") are CONDITIONALLY MANDATORY.

3. All Printer default value attributes ("xxx") are CONDITIONALLY MANDATORY.

Note: If the Printer implements either the default value attribute or the supported values attribute, the Printer MUST also implement the other and vice versa.

The table only shows exceptions to the above rules.   The first column
of the table (Job) shows the name and syntax for each Job Template

attribute in the Job object (in the create request, the same name and syntax is used). All of the attributes in the first column make up the "job-template" group. The last two columns show the name and syntax for each Job Template attribute in the Printer object (the default value attribute and the supported attribute). A "No" in the table means the Printer SHALL NOT support the attribute. A "MAN" indicates that the attribute is MANDATORY.

Job	Printer: Default Value	Printer: Supported
job-name (name, MAN)	No	No
job-sheets (type4 keyword)	job-sheets (type4 keyword)	job-sheets-supported (1setOf type4 keyword)
notify-events (1setOf type2 keyword)	notify-events (1setOf type2 keyword)	notify-events-supported (1setOf type2 keyword)
notify-addresses (1setOf uri)	No	notify-addresses -supported (1setOf uri scheme)
job-priority (integer 1-100)	job-priority (integer 1-100)	job-priority-supported (rangeOf integer 1-100)
job-hold-until (type4 keyword)	job-hold-until (type4 keyword)	job-hold-until-supported (1setOf type4 keyword)
multiple-document-handling (type2 keyword)	multiple-document-handling (type2 keyword)	multiple-document-handling-supported (1setOf type2 keyword)
best-effort (boolean)	best-effort (boolean, MAN)	best-effort-supported (boolean, MAN)
media (type4 keyword)	media (type4 keyword)	media-supported (1setOf type4 keyword)
number-up (type3 keyword)	number-up (type3 keyword)	number-up-supported (1setOf type3 keyword)
sides	sides	sides-supported

(type2 keyword)	(type2 keyword)	(1setOf type2 keyword)
printer-resolution	printer-resolution	printer-resolution-

(type2 enum)	(type2 enum)	supported (1setOf type2 enum)
print-quality (type2 enum)	print-quality (type2 enum)	print-quality-supported (1setOf type2 enum)
finishings (1setOf type2 enum)	finishings (1setOf type2 enum)	finishings-supported (1setOf type2 enum)
copies (integer: 1 - MAX)	copies (integer: 1 - MAX)	copies-supported (rangeOf integer 1- MAX)
document-format (type2 enum)	document-format (type2 enum)	document-format-supported (1setOf type2 enum)
compression (type3 keyword)	No	compression-supported (1setOf type3 keyword)
job-k-octets (integer)	No	job-k-octets-supported (rangeOf integer)
job-impressions (integer)	No	job-impressions-supported (rangeOf integer)
job-media-sheets (integer)	No	job-media-sheets-supported (rangeOf integer)

4.2.1 job-name (name)

This attribute is the name of the job. It is a name that is more user friendly than the "job-uri" attribute value. It does not need to be unique.

If "job-name" is not supplied in the create request, the Printer, on creation of the Job, SHALL generate a name. The name MAY be generated using the name of the first Document in the Job (the "document-name" attribute). If "job-name" is supplied in the create request, the Printer SHALL use its value as the name of the created Job.

4.2.2 job-sheets (type4 keyword)

This attribute determines which if any banner page(s) SHALL be printed with a job.

Standard values are:

  'none': no job sheet is printed
  'standard': a site specific standard job sheet (start only) or sheets (start and end) is printed

To force no job sheets, the system administrator SHALL set the supported value to only 'none'. To force the use of banner pages, the supported values SHALL not include 'none'. In this case, if a client requests 'none', the create request is rejected.

4.2.3 notify-events (1setOf type2 keyword)

This attribute specifies the events for which the end user desires some sort of notification. The "notify-addresses" attribute is used to describe the destination addresses for these events.

Standard values are:

  'none': the Printer SHALL not notify.
  'all': the Printer SHALL notify when any of the events occur.
  'job-completion': the Printer SHALL notify when the job containing this value completes (i.e., enters the 'completed', 'canceled', or 'aborted' state) with or without errors.
  'job-problems': the Printer SHALL notify when this job has a problem while this job is processing (i.e., when the Job moves from the 'processing' to the 'processing-stopped' state). Problems include any of the "job-state-reasons" or "printer-state-reason" values.
  'printer-problems': the Printer SHALL notify when this job is affected by a Printer problem. This happens when the printer enters the 'stopped' state while this job is in the 'pending', 'pending-held', 'processing', or 'processing-stopped' state. If the Printer enters the 'stopped' state the reason in the "printer-state-reasons" attribute.

4.2.4 notify-addresses (1setOf uri)

This attribute describes both where (the address) and how (the

mechanism for delivery) events are to be delivered. The Printer SHALL
use this attribute as the set of addresses and methods for sending

messages when an event occurs that the end user (job submitter) has registered an interest in.

Standard uri scheme values are:

'mailto': email is used
'http': an HTTP method is used to add HTML formatted events to the end of the specified HTML file.
'ftp': FTP is used to append a record at the end of a specified text file.

4.2.5 job-priority (integer(1:100))

This attribute specifies a priority for scheduling the print-job. A higher value specifies a higher priority. The value 1 is defined to indicate the lowest possible priority. The value 100 is defined to indicate the highest possible priority. Among those jobs that are ready to print, a Printer SHALL print all jobs with a priority value of n before printing those with a priority value of n-1 for all n. The mapping of vendor-defined priority over this range is implementation-specific.

4.2.6 job-hold-until (type4 keyword)

This job attribute specifies the named time period during which the Job print job SHALL become a candidate for printing.

Standard values for named time periods are:

'no-hold': immediately, if there are not other reasons to hold the job.
'day-time': during the day.
'evening': evening
'night': night
'weekend': weekend
'second-shift': second-shift
'third-shift': third-shift (after midnight)

An administrator SHALL associate allowable print times with a named time period (by means outside IPP 1.0). An administrator is encouraged to pick names that suggest the type of time period.

If the value of this attribute specifies a time period that is in the future, the Printer SHALL add the 'job-hold-until-specified' value to the job's "job-state-reasons" attribute, move the job to the 'pending-held' state, and SHALL NOT schedule the job for printing until the specified time-period arrives. When the specified time period arrives, the Printer SHALL remove the 'job-hold-until-specified' value from the job's "job-state-reason attribute" and, if no other job reasons remain, SHALL consider the job as a candidate for processing by moving the job to the 'pending' state.

If this job attribute value is the named value 'no-hold', or the time period has already started , the job SHALL be a candidate for processing immediately.

4.2.7 multiple-documenthandling (type2 keyword)

This job attribute is relevant only if a job consists of two or more documents. It controls finishing operations, job-sheet placement, and the order of documents when the copies attribute exceeds 1.

Standard values are:

'single-document': If the files for the job are a and b, then files a and b SHALL be treated as a single document for finishing operations. Also, there SHALL be no slip sheets between files a and b and the Printer SHALL NOT force each document to start on a new page or new media sheet. If more than one copy is made, the ordering SHALL be a, b, a, b, ..., and the Printer SHALL force each copy to start on a new sheet.
'separate-documents-uncollated-copies': If the files for the job are a and b, then each file SHALL be treated as a single document for finishing operations. Also, a client may specify that a slip sheet be placed between files a and b and the Printer shall force each document copy to start on a new sheet. If more than one copy is made, the ordering SHALL be a, a, b, b, ...
'separate-documents-collated-copies': If the files for the job are a and b, then each file SHALL be treated as a single document for finishing operations. Also, a client may specify that a slip sheet be placed between files a and b. If more than one copy is made, the ordering SHALL be a, b, a, b, ..., and the Printer shall force each document copy to start on a new sheet

4.2.8 best-effort (boolean)

This attribute determines how to control a conflict between what a client requests in a create request and what a Printer supports. The value 'true' means that a best effort attempt to print the Job is possible (or acceptable). In order to achieve this, the Printer might have to substitute some supported value for a requested value which is unsupported. The value 'false' means that total fidelity is required;

a best effort attempt to print the Job is not possible (or not acceptable). In other words, the job must be printed exactly as specified in the create request. If one or more of the client-supplied values in the create request is not supported by the Printer, the Printer rejects the create request.

There are two cases to consider:

- The Printer's "best-effort-supported" attribute is set to 'true': This indicates that the Printer is capable of best effort printing. In this case either the "best-effort" Job Template attribute in the create request or the Printer's "best-effort" (the default value attribute) can be set to either 'true' or 'false'. If "best-effort" is set to 'false' the Printer ensures full fidelity of the other IPP attributes in the create request. That is, if any other attribute in the create-request is set to a value that is not supported, the Printer rejects the create request. If "best-effort" attribute is set to 'true', the the Printer makes whatever substitutions are necessary to ensure that the job is printed. For example, if a client supplies a "finishings" Job Template attribute set to 'staple' but the printer does not support stapling (not a feature or it is temporarily out of staples) then if the "best-effort" attribute is set to 'true' the job can still be printed but it is not stapled. If "best-effort" is set to 'false', the create request is rejected since the Printer can not guarantee that the job will be stapled.

- The Printer's "best-effort-supported" attribute is set to 'false': This indicates that the Printer is not capable of (or will not support) best effort printing. The Job Template attributes supplied by the client in the create request must match all of the Printer's supported values, or else the Printer rejects the create request. In this case Printer's "best-effort" attribute (the default value attribute) MUST also be set to 'false'. If the client supplies the "best-effort" Job Template create request, it too MUST be set to 'false'. If it is set to 'true', the Printer rejects the create request.

Note: that the "best-effort" attribute in a create request is unlikely to be used much. Many clients will submit a job with no attributes, and the Printer will use default values. Other clients will submit a job via a GUI that limits the attribute values to values which are supported. Best-effort is useful in the GUI context only if a user

expects the job to be moved to another printer and prefers a sub-
optimal result to nothing at all. Best-effort is most useful in the
case where an end-user uses a command line interface to request
attributes that might not be supported.

4.2.9 media (type4 keyword)

This job attribute identifies the medium that the Printer uses for all pages of the Job.

The values for "media" include medium-names, medium-sizes, input-trays and electronic forms so that one attribute specifies the media. If a printer allows a client to specify a medium name as the value of this attribute, such a medium name implicitly selects an input-tray that contains the specified medium. If a printer allows a client to specify a medium size as the value of this attribute, such a medium size implicitly selects a medium name which in turn implicitly selects an input-tray that contains the medium with the specified size. If a printer allows a client to specify an input-tray as the value of this attribute, such an input-tray implicitly selects the medium that is in that input-tray at the time the job prints. This case includes manual-feed input-trays. If a printer allows a client to specify an electronic form as the value of this attribute, such an electronic form implicitly selects a medium-name which in turn implicitly selects an input-tray that contains the medium specified by the electronic form. The electronic form also implicitly selects an image that the Printer SHALL merge with the data from the document as its prints each page.

