INTERNET-DRAFT
draft-ietf-ipp-model-05.txt

R. deBry
IBM Corporation
T. Hastings
Xerox Corporation
R. Herriot
Sun Microsystems
S. Isaacson
Novell, Inc.
P. Powell
San Diego State University
September 3, 1997

Internet Printing Protocol/1.0: Model and Semantics

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

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

Abstract

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

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 components of real world printing solutions. The Internet is a distributed computing environment where requesters of print services (clients, applications, printer drivers, etc.) cooperate and interact with print service providers. This model and semantics document describes a simple, abstract model for IPP even though the underlying configurations may be complex "n-tier" client/server systems. An important simplifying step in the IPP model is to expose only the key objects and interfaces required for printing. The model does not include features, interfaces, and relationships that are beyond the scope of IPP [20].

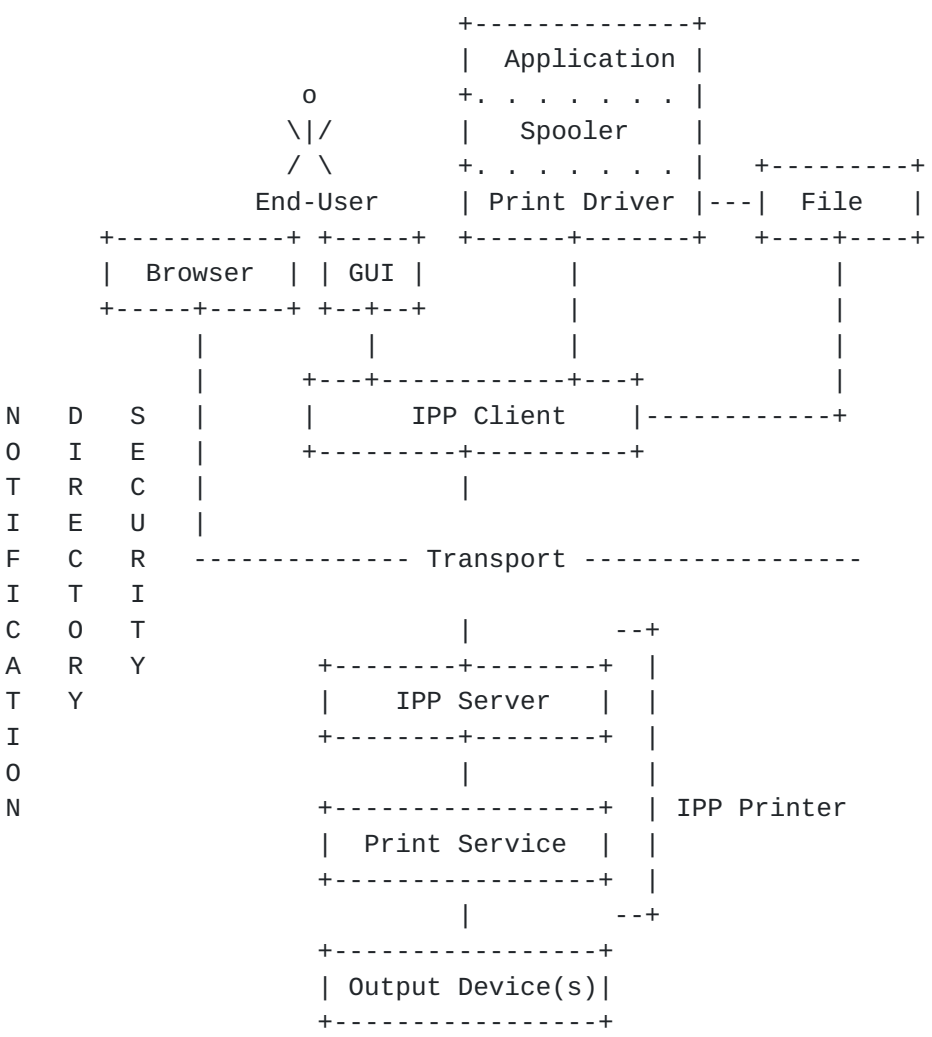
The IPP model encapsulates the important components of distributed printing into three object types:

- 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).