Standard values are (taken from ISO DPA and the Printer MIB) and are listed in section 13.

4.2.10 number-up (type3 keyword)

This job attribute specifies the number of source page-images to impose upon a single side of an instance of a selected medium.

Standard values are:

'none': The Printer SHALL not include any embellishments and SHALL place one logical page on a single side of an instance of the selected medium without any translation, scaling, or rotation.
'one': The Printer SHALL place one logical page on a single side of an instance of the selected medium (MAY add some sort of translation, scaling, or rotation).
'two': The Printer SHALL place two logical page on a single side of an instance of the selected medium (MAY add some sort of translation, scaling, or rotation).
'four': The Printer SHALL place four logical page on a single side

of an instance of the selected medium (MAY add some sort of
translation, scaling, or rotation).

This attribute primarily controls the translation, scaling and rotation of page images, but a site may choose to add embellishments, such as borders to each logical page.

4.2.11 sides (type2 keyword)

This attribute specifies how source page-images are to be imposed upon the sides of an instance of a selected medium.

The standard values are:

'one-sided': imposes each consecutive source page-image upon the same side of consecutive media sheets.
'two-sided-long-edge': imposes each consecutive pair of source page-image upon front and back sides of consecutive media sheets, such that the orientation of each pair of source-pages on the medium would be correct for the reader as if for binding on the long edge. This imposition is sometimes called 'duplex'.
'two-sided-short-edge': imposes each consecutive pair of source page-image upon front and back sides of consecutive media sheets, such that the orientation of each pair of source-pages on the medium would be correct for the reader as if for binding on the short edge. This imposition is sometimes called 'tumble' or 'head-to-toe'.

'two-sided-long-edge', 'two-sided-short-edge', 'tumble'. 'duplex', and 'head-to-toe' all work the same for portrait or landscape, that is, 'head-to-toe' is 'tumble' in portrait but 'duplex' in landscape. 'head-to-head' also switches between 'duplex' and 'tumble' when using portrait and landscape modes.

4.2.12 printer-resolution (type2 enum)

This attribute identifies the resolution that Printer uses for a certain Job.

The values are type2 enums which represent single integers or pair of integers. The latter are to specify the resolution when the x and y dimensions differ. When two integers are specified, the first is in the x direction, i.e., in the direction of the shortest dimension of the medium, so that the value is independent of whether the printer feeds long edge or short edge first.

The standard values are:

```
  'normal'(3):
```

```
  'res-100'(4):  100 x 100 dpi
  'res-200'(5):  200 x 200 dpi
  'res-240'(6):  240 x 240 dpi
  'res-300'(7):  300 x 300 dpi
  'res-360'(8):  360 x 360 dpi
  'res-400'(9):  400 x 400 dpi
  'res-600'(10):  600 x 600 dpi
  'res-720'(11):  720 x 720 dpi
  'res-800'(12):  800 x 800 dpi
  'res-1200'(13):  1200 x 1200 dpi
  'res-1440'(14):  1440 x 1440 dpi
  'res-1600'(15):  1600 x 1600 dpi
  'res-1800'(16):  1800 x 1800 dpi
  'res-100x200'(100):  100 x 200 dpi
  'res-200x100'(101):  200 x 100 dpi
  'res-300x600'(102):  300 x 600 dpi
  'res-600x300'(103):  600 x 300 dpi
  'res-360x720'(104):  360 x 260 dpi
  'res-720x360'(105):  720 x 360 dpi
  'res-400x800'(106):  400 x 800 dpi
  'res-800x400'(107):  800 x 400 dpi
  'res-600x1200'(108):  600 x 1200 dpi
  'res-1200x600'(109):  1200 x 600 dpi
  'res-720x1440'(110):  720 x 1440 dpi
  'res-1440x720'(111):  1440 x 720 dpi
  'res-1800x600'(112):  1800 x 600 dpi
```

ISSUE: Should we add metric enums? Such as

```
  'res-8x3.85'(200):  8 lines per 3.85 mm (fax)
  'res-8x7.7'(201): 8 lines per 7.7 mm (fax)
  'res-8x15.4'(202): 8 lines per 15.4 mm (fax)
  'res-16x15.4'(203): 16 lines per 15.4 mm (fax)
```

4.2.13 print-quality (type2 enum)

This attribute specifies the print quality that the Printeruses for a certain Job.

The standard values are:

```
  'draft'(3): lowest quality available on the printer
  'normal'(4): normal or intermediate quality on the printer
```

  'high'(5): highest quality available on the printer

4.2.14 copies (integer(1:2**31 - 1))

This attribute specifies the number of copies of the job to be printed.

Note: The effect of this attribute on job with multiple documents is controlled by the "multiple-document-handling" job attribute (section 4.2.7).

4.2.15 finishing (1setOf type2 enum)

This attribute identifies the finishing operations that the Printer uses for each copy of each printed document in a particular Job. The definition of a copy for Jobs with multiple documents is controlled by the "multiple-document-handling" attribute.

Standard values are:

'none'(3): Perform no finishing
'staple'(4): Bind the document(s) with one or more staples. The exact number and placement of the staples is site-defined.
'staple-top-left'(5): Place one or more staples on the top left corner of the document(s).
'staple-bottom-left'(6): Place one or more staples on the bottom left corner of the document(s).
'staple-top-right'(7): Place one or more staples on the top right corner of the document(s).
'staple-bottom-right'(8): Place one or more staples on the bottom right corner of the document(s).
'saddle-stitch'(9): Bind the document(s) with one or more staples (wire stitches) along the middle fold. The exact number and placement of the stitches is site-defined.
'edge-stitch'(10): Bind the document(s) with one or more staples (wire stitches) along one edge. The exact number and placement of the staples is site-defined.
'punch'(11): This value indicates that holes are required in the finished document. The exact number and placement of the holes is site-defined The punch specification MAY be satisfied (in a site- and implementation-specific manner) either by drilling/punching, or by substituting pre-drilled media.
'cover'(12): This value is specified when it is desired to select a non-printed (or pre-printed) cover for the document. This does not supplant the specification of a printed cover (on cover stock

medium) by the document itself.

'bind'(13): This value indicates that a binding is to be applied to the document; the type and placement of the binding is site-defined."

4.2.16 document-format (type2 keyword)

This attribute defines the document format for each Document in a Job. The standard values for this attribute are enums. Since the complete list is rather long, the full enumeration of standard values is found in section 12 APPENDIX C: "document-format" enum values.

If the "document-format" is unknown for a certain document, the client SHALL NOT supply the attribute in the create request or the Send-Document Request.

4.2.17 compression (type3 keyword)

This attribute identifies compression algorithms used for compressed document data.

Standard values for this attribute are:

'none': no compression is used.
'zip':ZIP compression technology
'tar': UNIX TAR compression technology

4.2.18 job-k-octets (integer(0:2**31 - 1))

This attribute specifies the total size of the job in K octets, i.e., in units of 1024 octets. The value SHALL be rounded up, so that a job between 1 and 1024 octets SHALL be indicated as being 1, 1025 to 2048 SHALL be 2, etc.

Note: This attribute and the following two attributes ("job-impressions" and "job-media-sheets") are not intended to be counters; they are intended to be useful routing and scheduling information if known. The Printer tries to compute the value if it is not supplied in the create request. The Printer, however, might not be able to compute this value at the time the Job is created. If not, the Printer may support this attribute at any later time as it is able to compute value.

4.2.19 job-impressions (integer(0:2**31 - 1))

This attribute specifies the total size of the job in impressions.

4.2.20 job-media-sheets (integer(0:2**31 - 1))

This attribute specifies the total size of the job in media-sheets.

4.3 Job Description Attributes

The attributes in this section form the attribute group called "job-description". The following table summarizes these attributes. The third column indicates whether the attribute is a MANDATORY attribute. If it is not MANDATORY, then it is OPTIONAL.

Attribute	Syntax	MANDATORY?
job-uri	uri	MANDATORY
job-uri-user	uri	
job-originating-user	name	MANDATORY
job-originating-host	name	
user-locale	locale	
job-state	type1 enum	MANDATORY
job-state-reasons	1setOf type2 keyword	
job-state-message	text	
output-device-assigned	name	
time-since-pending	milliseconds	MANDATORY
time-since-processing	milliseconds	MANDATORY
time-since-completed	milliseconds	MANDATORY
number-of-intervening-jobs	integer	MANDATORY
job-message-from-operator	text	
job-k-octets-processed	integer	
job-impressions-completed	integer	
job-media-sheets-completed	integer	

4.3.1 job-uri (uri)

This attribute contains the URI for the job. The Printer, on receipt of a new job, generates a URI which identifies the new Job on that Printer. The Printer returns the value of the "job-uri" attribute as part of the response to a create request.   The precise format of a job URI is implementation dependent.

4.3.2 job-uri-user (uri)

Similar to "job-uri", this attribute contains the URI referencing an HTML page containing information about the Job.

4.3.3 job-originating-user (name)

This attribute specifies the user name of the person submitting the print job. The Printer SHALL set this attribute to the most authenticated name that it can obtain from the protocol over which the operation was received from the client.

4.3.4 job-originating-host (name)

This attribute identifies the originating host of the job. The Printer SHALL set this attribute to the most authenticated host name it can obtain from the protocol over which the operation was received from the client.

4.3.5 user-locale (locale)

This attribute identifies the human language and optional the country of the end user. The Printer sets this attribute to the most reliable value it can obtain from the protocol over which the Print operation was received from the client.

The Printer uses this attribute to determine the locale it SHOULD use for localizing any text strings that it sends back to the end user. These include status messages, text attributes, and notification messages.

4.3.6 job-state (type1 enum)

This attribute identifies the current state of the job. Even though the IPP protocol defines eight values for job states, implementations only need to support those states which are appropriate for the particular implementation. In other words, a Printer supports only those job states implemented by the output device and available to the Printer object implementation.

Standard values are:

'unknown'(2): The job state is not known, or its state is indeterminate.
'pending'(3): The job is a candidate to start processing, but is

```
     not yet processing.
  'pending-held'(4):  The job is not a candidate for processing for
     any number of reasons but will return to the 'pending' state as
```

soon as the reasons are no longer present. The job's "job-state-reason" attribute SHALL indicate why the job is no longer a candidate for processing.

'processing'(5): Either:

1. the job is using, or is attempting to use, one or more document transforms which include (1) purely software processes that are interpreting a PDL, and (2) hardware devices that are interpreting a PDL, making marks on a medium, and/or performing finishing, such as stapling OR
2. the server has made the job ready for printing, but the output device is not yet printing it, either because the job hasn't reached the output device or because the job is queued in the output device or some other spooler, awaiting the output device to print it.

When the job is in the 'processing' state, the entire job state includes the detailed status represented in the printer's "printer-state", "printer-state-reasons", and "printer-state-message" attributes.

Implementations MAY include additional values in the job's "job-state-reasons" attribute to indicate the progress of the job, such as adding the 'job-printing' value to indicate when the output device is actually making marks on paper. Most implementations won't bother with this nuance.

'processing-stopped'(6): The job has stopped while processing for any number of reasons and will return to the 'processing' state as soon as the reasons are no longer present.

The job's "job-state-reason" attribute MAY indicate why the job has stopped processing. For example, if the output device is stopped, the 'printer-stopped' value MAY be included in the job's "job-state-reasons" attribute. For example, if the output device is stopped, the 'printer-stopped' value MAY be included in the job's "job-state-reasons" attribute.

NOTE - When an output device is stopped, the device usually indicates its condition in human readable form locally at the device. A client can obtain more complete device status remotely by querying the printer's "printer-state", "printer-state-reasons" and "printer-state-message" attributes.

'canceled'(7): The job has been canceled by a Cancel-Job operation and is either (1) in the process of terminating or (2) has completed terminating. The job's "job-state-reasons" attribute SHOULD contain either the 'canceled-by-user' or 'canceled-by-operator' value.

'aborted'(8): The job has been aborted by the system, usually while the job was in the 'processing' or 'processing-stopped'

```
     state.
  'completed'(9):  The job has completed successfully or with
     warnings or errors after processing and all of the job media
     sheets have been successfully stacked in the appropriate output
```

     bin(s). The job's "job-state-reasons" attribute SHOULD contain
     one of: 'completed-successfully', 'completed-with-warnings', or
     'completed-with-errors' values.

The final value for this attribute SHALL be one of: 'completed', 'canceled', or 'aborted' before the Printer removes the job altogether. The length of time that jobs remain in the 'canceled', 'aborted', and 'completed' states depends on implementation.

The following figure shows the normal job state transitions.

```
                                                   +----> canceled
                                                  /
    +----> pending --------> processing ---------+------> completed
    |         ^                   ^               \
--->+         |                   |                +----> aborted
    |         v                   v               /
    +----> pending-held    processing-stopped ---+
```

Normally a job progresses from left to right. Other state transitions are unlikely, but are not forbidden. Not shown are the transitions to the 'canceled' state from the 'pending', 'pending-held', 'processing', and 'processing-stopped' states.

4.3.7 **job-state-reasons (1setOf** type2 keyword)

This attribute provides additional information about the job's current state, i.e., information that augments the value of the job's "job-state" attribute.

Implementation of these values is OPTIONAL, i.e., a Printer NEED NOT implement them, even if (1) the output device supports the functionality represented by the reason and (2) is available to the Printer object implementation. These values MAY be used with any job state or states for which the reason makes sense. Furthermore, when implemented, the Printer SHALL return these values when the reason applies and SHALL NOT return them when the reason no longer applies whether the value of the job's "job-state" attribute changed or not. When the job does not have any reasons for being in its current state, the Printer shall set the value of the job's "job-state-reasons" attribute to 'none'.

NOTE - While values cannot be added to the 'job-state' attribute without impacting deployed clients that take actions upon receiving

"job-state"values, it is the intent that additional "job-state-
reasons" values can be defined and registered without impacting such

deployed clients. In other words, the "job-state-reasons" attribute is intended to be extensible.

The following standard values are defined:

NOTE - For easy of understanding the order of the reasons is presented in the order in which the reason is most likely to occur:

'none': There are no reasons for the job's current state.
'job-incoming': The CreateJob operation has been accepted by the Printer, but the Printer is expecting additional SendDocument operations and/or is accessing/accepting document data.
'job-outgoing': The Printer is transmitting the job to the output device.
'job-hold-until-specified': The value of the job's "job-hold-until" attribute specifies a time period that is still in the future. The job SHALL NOT be a candidate for processing until this reason is removed and there are no other reasons to hold the job.
'resources-are-not-ready': At least one of the resources needed by the job, such as media, fonts, resource objects, etc., is not ready on any of the physical printer's for which the job is a candidate. This condition MAY be detected when the job is accepted, or subsequently while the job is pending or processing, depending on implementation.
'printer-stopped-partly': The value of the Printer's "printer-state-reasons" attribute contains the value 'stopped-partly'.
'printer-stopped': The value of the Printer's "printer-state" attribute is 'stopped'.
'job-printing': The output device is marking media. This value is useful for Printers which spend a great deal of time processing when no marking is happening and then want to show that marking is now happening.
'job-cancelled-by-user': The job was cancelled by the user using the CancelJob request, i.e., by a user whose name is the same as the value of the job's "job-originating-user" attribute.
'job-cancelled-by-operator': The job was cancelled by the operator using the CancelJob request, i.e., by a user whose name is different than the value of the job's "job-originating-user" attribute.
'job-completed-successfully': The job completed successfully.
'job-completed-with-warnings': The job completed with warnings.
'job-completed-with-errors': The job completed with errors (and possibly warnings too).

```
   'logfile-pending ':  The job's logfile is pending file transfer.
   'logfile-transferring':  The job's logfile is being transferred.
```

4.3.8 job-state-message (text)

This attributes specifies supplemental information about the Job State in human readable text.

4.3.9 output-device-assigned (name)

This attribute identifies the Output Device to which the Printer has assigned this job. If an output device implements an embedded IPP Printer, the Printer NEED NOT set this attribute. If a Print Server implements a Printer, the value MAY be empty until the Printer assigns an output device to the job.

4.3.10 time-since-pending (milliseconds)

This attribute indicates the amount of time that has passed since the Job was first put into the pending state..

4.3.11 time-since-processing (milliseconds)

This attribute indicates the amount of time that has passed since the Job first entered the processing state.

4.3.12 time-since-completed (milliseconds)

This attribute indicates the amount of time that has passed since the Job was completed.

4.3.13 number-of-intervening-jobs (integer(0:2**31 - 1))

This attribute indicates the number of jobs that are "ahead" of this job in the current scheduled order. For efficiency, it is only necessary to calculate this value when an operation is performed that requests this attribute.

Note: This attribute is necessary since an end user may request just their own jobs and they need some relative position indicator if there are other jobs interspersed in the waiting list which are not returned in the response or cannot be because of site security policy restrictions.

4.3.14 job-message-from-operator (text)

This attribute provides a message from an operator, system administrator or "intelligent" process to indicate to the end user the reasons for modification or other management action taken on a job.

4.3.15 job-k-octets-processed (integer(0:2**31 - 1))

This attribute specifies the number of octets completed in K octets, i.e., in units of 1024 octets. The value SHALL be rounded up, so that a job between 1 and 1024 octets SHALL be indicated as being 1, 1025 to 2048 **SHALL be 2, etc.**

Note: This attribute and the following two attributes ("job-impressions-completed" and "job-sheets-completed") are intended to be counters as in the Job Monitoring MIB [27].

4.3.16 job-impressions-completed (integer(0:2**31 - 1))

This job attribute specifies the number of impressions completed. This attribute is intended to be a counter as in the Job Monitoring MIB.

4.3.17 job-media-sheets-completed (integer(0:2**31 - 1))

This job attribute specifies the media-sheets completed. This attribute is intended to be a counter as in the Job Monitoring MIB.

4.4 Document Attributes

This group of attributes describes the document data for the job. For single-Document Jobs, they are supplied in the Print-Job or Print-URI requests. For multi-Document Jobs, they are supplied in each Send-Document or Send-URI request.

Attribute	Syntax	MANDATORY?
document-name	name	MANDATORY
document-format	type 2 enum	
document-uri	uri	

4.4.1 document-name (name)

This attribute contains the name of the document used by the client to initially identify the document. When a client prints by reference, i.e. includes the document-URI attribute and no document content, this attribute SHALL be absent.

4.4.2 document-format (type2 enum)

See section 4.2.16 that describes the "document-format" Job Template attribute.

4.4.3 document-uri (uri)

This attribute contains the URI of the document when the document content is not included in the Send Document operation. Document-number is the only other attribute allowed when a document-URI attribute is present in a Send Document operation.

4.5 Printer Description Attributes

These attributes form the attribute group called "printer-description". A Printer object may be realized in either a print server or output device. Note: How these attributes are set by an Administrator is outside the scope of this specification. The following table summarizes these attributes, their syntax, and whether or not they are MANDATORY. If they are not MANDATORY, they are OPTIONAL.

Attribute	Syntax	MANDATORY?
printer-uri	uri	MANDATORY
printer-uri-user	uri	
printer-name	name	MANDATORY
printer-location	text	
printer-description	text	
printer-more-info-site	uri	
printer-driver-installer	uri	
printer-make-and-model	text	
printer-more-info-manufacturer	uri	
printer-state	type1 enum	MANDATORY
printer-state-reasons	type2 keyword	
printer-state-message	text	
printer-is-accepting-jobs	boolean	MANDATORY
queued-job-count	integer	
printer-message-from-operator	text	
printer-locale	locale	MANDATORY
printer-locales-supported	1setOf locale	MANDATORY
color-supported	boolean	

4.5.1 printer-uri (uri)

This attribute contains the URI for the printer. An administrator SHALL determine a printer's URI and SHALL set this attribute to that URI. The precise format of a printer URI SHALL be implementation dependent.

4.5.2 printer-uri-user (uri)

Similar to "printer-uri", this attribute contains the URI for an HTML page with more information about this printer.

4.5.3 printer-name (name)

This attribute contains the name of the printer. It is a name that is more user friendly than the printer-URI. An administrator SHALL determine a printer's name and SHALL set this attribute to that name. This name may be the last part of the printer's URI or it may be unrelated. In non-US-English locales, a name may contain characters that are not allowed in a URI.

4.5.4 printer-location (text)

This attribute identifies the location of this printer.

4.5.5 printer-description (text)

This attribute identifies the descriptive information about the this Printer. This could include things like: "This printer can be used for printing color transparencies for HR presentations", or "Out of courtesy for others, please print only small (1-5 page) jobs at this printer", or even "This printer is going away on July 1, 1997, please find a new printer".

4.5.6 printer-more-info-site (uri)

This attribute contains a URI used to obtain more information about this specific printer. The information obtained from this URI is intended for end user consumption. Features outside the scope of IPP can be accessed from this URI. The information is intended to be specific to this printer instance and site services (e.g. job pricing, services offered, end user assistance). The printer manufacturer may initially populate this attribute.

4.5.7 printer-driver-installer (uri)

This attribute contains a URI to use to locate the driver installer for this printer.   This attribute is intended for consumption by automata. The mechanics of print driver installation is outside the scope of IPP. The printer manufacturer may initially populate this attribute.

4.5.8 printer-make-and-model (text)

This attribute identifies the make and model of the printer.

4.5.9 printer-more-info-manufacturer (uri)

This attribute contains a URI used to obtain more information about this type of printer. The information obtained from this URI is intended for end user consumption. Features outside the scope of IPP can be accessed from this URI (e.g., latest firmware, upgrades, print drivers, optional features available). The information is intended to be germane to this printer without regard to site specific modifications or services.