The terminology used in the remainder of this document is defined in Appendix A (section 11) . Terms such as "attributes", "keywords", and "supports" have special meaning in this document and are defined in the model terminology section. . Terms such as "MANDATORY", "MUST", and "OPTIONAL" have special meaning relating to conformance. These terms are defined in the section on conformance terminology, most of which is taken from RFC 2119 [25].

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.



An IPP Printer encapsulates the functions normally associated with physical output devices along with the spooling, scheduling and multiple device management functions often associated with a print server. IPP Printers are optionally registered as entries in a directory where end users find and select them based on some sort of filtered and context based searching mechanism. 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.). The more dynamic information associated with a Printer includes state, currently loaded and ready media, number of jobs at the Printer, errors, warnings, and so forth.

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 print jobs. An IPP server is just that part of the IPP Printer that implements the server-side 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.

When a job is submitted to the Printer and the Printer validates the data in the submission request, the Printer creates 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 that are then routed using the specified 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. For each object instance, the actual set of supported attributes and values describe the specific implementation. The object's attributes and values describe its capabilities, realizable features, job processing functions, and default behaviors and characteristics. For example, the object type "Printer" is defined as a 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 is labeled as:

- "MANDATORY": each object instance MUST support the attribute.
- "OPTIONAL": each object instance MAY support the attribute.

There is no such similar labeling of attribute values. However, if an implementation supports an attribute, it must support at least one (and possibly all) of the possible values for that attribute.

2.1 Printer Object

A major component of the IPP model is the Printer object. An instance of a Printer object implements the IPP protocol. Using the protocol, end users may query the attributes of the Printer and submit Jobs to the Printer (section 3 describes each of the Printer specific operations in detail). The actual implementation components behind

the Printer abstraction may take on different forms and different configurations. However, the model abstraction allows the details of

the configuration of real components to remain transparent to the end user.

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)

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, such as a fax-out device, an 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

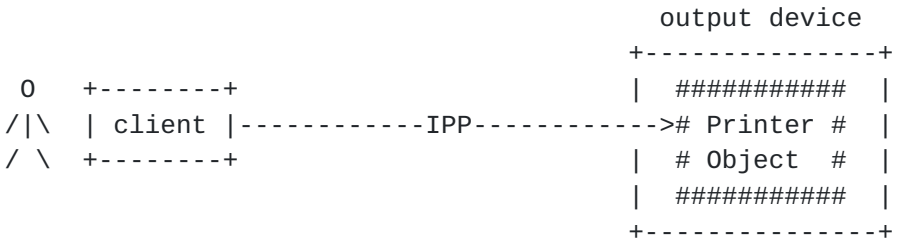
See the following figures for some examples on how to view IPP Printer objects implemented on top of various print system configurations. The embedded case below represents configurations 1 and 2. The hosted and fan-out figures below represent configuration 3.

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.

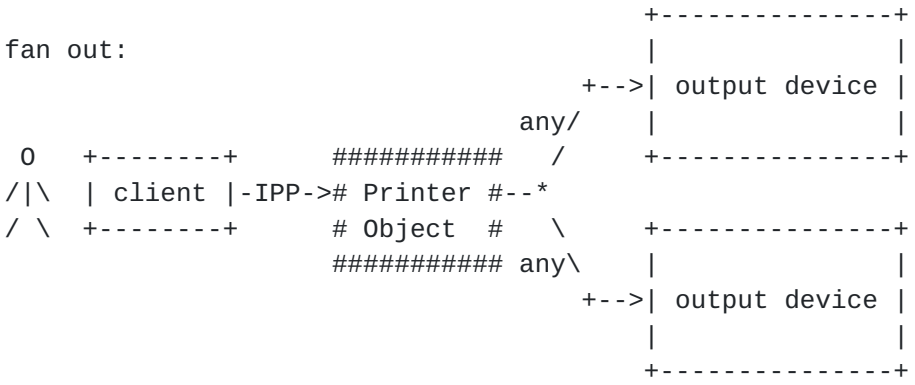
any indicates any network protocol or direct
connect, including IPP

embedded printer:



hosted printer:

fan out:



It is assumed that URIs for IPP Printers are available to end users or programs that wish to interact with the Printer. 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.

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 an IPP Printer. Section 3 describes each of the Job specific operations in detail.

The characteristics and state of an IPP Job are described by its attributes. 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)

2.3 Document Object

A Document object is used to model a document. A Document consists of either printable data or a URI reference to printable data An instance of a Document object is a set of attributes which describe the document data. Document Attributes (see section 4.4). Document attributes only describe the data to be printed; they do not include any specialized document processing instructions. All processing instructions are at the Job level and are called called Job Template attributes and they apply equally to all Documents within a Job.

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, however the

identical Job data could be sent to either the same or a different Printer. In this case, the new Job would be an identical, but different Jobsince it each would have a different Job identifier. A

Job object contains one or more Documents. If the Document is simply a reference to some print data stream, the referenced document may be used in multiple Documents in the same Job or even in different Jobs, though each Document contains its own copy of the reference. 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 [34]. In addition to an identifier, instances of Printers, Jobs, and Documents 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 through some mechanism outside the scope of IPP itself. If the client does not supply Job name or a Document name, the Printer generates a name. In all cases, the name only has local meaning, and it is not constrained to be unique.

To summarize:

- Each Printer will be uniquely identified with a URI. The Printer's "printer-uri" attribute contains the URI.
- Each Job will be uniquely identified with a URI. The Job's "job-uri" attribute contains the URI.
- Each Printer has name (which is not necessarily unique). The administrator chooses and sets this name through some mechanism outside the scope of IPP itself. The Printer's "printer-name" attribute contains the name.
- Each Job has name (which is not necessarily unique). The client optionally supplies this name in the create request. If the client does not supply this name, the Printer generates a name for the Job. The Job's "job-name" attribute contains the name.
- Each Document has name (which is not necessarily unique). The client optionally supplies this name in the request which creates the Document. If the client does not supply this name, the Printer generates a name for the Document. The Document's "document-name" attribute contains the name.

Note: If Documents are printed by reference, the corresponding Document object contains a "document-uri" attribute. The value of

this attribute is a reference to the document data to be printed; it is not a unique identifier of the Document object itself.

Proposal:

All instances of Printer 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. All Printer operations are directed to the Printer's URI. All instances of Job objects have an opaque identifier that when used in combination with its Printer URI, allow it to be persistently and unambiguously referenced. Instances of Document objects have no identifier since they are always created and queried using Job operations; that is the model does not define semantics for querying the attributes of just one Document within a multi-Document Job.

In addition to these identifier attributes, instances of Printer, Job, and Document 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. A client optionally supplies Job and Document names in the operations that create Jobs and Documents. If a client does not supply a name, the Printer generates a name. In all cases, the name only has local meaning, and it is not in any way constrained to be unique.

To summarize:

- Each Printer will be uniquely identified with a URI. The Printer's "printer-uri" attribute contains this URI.
- Each Job will be uniquely identified with a Job ID and a Printer URI. The Job's "job-id" attribute contains the Job ID. The Printer's "printer-uri" attribute contains the Printer's URI.
- Each Printer has name (which is not necessarily unique). The administrator chooses and sets this name through some mechanism outside the scope of IPP itself. The Printer's "printer-name" attribute contains the name.
- Each Job has name (which is not necessarily unique). The client optionally supplies this name in the create request. If the client does not supply this name, the Printer generates a name for the Job. The Job's "job-name" attribute contains the name.
- Each Document has name (which is not necessarily unique). The client optionally supplies this name in the request that creates the Document. If the client does not supply this name, the Printer generates a name for the Document. The Document's "document-name" attribute contains the name.

Note: If Documents are printed by reference, the corresponding Document object contains a "document-uri" attribute. The value of

this attribute is a reference to the document data to be printed; it is not a unique identifier of the Document object itself.