4.5.10 printer-state (type1 keyword)

This attribute identifies the current state of the printer. The "printer-state reasons" attribute augments the "printer-state" attribute to give more detailed information about the Printer in the given printer state.

A Printer SHALL continually keep this attribute set to the value in the table below which most accurately reflects the state of the Printer. A Printer NEED NOT implement all values if they are not applicable to a given implementation.

The following standard values are defined:

'unknown'(2): The Printer state is not known, or is indeterminate. A Printer SHALL use this state only if it cannot determine its actual state.

'idle'(3): If a Printer receives a job (whose required resources are ready) while in this state, such a job SHALL transit into the processing state immediately. If the printer-state-reasons attribute contains any reasons, they SHALL be reasons that would not prevent a job from transiting into the processing state immediately, e.g., toner-low. Note: if a Printer controls more than one output device, the above definition implies that a Printer is idle if at least one output device is idle.

'processing'(4): If a Printer receives a job (whose required resources are ready) while in this state, such a job SHALL transit into the pending state immediately. Such a job SHALL transit into the processing state only after jobs ahead of it complete. If the printer-state-reasons attribute contains any reasons, they SHALL be reasons that do not prevent the current job from printing, e.g. toner-low. Note: if a Printer controls more than one output device, the above definition implies that a

Printer is processing if at least one output device is
processing, and none is idle.

'stopped'(5): If a Printer receives a job (whose required resources are ready) while in this state, such a job SHALL transit into the pending state immediately. Such a job SHALL transit into the processing state only after some human fixes the problem that stopped the printer and after jobs ahead of it complete printing. The "printer-state-reasons" attribute SHALL contain at least one reason, e.g. media-jam, which prevents it from either processing the current job or transiting a pending job to the processing state.

Note: if a Printer controls more than one output device, the above definition implies that a Printer is stopped only if all output devices are stopped. Also, it is tempting to define stopped as when a sufficient number of output devices are stopped and leave it to an implementation to define the sufficient number. But such a rule complicates the definition of stopped and processing. For example, with this alternate definition of stopped, a job can move from idle to processing without human intervention, even though the Printer is stopped.

4.5.11 **printer-state-reasons (1setOf type2 keyword)**

This attribute supplies additional detail about the printer's state.

Each MAY have asuffix to indicate its level of severity. The three levels are: report (least severe), warning, and error (most severe).

- '-report': This suffix indicates that the reason is a "report". An implementation may choose to omit some or all reports. Some reports specify finer granularity about the printer state; others serve as a precursor to a warning. A report SHALL contain nothing that could affect the printed output.
- '-warning': This suffix indicates that the reason is a "warning". An implementation may choose to omit some or all warnings. Warnings serve as a precursor to an error. A warning SHALL contain nothing that prevents a job from completing, though in some cases the output may be of lower quality.
- '-error': This suffix indicates that the reason is an "error". . An implementation SHALL include all errors. If this attribute contains one or more errors, printer SHALL be in the stopped state.

If the implementation does not add any one of the three suffixes, all parties SHALL assume that the reason is an "error".

If a logical Printer controls more than one output device, each value
of this attribute MAY apply to one or more of the output devices. An
error on one output device that does not stop the logical Printer as a

whole MAY appear as a warning in the Printer's "printer-state-reasons attribute". The "printer-state" for such a Printer may have a value of 'stopped' even though there are with no "printer-state-reasons" values that are "errors".

The following standard values are defined:

'media-needed': A tray has run out of media.
'media-jam': The printer has a media jam.
'paused': Someone has paused the Printer. In this state, a Printer SHALL not produce printed output, but it SHALL perform other operations requested by a client. If a Printer had been printing a job when the Printer was paused, the Printer SHALL resume printing that job when the Printer is no longer paused and leave no evidence in the printed output of such a pause.
'shutdown': Someone has removed a Printer from service, and it may be powered down or physical removed. In this state, a Printer SHALL not produce printed output, and unless the Printer is realized by a print server that is still active, the Printer SHALL perform no other operations requested by a client, including returning this value. If a Printer had been printing a job when it was shutdown, the Printer need not resume printing that job when the Printer is no longer shutdown. If the Printer resumes printing such a job, it may leave evidence in the printed output of such a shutdown, e.g. the part printed before the shutdown may be printed a second time after the shutdown.
'connecting-to-device': The server has scheduled a job on the Printer and is in the process of connecting to a shared network output device (and might not be able to actually start printing the job for an arbitrarily long time depending on the usage of the output device by other servers on the network).
'timed-out': The server was able to connect to the output device (or is always connected), but was unable to get a response from the output device.
'stopping': The printer will be stopping in a while and will change its reason to printer-stopped. This reason is a non-critical, even for a Printer with a single output device. When an output-device ceases accepting jobs, the Printer will have this state while the output device completes printing.
'stopped-partly': When a Printer controls more than one output device, this reason indicates that one or more output devices are stopped. If the reason is a report, fewer than half of the output devices are stopped. If the reason is a warning, fewer than all of the output devices are stopped.

```
  'toner-low': The Printer is low on toner.
  'spool-area-full': The limit of persistent storage allocated for
     spooling has been reached.
```

4.5.12 printer-is-accepting-jobs (boolean)

This attribute determines whether the printer is currently accepting job. If the value is true, the printer is accepting jobs. If the value is false, the printer is currently rejecting any jobs submitted to it.

Note: This value is independent of the printer state and printer-state-reasons because its value does not affect the current job; rather it affects future jobs. This attribute may cause the Printer to reject jobs when the printer-state is idle or it may cause the Printer to accepts jobs when the printer-state is stopped.

4.5.13 printer-state-message (text)

This attribute specifies the additional information about the printer state in human readable text and it SHALL be set by the Printer (or the Administrator by some mechanism outside the scope of IPP).

4.5.14 queued-job-count (integer(0:2**31 - 1))

This attribute contains a count of the number of jobs that are either pending and/or processing and SHALL be set by the Printer.

4.5.15 printer-message-from-the-operator (text)

This attribute provides a message from an operator, system administrator or "intelligent" process to indicate to the end user information or status of the printer, such as why it is unavailable or when it is expected to be available.

4.5.16 printer-locale (locale)

This attribute specifies the current locale that the Printer is operating in.

4.5.17 printer-locales-supported (1setOf locale)

This attribute specifies the supported locales that the Printer operates in.

4.5.18 color-supported (boolean)

This attribute identifies whether the Printer is capable of any type of color printing at all. All document instructions having to do with

color are embedded within the document PDL (none are external IPP
attributes).

4.5.19 pdl-override (type2 keyword)

A client supplies Job Template attributes to affect the rendering, production and finishing of the documents in the job. Similar types of instructions may also be contained in the document to be printed, that is, within the Page Description Language (PDL) of the document data. If there is a conflict between the value of one of these IPP Job Template attributes, and a corresponding instruction in the document (either implicit or explicit), it is desirable that the value of the IPP attribute take precedence over the document instruction. Until companies that supply interpreters for PDLs, such as PostScript and PCL allow a way for external attributes (such as IPP attributes) to take precedence over internal job production instructions, a Printer might not be able to support the semantics that IPP attributes override (take on a higher precedence) the embedded PDL instructions. This attribute expresses how a particular Printer implementation handles these conflicts.

This attribute takes on the following values:

- 'guaranteed': This value indicates that the Printer guarantees that all IPP attribute values take precedence over embedded PDL instructions.
- 'attempted': This value indicates that the Printer attempts to make sure that IPP attribute values take precedence over embedded PDL instructions, however there is no guarantee.
- 'ignored': This value indicates that the Printer ignores all IPP Job Template attributes and it makes no attempts to ensure that IPP attribute values take precedence over embedded PDL instructions.

This is a MANDATORY attribute.

Note: Since 'attempted' does not offer any type of guarantee, a given implementation might not do a very "good" job of attempting to ensure that IPP attributes take a higher precedence over PDL instructions embedded in the document data, but it would still be a conforming implementation.

If the value of this attribute is 'guaranteed', the implementation MUST guarantee that the IPP attribute values take precedence over any related job processing instructions in the PDL Job's document data. This can be done by modifying the interpreter within the output device itself to understand IPP attributes, or by merging theses Job Template attributes directly into the document data, or in any other

implementation specific manner. In any case, the semantics of
'guaranteed' MUST be preserved.

4.5.20 Security Related Attributes

The security document [22] describes four common usage scenarios:

- no security
- message protection
- client authentication and authorization
- mutual authentication, authorization, and message protection

In order to let an end user know what to expect in terms of security, there are two attributes described below. Since by definition an end user, because of security reasons, might not be allowed to query these two attributes, therefore, it is important that if these two attributes are supported, then they are also populated in the directory entry (see [24]).

These attributes allow for minimal client/server negotiation regarding security features. If the Printer requires the feature, the client can decide whether or not to participate. If the client does not support the feature, and the Printer requires it, then the client knows before hand that such an interaction would fail.

Standard values for these two attributes include:

'supported' - means that the Printer is capable of supporting the security feature (somehow), but it is does not require the client to use it.
'required' - means that the Printer is capable of supporting the security feature (somehow) and the client is required to use it.
'none' - means that the Printer is not capable of supporting message protection at all.

Note: This is a single-valued attribute, not a multi-valued attribute, i.e., an implementation can not support 'none' and 'required' or any other combination of values.

4.5.20.1 message-protection-supported (keyword)

This attribute is used to determine whether or not a printer supports or requires message protection (whether it be through encryption or some other privacy mechanism).

4.5.20.2 authentication-authorization-supported (keyword)

This attribute is used to determine whether or not a printer supports or requires authentication and authorization.

5. Conformance

This section describes conformance issues and requirements. This document introduces model entities such as objects, operations, attributes, and attribute values. These conformance sections describe the conformance requirements which apply to these model entities.

5.1 Conditionally Mandatory

For example, a conditionally mandatory attribute means that a Printer implementation need not support the attribute if the attribute controls a feature that the output device does not implement or expose. For example, for an output device that can only print on one side, a Printer need not support the "sides" attribute. For an output device that does not support any of the finishing attribute values, a Printer need not support the "finishing" attribute.

5.2 Client Conformance Requirements

There are no conformance requirements placed on the user interfaces provided by IPP clients or their applications. For example, one application might not allow an end user to submit multiple documents per job, while another does. One application might first query a Printer object in order to supply a graphical user interface (GUI) dialogue box with supported and default values whereas a different implementation might not.

When sending a Get-Attributes or create request, an IPP client need not supply any attributes.

A client SHALL be able to accept any of the attribute syntaxes defined in Section 4.1 that may be returned to it in a response from a Printer

A query response may contain attributes and values that the client does not expect. Therefore, a client implementation MUST gracefully handle such responses and not refuse to interoperate with a conforming Printer that is returning extended registered or private attributes and/or attribute values that conform to Section 6. Clients may choose to ignore any attributes that it does not understand.

5.3 Printer Object Conformance Requirements

This section specifies the conformance requirements for conforming Printer object implementations with respect to objects, operations, and attributes.

5.3.1 Objects

Conforming Printer implementations SHALL implement all of the model objects as defined in this specification in the indicated sections:

  Section 2.1 Printer Object
  Section 2.2 Job Object
  Section 2.3 Document Object

5.3.2 Operations

Conforming Printer implementations SHALL implement all of the MANDATORY model operations, including mandatory responses, as defined in this specification in the indicated sections:

For a Printer object:

Get-Operations (section 3.1.1)	MANDATORY
Print-Job (section 3.1.2)	MANDATORY
Print-URI (section 3.1.3)	OPTIONAL
Validate-Job (section 3.1.4)	MANDATORY
Create-Job (section 3.1.5)	OPTIONAL
Get-Jobs (section 3.1.8)	MANDATORY
Get-Attributes (section 3.1.9)	MANDATORY

For a Job object:

Send-Document (section 3.1.6)	OPTIONAL
Send-URI (section 3.1.7)	OPTIONAL
Cancel-Job (section 3.1.8)	MANDATORY
Get-Attributes (section 3.1.9)	MANDATORY

5.3.3 Attributes

Conforming Printer implementations SHALL support all of the MANDATORY attributes, as defined in this specification in the indicated sections.

Conforming Printer implementations SHALL support all CONDITIONALLY

MANDATORY attributes as defined in this specification (in the
indicated sections) if in the implementation the condition us true.

If a Printer implements a "xxx-supported" attribute it MUST implement the corresponding "xxx" default value attribute and vice versa.

5.3.4 Printer extensions

A conforming Printer may support registered extensions and private extensions, as long as they meet the requirements specified in Section 6.

5.3.5 Attribute Syntaxes

A Printer SHALL be able to accept any of the attribute syntaxes defined in Section 4.1 in any operation in which a client may supply attributes or parameters. Furthermore, a Printer SHALL return attributes to the client in operation responses that conform to the syntax specified in Section 4.1.

5.4 Security Conformance Requirements

The security mechanisms being considered for IPP fall outside the scope of the application layer protocol itself. There are two mechanisms used to begin secure communications using IPP:

1. Information in the directory entry for an IPP Printer (or from additional information at a Web site hosting the IPP Printer) indicate which, if any, security protocols are used in conjunction with IPP.

2. The URI for the IPP Printer contains the security protocol information (https://..., etc.).

In either case, the security protocol (if any) is initiated first which allows for the negotiation of security features. IPP is then run as an application protocol on top of the security protocols. One cannot "bootstrap" the security features from IPP itself.

ISSUE: The above is not quite correct. Waiting for better description from the security document [22].

6. IANA Considerations (registered and private extensions)

During the development of this standard, the IPP working group

(working with IANA) will register additional keywords and enums while
the standard is in the proposed and draft states according to the

procedures described in this section. IANA will handle registration of additional enums after this standard is approved in cooperation with an IANA-appointed registration editor from the IPP working group according to the procedures described in this section.

6.1 Typed Extensions

This document identifies both keywords and enum values. Throughout this document, references to "typeN enum" or "typeN keyword" can be found (where N can be 1, 2, 3 or 4). The typeN extension has more registration controls than a typeM extension where M is greater than N. **That is, there are more controls in place to extend a type1 value** over a type 2 value. However, any typeN value MAY be registered using a process for some typeM where M is less than N, however such registration is NOT REQUIRED. For example, a type4 value MAY be registered in a type1 manner (i.e., by being included in a future version of an IPP specification). For private (unregistered) keyword extensions, implementers SHOULD use keywords with a suitable distinguishing prefix, such as "xxx-" where xxx is the (lowercase) company name registered with IANA for use in domain names. For private (unregistered) enum extension, implementers SHOULD support values in the reserved integer range (see "enum").

The definitions of these various types are as follows.

6.1.1 Type1

The IPP standard must be revised to add a new keyword or a new enum. No private keywords or enums are allowed.

This draft contains the following type1 keywords:

- <fill in>

This draft contains the following type1 enums:

- <fill in>

6.1.2 Type2

Implementers can, at any time, add new keyword or enum values by proposing them to the IPP working group for registration (or an IANA-appointed registry advisor after the IPP working group is no longer certified) where they are reviewed for approval. IANA keeps the registry.

This draft contains the following type2 keywords:

- <fill in>

This draft contains the following type2 enums:

- <fill in>

6.1.3 Type3

Implementers can, at any time, add new keyword and enum values by submitting a registration request directly to IANA, no IPP working group or IANA-appointed registry advisor review is required.

This draft contains the following type3 keywords:

- <fill in>

This draft contains the following type3 enums:

- <fill in>

6.1.4 Type4

Anyone (system administrators, system integrators, site managers, etc.) can, at any time, add new installation-defined values (keywords or new enum values) to a local system. Care SHOULD be taken by the implementers to see that keywords do not conflict with other keywords defined by the standard or as defined by the implementing product. There is no registration or approval procedure for type4 values.

This draft contains the following type4 keywords:

- <fill in>

This draft contains the following type4 enums:

- <fill in>

6.2 Registration of MIME types/sub-types for document-formats

The "document-format" attribute has MIME type/sub-type values for indicating document formats which IANA registers as "media type" names.

7. Security Considerations

There is another Internet-Draft called "Internet Printing Protocol/1.0: Security" [22]. That document is being drafted and reviewed in parallel with this document. The mapping of IPP on top of appropriate security protocols will be described in that document. IPP does not introduce any new, general purpose security mechanisms for authentication and encryption.

A Printer may choose, for security reasons, not to return all attributes that a client requests. It may even return none of the requested attributes. In such cases, the status returned is the same as if the Printer had returned all requested attributes. The client cannot tell by such a response whether the requested attribute was present or absent on the Printer.

8. References

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

[2] R Fielding, et al, _Hypertext Transfer Protocol _ HTTP/1.1_ RFC 2068, January 1997

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

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

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

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

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

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

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

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

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

[20] Internet Printing Protocol: Requirements

[21] Internet Printing Protocol/1.0: Model and Semantics (This document)

[22] Internet Printing Protocol/1.0: Security

[23] Internet Printing Protocol/1.0: Protocol Specification

[24] Internet Printing Protocol/1.0: Directory Schema

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

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

[27] T. Hastings, "Job Monitoring MIB", <draft-ietf-print-mib-monitoring-01.txt>, June 1997.

[28] F. Yergeau, "UTF-8, a transformation format of Unicode and ISO 10646, RFC 2044, October 1996.

[29] Turner, R. "Printer MIB", draft-ietf-printmib-mib-info-02.txt, July 8, 1997. This I-D is an update to RFC 1759, March 1995 [1].

[30] P. Mockapetris, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION", RFC 1035, November 1987.

9. Author's Address

Scott A. Isaacson (Editor)
Novell, Inc.
122 E 1700 S
Provo, UT 84606

Phone: 801-861-7366
Fax: 801-861-4025
EMail: scott_isaacson@novell.com

Tom Hastings
Xerox Corporation

```
      701 S. Aviation Blvd.
      El Segundo, CA   90245
```

Phone: 310-333-6413
Fax: 310-333-5514
EMail: hastings@cp10.es.xerox.com

Robert Herriot
Sun Microsystems Inc.
2550 Garcia Ave., MPK-17
Mountain View, CA 94043

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

Roger deBry
HUC/003G
IBM Corporation
P.O. Box 1900
Boulder, CO 80301-9191

Phone: (303) 924-4080
Fax: (303) 924-9889
Email: debry@vnet.ibm.com

Patrick Powell
San Diego State University
9475 Chesapeake Dr., Suite D
San Diego, CA 95123

Phone: (619) 874-6543
Fax: (619) 279-8424
Email: papowell@sdsu.edu

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

Other Participants:

Chuck Adams - Tektronix
Jeff Barnett - IBM
Ron Bergman - Data Products
Keith Carter, IBM Corporation
Jeff Copeland - QMS
Andy Davidson - Tektronix

Mabry Dozier - QMS
Lee Farrell - Canon Information Systems
Steve Gebert - IBM

David Kellerman - Northlake Software
Rick Landau - Digital
Harry Lewis - IBM
Pete Loya - HP
Ray Lutz - Cognisys
Mike MacKay, Novell, Inc.
Carl-Uno Manros, Xerox, Corp.
Jay Martin - Underscore
Stan McConnell - Xerox
Pat Nogay - IBM
Bob Pentecost - HP
Rob Rhoads - Intel
David Roach - Unisys
Hiroyuki Sato - Canon
Bob Setterbo - Adobe
Devon Taylor, Novell, Inc.
Mike Timperman - Lexmark
Randy Turner - Sharp
Atsushi Yuki - Kyocera
Lloyd Young - Lexmark
Bill Wagner - DPI
Jim Walker - DAZEL
Chris Wellens - Interworking Labs
Rob Whittle - Novell
Don Wright - Lexmark
Peter Zehler, Xerox, Corp.

10. APPENDIX A: Terminology

This specification uses the terminology defined in this section.

10.1 Conformance Terminology

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

10.1.1 MUST

This word, or the terms "REQUIRED", "SHALL" or "MANDATORY", means that the definition is an absolute requirement of the specification.

10.1.2 MUST NOT

This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.

10.1.3 SHOULD

This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

10.1.4 SHOULD NOT

This phrase, or the phrase "NOT RECOMMENDED" means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

10.1.5 MAY

This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the

same vein an implementation which does include a particular option
MUST be prepared to interoperate with another implementation which

does not include the option (except, of course, for the feature the option provides.)

10.1.6 CONDITIONALLY MANDATORY

This term means that an item MUST be implemented in a conforming implementation if the specified condition is true. Conversely, a conforming implementation NEED NOT implement the item if the specified condition is false.

10.1.7 NEED NOT

The verb "NEED NOT" indicates an action that the subject of the sentence does not have to implement in order to claim conformance to the standard. The verb "NEED NOT" is used instead of "MAY NOT" since "MAY NOT" sounds like a prohibition.

10.2 Model Terminology

10.2.1 Keyword

Keywords are used within this document as identifiers of semantic entities within the abstract model. Attribute names, some attribute values, attribute syntaxes, and attribute group names are represented as keywords. In this document, a keyword is a sequence of characters (length of 1 to 255) which consists of the following ASCII characters: lower-case letters ("a" - "z"), digits ("0" - "9"), hyphen ("-"), and underscore ("_"). A keyword starts with a lower-case letter. In the actual protocol, these keywords will be represented using an appropriate protocol encoding.

10.2.2 Parameters

A parameter is an item of information supplied in an operation consisting of a parameter name and a parameter value(s). Each parameter has a specific syntax. Clients supply input parameters in operation requests and servers return output parameters in operation responses. Most parameters have corresponding object attributes; some do not. All parameters are defined in section 3.

10.2.2.1 Parameter Name

Each parameter is uniquely identified in this document by its parameter name which is a keyword. The keyword parameter name is given in the section header describing that parameter. In running text in this document, parameter names are indicated inside double quotation marks (").

10.2.2.2 Parameter Value

Each parameter has one or more values. Parameter values are represented in the syntax type specified for that parameter. In running text in this document, parameter values are indicated inside single quotation marks ('), whether their parameter syntax is keyword, integer, text, etc.

10.2.2.3 Parameter Syntax

Each parameter is defined using an explicit syntax. In this document, each syntax type is defined as a keyword with specific meaning. The protocol specification document [23] indicates the actual on-the-wire encoding for each parameter syntax. Parameter syntaxes are the same as attribute syntaxes and they are defined in section 4.1.

10.2.3 Attributes

An attribute is an item of information that is associated with an instance of an IPP object. An attribute consists of an attribute name and an attribute value(s). Each attribute has a specific syntax.. All attributes are defined in section 4.

An interesting set of attributes is called Job Template Attributes (these attributes are described in detail in section 4.2.) The client optionally supplies Job Template attributes as input parameters in a create request (operation requests that create Job objects). The Printer object has associated attributes which define supported and default values for the Printer. Thee following rules apply:

- When a Job Template attribute is supplied as an input parameter in a create request, the attribute and its value describe the desired job processing behavior.

- The Printer object's supported attribute describes what behaviors are possible.

- The Printer object's default value attribute describes what will be done when no other job processing information is supplied by the client.

10.2.3.1 Attribute Name

Each attribute is uniquely identified in this document by its attribute name which is a keyword. The keyword attribute name is

given in the section header describing that attribute. In running
text in this document, attribute names are indicated inside double
quotation marks (").

10.2.3.2 Attribute Group Name

Related attributes are grouped into named groups. The name of the group is a keyword. The group name may be used in an input parameter in place of naming all the attributes in the group explicitly. Attribute groups are defined in section 4.

10.2.3.3 Attribute Value

Each attribute has one or more values. Attribute values are represented in the syntax type specified for that attribute. In running text in this document, attribute values are indicated inside single quotation marks ('), whether their attribute syntax is keyword, integer, text, etc.

10.2.3.4 Attribute Syntax

Each attribute is defined using an explicit attribute syntax. In this document, each attribute syntax is defined as a keyword with specific meaning. The protocol specification document [23] indicates the actual on-the-wire encoding for each attribute syntax. Attribute syntaxes are defined in section 4.1.

10.2.4 Supports

By definition, an implementation (an instance of an IPP object) supports an attribute (or a particular value of an attribute) only if that implementation responds with the corresponding attribute and value in a response to a query for that attribute. A given implementation may exhibit a behavior that corresponds to the value of some attribute, but if the implementation, when queried for that attribute, doesn't respond with the supported attribute populated with that specific value, then as far as IPP is concerned, that implementation does not support that feature.

A conforming implementation SHALL support all MANDATORY attributes and all CONDITIONALLY MANDATORY attributes where the implementation satisfies the condition associated with the CONDITIONALLY MANDATORY attribute. Conformance to IPP does not mandate that all implementations support all possible values representing all possible job processing behaviors and features. For example, if a given instance of a Printer supports only certain document formats, then that Printer responds with the "document-format-supported" attribute populated with a set of values, possibly only one, taken from the entire set of possible values defined in this model document. This

set of values represent the Printer's set of supported document
formats. Another example is the "finishings-supported" attribute. If
a Printer is not physically capable of stapling (there is no stapler

in the output device itself), the "finishings-supported" attribute MUST NOT be populated with the value of 'staple'.

In order to ease the implementation burden, if a certain implementation supports only one well-known value of some "xxx-supported" attribute, it is NOT REQUIRED that that implementation support that attribute. For example, if a Printer object represents a physical device that can not print on two sides of the media, the only possible value for the "sides-supported" attribute could be 'one-sided'. In this case, it is NOT REQUIRED that the implementation support the "sides-supported" attribute since a client could infer that the implementation supports only single-sided printing from the absence of the "sides-supported" attribute. If for some reason, an implementation only supported "two-sided-long-edge" then that implementation MUST support the "sides-supported" attribute to be a conforming implementation. The "well-known value" for CONDITIONALLY MANDATORY attribute is specified in the section describing that attribute.

Note: The supported attributes are set (populated) by some administrative process or automatic sensing mechanism that is outside the scope of IPP.

11. **APPENDIX B:** Status Codes

This section defines status code keywords that are used to provide semantic information on the results of an operation request. Each operation response MUST include a status code. For error type status codes, the response MAY also contain a message that provides a short textual description of the status. The status code is intended for use by automata, and the message is intended for the human end user. Since the message is an OPTIONAL component of the operation response, an IPP application (i.e. a browser, GUI, print driver or gateway) is NOT REQUIRED to examine or display the message.

The prefix of the status keyword defines the class of response as follows:

```
  "informational" - Request received, continuing process
  "successful" - The action was successfully received, understood,
     and accepted
  "redirection" - Further action must be taken in order to complete
     the request
  "client-error" - The request contains bad syntax or cannot be
```

```
      fulfilled
   "server-error" - The server failed to fulfill an apparently valid
      request
```

IPP status codes are extensible. IPP applications are NOT REQUIRED to understand the meaning of all registered status codes, though such understanding is obviously desirable. However, applications SHALL understand the class of any status code, as indicated by the prefix, and treat any unrecognized response as being equivalent to the first status code of that class, with the exception that an unrecognized response shall not be cached. For example, if an unrecognized status code of "client-error-foo-bar" is received by the client, it can safely assume that there was something wrong with its request and treat the response as if it had received a "client-error-bad-request" status code. In such cases, IPP applications could present the OPTIONAL message (if present) to the end user since the message is likely to contain human readable information which will help to explain the unusual status.

11.1 Status Codes (type2 keyword)

Each status code is described below. Section 11.2 contains a table that indicates which status codes apply to which operations.

11.1.1 Informational

This class of status code indicates a provisional response and is to be used for informational purposes only.

There are no status codes defined in IPP 1.0 for this class of status code.

11.1.2 Successful Status Codes

This class of status code indicates that the client's request was successfully received, understood, and accepted.

11.1.2.1 successful-ok

The request has succeeded.

11.1.3 Redirection Status Codes

This class of status code indicates that further action needs to be taken to fulfill the request.

There are no status codes defined in IPP 1.0 for this class of status code.

11.1.4 Client Error Status Codes

This class of status code is intended for cases in which the client seems to have erred. The server SHOULD return a message containing an explanation of the error situation and whether it is a temporary or permanent condition.

11.1.4.1 client-error-bad-request

The request could not be understood by the server due to malformed syntax. The IPP application SHOULD NOT repeat the request without modifications.

11.1.4.2 client-error-forbidden

The server understood the request, but is refusing to fulfill it. Additional authentication information or authorization credentials will not help and the request SHOULD NOT be repeated. This status code is commonly used when the server does not wish to reveal exactly why the request has been refused or when no other response is applicable.

11.1.4.3 client-error-not-authenticated

The request requires user authentication. The IPP client may repeat the request with suitable authentication information. If the request already included authentication information, then this status code indicates that authorization has been refused for those credentials. If this response contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the response message may contain relevant diagnostic information. This status codes reveals more information than "client-error-forbidden".

11.1.4.4 client-error-not-authorized

The requester is not authorized to perform the request. Additional authentication information or authorization credentials will not help and the request SHOULD be repeated. This status code is used when the server wishes to reveal that the authentication information is understandable, however, the requester is explicitly not authorized to perform the request. This status codes reveals more information than "client-error-forbidden".

11.1.4.5 client-error-not-possible

This status code is used when the request is for something that can happen. For example, there might be a request to cancel a job that has already been aborted by the system. The IPP client SHOULD NOT repeat the request.

11.1.4.6 client-error-timeout

The client did not produce a request within the time that the server was prepared to wait. For example, a client issued a Create-Job operation and then, after a long period of time, issued a Send-Document operation and this error status code was returned in response to the Send-Document request. The server might have been forced to clean up resources that had been held for the waiting additional Documents. The server was forced to close the Job since the client took too long. The client SHOULD NOT repeat the request without modifications.

11.1.4.7 client-error-not-found

The server has not found anything matching the request URI. No indication is given of whether the condition is temporary or permanent. For example, a client with an old reference to a Job (a URI) tries to cancel the Job, however in the mean time the Job might have been completed and all record of it at the Printer has been deleted. This status code, 'client-error-not-found' is returned indicating that the referenced Job can not be found. This error status code is also used when a client supplies a URI as a reference to the document data in either a Print-URI or Send-URI operation however the document can not be found.

ISSUE: Shall a printer be required to validate the URI at submit time? If the Printer tries to resolve the URI at job processing time, how is this error returned? In a new "job-state-reasons" value? In a "job-state-message" value?

In practice, an IPP application should avoid a not found situation by first querying and presenting a list of valid Printer URIs and Job URIs to the end-user.

11.1.4.8 client-error-gone

The requested object is no longer available at the server and no forwarding address is known. This condition should be considered

permanent. Clients with link editing capabilities should delete
references to the request URI after user approval. If the server does
not know or has no facility to determine, whether or not the condition

is permanent, the status code "client-error-not-found" should be used instead.

This response is primarily intended to assist the task of web maintenance by notifying the recipient that the resource is intentionally unavailable and that the server owners desire that remote links to that resource be removed. It is not necessary to mark all permanently unavailable resources as "gone" or to keep the mark for any length of time -- that is left to the discretion of the server owner.

11.1.4.9 client-error-request-entity-too-large

The server is refusing to process a request because the request entity is larger than the server is willing or able to process. An IPP Printer returns this status code when it limits the size of print jobs and it receives a print job that exceeds that limit or when the operation parameters are so many that their encoding causes the request entity to exceed server capacity.

11.1.4.10 client-error-request-URI-too-long

The server is refusing to service the request because the request URI is longer than the server is willing to interpret. This rare condition is only likely to occur when a client has improperly submitted a request with long query information (e.g. an IPP application allows an end-user to enter an invalid URI), when the client has descended into a URI "black hole" of redirection (e.g., a redirected URI prefix that points to a suffix of itself), or when the server is under attack by a client attempting to exploit security holes present in some servers using fixed-length buffers for reading or manipulating the Request-URI.

11.1.4.11 client-error-unsupported-document-format

The server is refusing to service the request because the print data is in a format, as specified in the "document-format" input parameter, that is not supported by the IPP Printer.

11.1.4.12 client-error-attribute- not-supported

For a Create-Job, Print-Job or Validate operation, if the IPP Printer does not support at least one attribute or one attribute value supplied in the request, the Printer shall return this status. For example, if the request requires A4 paper and that paper size is not supported by the Printer, the Printer shall return this status.

For a Get-Jobs operation, if the IPP Printer does not support one of the requested attributes, , the Printer shall return this status.

In practice, an IPP application should avoid this situation by querying an IPP Printer for its valid attributes and values before performing an operation on the Printer.

11.1.5 Server Error Status Codes

This class of status codes indicates cases in which the server is aware that it has erred or is incapable of performing the request. The server SHOULD include a message containing an explanation of the error situation, and whether it is a temporary or permanent condition.

11.1.5.1 server-error-internal- error

The server encountered an unexpected condition that prevented it from fulfilling the request. This error status code differs from "server-error-temporary-error" in that it implies a more permanent type of internal error. It also differs from "server-error-device-error" in that it implies an unexpected condition (unlike a paper-jam or out-of-toner problem which is undesirable but expected). This error status code indicates that probably some knowledgeable human intervention is required.

11.1.5.2 server-error-operation-not-supported

The server does not support the functionality required to fulfill the request. This is the appropriate response when the server does not recognize an operation or is not capable of supporting it.

11.1.5.3 server-error-service-unavailable

The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. The implication is that this is a temporary condition which will be alleviated after some delay. If known, the length of the delay may be indicated in the message. If no delay is given, the IPP application should handle the response as it would for a "server-error-temporary-internal-error" response. If the condition is more permanent, the error status codes "client-error-gone" or "client-error-not-found" could be used.

11.1.5.4 server-error- version-not-supported

The server does not support, or refuses to support, the IPP protocol

version that was used in the request message. The server is
indicating that it is unable or unwilling to complete the request
using the same version as supplied in the request other than with this

error message. The response should contain a Message describing why that version is not supported and what other versions are supported by that server.

A conforming IPP client shall specify the valid version (IPP 1.0)on each request. A conforming IPP server (IPP 1.0) SHALL NOT return this status code to a conforming IPP 1.0 client. An IPP server shall return this status code to a non-conforming IPP client.

11.1.5.5 server-error-device-error

A printer error, such as a paper jam, occurs while the IPP Printer processes a Print or Send operation. The response contains the true Job Status (the values of the "job-state" and "job-state-reasons" attributes). Additional information can be returned in the optional "job-state-message" attribute value or in the OPTIONAL status code message that describes the error in more detail. This error status code MAY be returned even though the operation was successful (the Job was submitted and is now in the 'pending' state waiting to be processed).

11.1.5.6 server-error-temporary-error

A temporary error such as a buffer full write error, a memory overflow (i.e. the document data exceeds the memory of the Printer), or a disk full condition, occurs while the IPP Printer processes an operation. The client MAY try the unmodified request again at some later point in time with an expectation that the temporary internal error condition may have been cleared.

11.2 Status Keywords for IPP Operations

PJ = Print-Job, PU = Print-URI, CJ = Create-Job, SD = Send-Document
SU = Send-URI, V = Validate, GA = Get-Attributes, GJ = Get-Jobs
GO = Get-Operations, C = Cancel-Job

	IPP Operations									
IPP Status Keyword	PJ	PU	CJ	SD	SU	V	GA	GJ	GO	C
successful-OK	x	x	x	x	x	x	x	x	x	x
client-error-bad-request	x	x	x	x	x	x	x	x	x	x
client-error-not-authenticated	x	x	x	x	x	x	x	x	x	x
client-error-not-authorized	x	x	x	x	x	x	x	x	x	x
client-error-forbidden	x	x	x	x	x	x	x	x	x	x
client-error-not-possible	x	x	x	x	x	x	x	x	x	x
client-error-not-found	x	x	x	x	x	x	x	x	x	x
client-error-timeout	x	x	x	x	x	x	x	x	x	x
client-error-gone	x	x	x	x	x	x	x	x	x	x
client-error-request-entity-too-large	x	X	X	X	X	X	X	X	X	X
client-error-request-URI-too-long	x	x	x	x	x	x	x	x	x	x
client-error-unsupported-document-format	x	x		x	x					
client-error-attribute-value-not-supported	x	x	x			x				
server-error-internal-error	x	x	x	x	x	x	x	x	x	x
server-error-service-unavailable	x	x	x	x	x	x	x	x	x	x
server-error-timeout	x	x	x	x	x	x	x	x	x	x
server-error-HTTP-version-not-supported	x	x	x	x	x	x	x	x	x	x
server-error-IPP-version-not-supported	x	x	x	x	x	x	x	x	x	x
server-error-device-error	x	x	x	x	x					
server-error-temporary-error	x	x	x	x	x					

12. APPENDIX C: "document-format" enum values

The Printer Working Group has registered a set of type2 enum values with IANA as part of the IETF Printer MIB [1] project. The standard value assigned by the PWG starts with the four letters: "lang", in order to follow SNMP ASN.1 rules that all enum symbols SHALL start with a lower case letter. The integer enum value is used as the value the "document-format" attribute.

This APPENDIX lists the document formats that are currently registered with IANA. As with all type 2 and type 3 enums, when additional enum values are registered with IANA, they may be used in IPP withhout updating this appendix.

ISSUE: PDF does not have an enum value, though it has a MIME type. Adobe needs to register PDF as one of the standard values.

The standard values registered at the current time are:

'other': 1 -
'langPCL': 3 - PCL. Starting with PCL version 5, HP-GL/2 is included as part of the PCL language. PCL and HP-GL/2 are registered trademarks of Hewlett-Packard Company.
'langHPGL': 4 - Hewlett-Packard Graphics Language. HP-GL is a registered trademark of Hewlett-Packard Company.
'langPJL': 5 - Peripheral Job Language. Appears in the data stream between data intended for a page description language. Hewlett-Packard Co.
'langPS': 6 - PostScript Language (tm) Postscript - a trademark of Adobe Systems Incorporated which may be registered in certain jurisdictions
'langIPDS': 7 - Intelligent Printer Data Stream Bi-directional print data stream for documents consisting of data objects (text, image, graphics, bar codes), resources (fonts, overlays) and page, form and finishing instructions. Facilitates system level device control, document tracking and error recovery throughout the print process. Pennant Systems, IBM
'langPPDS': 8 - IBM Personal Printer Data Stream. Originally called IBM ASCII, the name was changed to PPDS when the Laser Printer was introduced in 1989. Lexmark International, Inc.
'langEscapeP': 9 - Epson Corp.
'langEpson': 10 -
'langDDIF': 11 - Digital Document Interchange Format Digital Equipment Corp., Maynard MA
'langInterpress': 12 - Xerox Corp.
'langISO6429': 13 - ISO 6429. Control functions for Coded Character Sets (has ASCII control characters, plus additional controls for character imaging devices.) ISO Standard, Geneva, Switzerland
'langLineData': 14 - line-data: Lines of data as separate ASCII or EBCDIC records and containing no control functions (no CR, LF, HT, FF, etc.). For use with traditional line printers. May use CR and/or LF to delimit lines, instead of records. See ISO 10175 Document Printing Application (DPA) ISO standard, Geneva, Switzerland
'langMODCA': 15 - Mixed Object Document Content Architecture Definitions that allow the composition, interchange, and presentation of final form documents as a collection of data objects (text, image, graphics, bar codes), resources (fonts, overlays) and page, form and finishing instructions. Pennant Systems, IBM

'langREGIS': 16 - Remote Graphics Instruction Set, Digital
   Equipment Corp., Maynard MA
'SCS': 17 - SNA Character String Bi-directional print data stream
   for SNA LU-1 mode of communications IBM

'langSPDL': 18 - ISO 10180 Standard Page Description Language ISO Standard
'langTEK4014': 19 - Tektronix Corp.
'langPDS': 20 -
'langIGP': 21 - Printronix Corp.
'langCodeV': 22 - Magnum Code-V, Image and printer control language used to control impact/dot- matrix printers. QMS, Inc., Mobile AL
'langDSCDSE': 23 - DSC-DSE: Data Stream Compatible and Emulation Bi-directional print data stream for non-SNA (DSC) and SNA LU-3 3270 controller (DSE) communications IBM
'langWPS': 24 - Windows Printing System, Resource based command/data stream used by Microsoft At Work Peripherals. Developed by the Microsoft Corporation.
'langLN03': 25 - Early DEC-PPL3, Digital Equipment Corp.
'langCCITT': 26 -
'langQUIC': 27 - QUIC (Quality Information Code), Page Description Language for laser printers. Included graphics, printer control capability and emulation of other well- known printer . QMS, Inc.
'langCPAP': 28 - Common Printer Access Protocol Digital Equipment Corp
'langDecPPL': 29 - Digital ANSI-Compliant Printing Protocol (DEC-PPL) Digital Equipment Corp
'langSimpleText': 30 - simple-text: character coded data, including NUL, CR , LF, HT, and FF control characters. See ISO 10175 Document Printing Application (DPA) ISO standard, Geneva, Switzerlan
'langNPAP': 31 - Network Printer Alliance Protocol (NPAP). This protocol has been superseded by the IEEE 1284.1 TIPSI standard. (ref. LangTIPSI(49)).
'langDOC': 32 - Document Option Commands, Appears in the data stream between data intended for a page description . QMS, Inc
'langimPress': 33 - imPRESS, Page description language originally developed for the ImageServer line of systems. A binary language providing representations for text, simple graphics (rules, lines, conic sections), and some large forms (simple bit-map and CCITT group 3/4 encoded).The language was intended to be sent over an 8-bit channel and supported early document preparation languages (e.g. TeX and TROFF). QMS, Inc.
'langPinwriter': 34 - 24 wire dot matrix printer for USA, Europe, and Asia except Japan. More widely used in Germany, and some Asian countries than in US. NEC
'langNPDL': 35 - Page printer for Japanese market. NEC

```
  'langNEC201PL': 36 - Serial printer language used in the Japanese
     market.  NEC
  'langAutomatic': 37 - Automatic PDL sensing.  Automatic sensing of
     the interpreter language family by the printer examining the
```

document content. Which actual interpreter language families are sensed depends on the printer implementation.

'langPages': 38 - Page printer Advanced Graphic Escape Set IBM Japan
'langLIPS': 39 - LBP Image Processing System
'langTIFF': 40 - Tagged Image File Format (Aldus)
'langDiagnostic': 41 - A hex dump of the input to the interpreter
'langPSPrinter': 42 - The PostScript Language used for control (with any PDLs) Adobe Systems Incorporated
'langCaPSL': 43 - Canon Print Systems Language
'langEXCL': 44 - Extended Command Language Talaris Systems Inc
'langLCDS': 45 - Line Conditioned Data Stream Xerox Corporatio
'langXES': 46 - Xerox Escape Sequences Xerox Corporation
'langPCLXL': 47 - Printer Control Language. Extended language features for printing, and printer control. Technical reference manual # TBD. Hewlett-Packard Co.
'langART': 48 - Advanced Rendering Tools (ART). Page Description language originally developed for the Laser Press printers. Tehnical reference manual: "ART IV Reference Manual", No F33M. Fuji Xerox Co., Ltd.
'langTIPSI': 49 - Transport Independent Printer System Interface (ref. IEEE Std. 1284.1)
'langPrescribe': 50 - Page description and printer control language. It can be described with ordinary ASCII characters. Technical reference manual: "PRESCRIBE II Programming Manual"
'langLinePrinter': 51 - A simple-text character stream which supports the control codes LF, VT, FF and CR plus Centronics or Dataproducts Vertical Format Unit (VFU). language is commonly used on many older model line and matrix printers.
'langIDP': 52 - Imaging Device Protocol Apple Computer.
'langXJCL': 53 - Xerox Corp.

One special value is 'langAutomatic (37)'. However a client SHALL NOT supply the value 'langAutomatic' in a create request. If the "document-format" is unknown for a certain document, the client SHALL NOT supply the attribute in the create request or the Send-Document Request.

13. APPENDIX D: "media" keyword values

Standard keyword values are taken from several sources.

Standard values are defined (taken from ISO DPA[5] and the Printer

MIB[1]):

  'default': The default medium for the output device

'iso-a4-white': Specifies the ISO A4 white medium
'iso-a4-colored': Specifies the ISO A4 coloured medium
'iso-a4-transparent' Specifies the ISO A4 transparent medium
'iso-a3-white': Specifies the ISO A3 white medium
'iso-a3-colored': Specifies the ISO A3 coloured medium
'iso-a5-white': Specifies the ISO A5 white medium
'iso-a5-colored': Specifies the ISO A5 coloured medium
'iso-b4-white': Specifies the ISO B4 white medium
'iso-b4-colored': Specifies the ISO B4 coloured medium
'iso-b5-white': Specifies the ISO B5 white medium
'iso-b5-colored': Specifies the ISO B5 coloured medium
'jis-b4-white': Specifies the JIS B4 white medium
'jis-b4-colored': Specifies the JIS B4 coloured medium
'jis-b5-white': Specifies the JIS B5 white medium
'jis-b5-colored': Specifies the JIS B5 coloured medium

The following standard values are defined for North American media:

'na-letter-white': Specifies the North American letter white medium
'na-letter-colored': Specifies the North American letter coloured medium
'na-letter-transparent': Specifies the North American letter transparent medium
'na-legal-white': Specifies the North American legal white medium
'na-legal-colored': Specifies the North American legal coloured medium

The following standard values are defined for envelopes:

'iso-b4-envelope': Specifies the ISO B4 envelope medium
'iso-b5-envelope': Specifies the ISO B5 envelope medium
'iso-c3-envelope': Specifies the ISO C3 envelope medium
'iso-c4-envelope': Specifies the ISO C4 envelope medium
'iso-c5-envelope': Specifies the ISO C5 envelope medium
'iso-c6-envelope': Specifies the ISO C6 envelope medium
'iso-designated-long-envelope': Specifies the ISO Designated Long envelope medium
'na-10x13-envelope': Specifies the North American 10x13 envelope medium
'na-9x12-envelope': Specifies the North American 9x12 envelope medium
'monarch-envelope': Specifies the Monarch envelope
'na-number-10-envelope': Specifies the North American number 10

```
     business envelope medium
   'na-7x9-envelope': Specifies the North American 7x9 inch envelope
   'na-9x11-envelope': Specifies the North American 9x11 inch envelope
```

'na-10x14-envelope': Specifies the North American 10x14 inch envelope
'na-number-9-envelope': Specifies the North American number 9 business envelope
'na-6x9-envelope': Specifies the North American 6x9 inch envelope
'na-10x15-envelope': Specifies the North American 10x15 inch envelope

The following standard values are defined for the less commonly used media (white-only):

'executive-white': Specifies the white executive medium
'folio-white': Specifies the folio white medium
'invoice-white': Specifies the white invoice medium
'ledger-white': Specifies the white ledger medium
'quarto-white': Specified the white quarto medium
'iso-a0-white': Specifies the ISO A0 white medium
'iso-a1-white': Specifies the ISO A1 white medium
'iso-a2-white': Specifies the ISO A2 white medium
'iso-a6-white': Specifies the ISO A6 white medium
'iso-a7-white': Specifies the ISO A7 white medium
'iso-a8-white': Specifies the ISO A8 white medium
'iso-a9-white': Specifies the ISO A9 white medium
'iso-10-white': Specifies the ISO A10 white medium
'iso-b0-white': Specifies the ISO B0 white medium
'iso-b1-white': Specifies the ISO B1 white medium
'iso-b2-white': Specifies the ISO B2 white medium
'iso-b3-white': Specifies the ISO B3 white medium
'iso-b6-white': Specifies the ISO B6 white medium
'iso-b7-white': Specifies the ISO B7 white medium
'iso-b8-white': Specifies the ISO B8 white medium
'iso-b9-white': Specifies the ISO B9 white medium
'iso-b10-white': Specifies the ISO B10 white medium
'jis-b0-white': Specifies the JIS B0 white medium
'jis-b1-white': Specifies the JIS B1 white medium
'jis-b2-white': Specifies the JIS B2 white medium
'jis-b3-white': Specifies the JIS B3 white medium
'jis-b6-white': Specifies the JIS B6 white medium
'jis-b7-white': Specifies the JIS B7 white medium
'jis-b8-white': Specifies the JIS B8 white medium
'jis-b9-white': Specifies the JIS B9 white medium
'jis-b10-white': Specifies the JIS B10 white medium

The following standard values are defined for engineering media:

  'a': Specifies the engineering A size medium

- 'b': Specifies the engineering B size medium
- 'c': Specifies the engineering C size medium
- 'd': Specifies the engineering D size medium
- 'e': Specifies the engineering E size medium

The following standard values are defined for input-trays (from ISO DPA and the Printer MIB):

- 'top': The top input tray in the printer.
- 'middle': The middle input tray in the printer.
- 'bottom': The bottom input tray in the printer.
- 'envelope': The envelope input tray in the printer.
- 'manual': The manual feed input tray in the printer.
- 'large-capacity': The large capacity input tray in the printer.
- 'main': The main input tray
- 'side': The side input tray

The following standard values are defined for media sizes (from ISO DPA):

- 'iso-a0': Specifies the ISO A0 size: 841 mm by 1189 mm as defined in ISO 216
- 'iso-a1': Specifies the ISO A1 size: 594 mm by 841 mm as defined in ISO 216
- 'iso-a2': Specifies the ISO A2 size: 420 mm by 594 mm as defined in ISO 216
- 'iso-a3': Specifies the ISO A3 size: 297 mm by 420 mm as defined in ISO 216
- 'iso-a4': Specifies the ISO A4 size: 210 mm by 297 mm as defined in ISO 216
- 'iso-a5': Specifies the ISO A5 size: 148 mm by 210 mm as defined in ISO 216
- 'iso-a6': Specifies the ISO A6 size: 105 mm by 148 mm as defined in ISO 216
- 'iso-a7': Specifies the ISO A7 size: 74 mm by 105 mm as defined in ISO 216
- 'iso-a8': Specifies the ISO A8 size: 52 mm by 74 mm as defined in ISO 216
- 'iso-a9': Specifies the ISO A9 size: 37 mm by 52 mm as defined in ISO 216
- 'iso-a10': Specifies the ISO A10 size: 26 mm by 37 mm as defined in ISO 216
- 'iso-b0': Specifies the ISO B0 size: 1000 mm by 1414 mm as defined

in ISO 216
'iso-b1': Specifies the ISO B1 size: 707 mm by 1000 mm as defined
in ISO 216

'iso-b2': Specifies the ISO B2 size: 500 mm by 707 mm as defined in ISO 216
'iso-b3': Specifies the ISO B3 size: 353 mm by 500 mm as defined in ISO 216
'iso-b4': Specifies the ISO B4 size: 250 mm by 353 mm as defined in ISO 216
'iso-b5': Specifies the ISO B5 size: 176 mm by 250 mm as defined in ISO 216
'iso-b6': Specifies the ISO B6 size: 125 mm by 176 mm as defined in ISO 216
'iso-b7': Specifies the ISO B7 size: 88 mm by 125 mm as defined in ISO 216
'iso-b8': Specifies the ISO B8 size: 62 mm by 88 mm as defined in ISO 216
'iso-b9': Specifies the ISO B9 size: 44 mm by 62 mm as defined in ISO 216
'iso-b10': Specifies the ISO B10 size: 31 mm by 44 mm as defined in ISO 216
'na-letter': Specifies the North American letter size: 8.5 inches by 11 inches
'na-legal': Specifies the North American legal size: 8.5 inches by 14 inches
'executive': Specifies the executive size (7.25 X 10.5 in)
'folio': Specifies the folio size (8.5 X 13 in)
'invoice': Specifies the invoice size (5.5 X 8.5 in)
'ledger': Specifies the ledger size (11 X 17 in)
'quarto': Specifies the quarto size (8.5 X 10.83 in)
'iso-c3': Specifies the ISO C3 size: 324 mm by 458 mm as defined in ISO 269
'iso-c4': Specifies the ISO C4 size: 229 mm by 324 mm as defined in ISO 269
'iso-c5': Specifies the ISO C5 size: 162 mm by 229 mm as defined in ISO 269
'iso-c6': Specifies the ISO C6 size: 114 mm by 162 mm as defined in ISO 269
'iso-designated-long': Specifies the ISO Designated Long size: 110 mm by 220 mm as defined in ISO 269
'na-10x13-envelope': Specifies the North American 10x13 size: 10 inches by 13 inches
'na-9x12-envelope': Specifies the North American 9x12 size: 9 inches by 12 inches
'na-number-10-envelope': Specifies the North American number 10 business envelope size: 4.125 inches by 9.5 inches
'na-7x9-envelope': Specifies the North American 7x9 inch envelope

```
     size
  'na-9x11-envelope': Specifies the North American 9x11 inch envelope
     size
```

'na-10x14-envelope': Specifies the North American 10x14 inch envelope size
'na-number-9-envelope': Specifies the North American number 9 business envelope size
'na-6x9-envelope': Specifies the North American 6x9 envelope size
'na-10x15-envelope': Specifies the North American 10x15 envelope size
'monarch-envelope': Specifies the Monarch envelope size (3.87 x 7.5 in)
'jis-b0': Specifies the JIS B0 size: 1030mm x 1456mm
'jis-b1': Specifies the JIS B1 size: 728mm x 1030mm
'jis-b2': Specifies the JIS B2 size: 515mm x 728mm
'jis-b3': Specifies the JIS B3 size: 364mm x 515mm
'jis-b4': Specifies the JIS B4 size: 257mm x 364mm
'jis-b5': Specifies the JIS B5 size: 182mm x 257mm
'jis-b6': Specifies the JIS B6 size: 128mm x 182mm
'jis-b7': Specifies the JIS B7 size: 91mm x 128mm
'jis-b8': Specifies the JIS B8 size: 64mm x 91mm
'jis-b9': Specifies the JIS B9 size: 45mm x 64mm
'jis-b10': Specifies the JIS B10 size: 32mm x 45mm

Expires January 14, 1998