3. IPP Operations

Jobs and Printers each have a set of associated operations. Operations consist of requests and responses. When a client communicates with an IPP object, it issues an operation request to the URI for that object. Each request carries along with it any operation attributes and data required by the object to perform the operation. Each request requires a response from the object. Each response indicates success or failure of the operation with a status code, and may include operation attributes and an optional status message. The representation and encoding of IPP operations is described in "Internet Printing Protocol: Protocol Specification" [23]. This section describes the IPP operations in terms of their semantics and contents, including both the request and the response for each operation.

The Printer operations are fully defined in section 3.2:

- Print-Job (section 3.2.1)
- Print-URI (section 3.2.2)
- Validate-Job (section 3.2.3)
- Create-Job (section 3.2.4)
- Get-Attributes (section 3.2.5)
- Get-Jobs (section 3.2.6)

The Job operations are fully defined in section 3.3:

- Send-Document (section 3.3.1)
- Send-URI (section 3.3.2)
- Cancel-Job (section 3.3.3)
- Get-Attributes (section 3.3.4)

There are no explicit Document operations. However, there are Job operations that are used to create and query Document objects contained within a Job object. The Send-Document and Send-URI Job operations are used to add a new Document to an existing multi-Document Job. If requested, the Get-Attribute operation used for a Job also returns any Document object attributes. If the Job is a single Document Job, only a single set of Document attributes is returned. If however, the Job contains multiple Documents, the Get-Attribute Response contains multiple sets of Document attributes, one

set for each Document.

3.1 General Semantics

3.1.1 Operation Targets

All IPP operations are directed at an instance of an IPP object. For Printer operations, the operation is directed at an IPP Printer using its URI. That is, all Printer operations must somehow contain the target attribute "printer-uri" indicating the target of the operation. In the mapping of IPP over HTTP, the "printer-uri" attribute is actually encoded as the "request-URI" component of the HTTP operation. In other mappings, the target URI might be encoded using some other transport specific mechanism.

For Job operations, the operation is directed at an IPP Job using its URI. Like Printer operations, all Job operations must somehow contain the target attribute "job-uri" indicating the target of the operation. Also, like Printer operations, in the mapping of IPP over HTTP, the "job-uri" attribute is actually encoded as the "request-URI" component of the HTTP operation. In other mappings, the target URI might be encoded using some other transport specific mechanism.

Propsal:

For Job operations, the the operation is directed at the Printer's URI and it is accompanied by the Job ID . This allows the IPP Printer implementation to find the correct Job object. That is, both the "printer-uri" attribute and the "job-id" attribute must both be supplied as the target of every Job operation. Like Printer operations, in the mapping of IPP over HTTP, the "printer-uri" attribute is encoded in the "request-URI" component of the HTTP operation. However, the "job-id" attribute is supplied as an operation attribute within the operation data itself. If a Printer receives an IPP operation, it must check to see if the "job-id" attribute is included among the operation attributes. If it is there, the operation is being directed at the Job identified by the value of "job-id". If it is not there, the operation is being directed at the Printer itself.

3.1.2 Operation Status Codes and Messages

Every operation response returns a MANDATORY status code and an OPTIONAL status message. A status code provides information on the processing of a request. A status message provides a short textual description of the status of the operation. The status code is intended for use by automata and the status message is intended for

the human user. If a response does include a status message,an IPP application is not required to examine or display the status message. Status codes and suggested corresponding status messages are described

in section 12. In most cases, if the status code indicates an error, there are additional attributes in the response that are not returned in the successful case. One such example is the group of unsupported attributes and values. This set of attributes is returned in response to an operation request that includes attributes that are not supported by the object.

3.1.3 Security Concerns for IPPOperations

3.1.3.1 Authenticated Requester Identity

IPP is layered on top of security services that supply the requester's identity. It is assumed that identity supplied by the authentication service is the most authenticated identity required by a given site's configuration and current policy. It is also assumed that the layering allows for a single IPP implementation to be run over a consistent interface that supplies the authenticated identity. The authentication interface should allow for various modular and extensible authentication service implementations without requiring changes to the IPP implementation.

Once the authenticated identity of the requester has been supplied to the IPP implementation, the implementation uses that identity to enforce any authorization policy(ies) that might be in place. When a Job is created, the identity of the requester of the create operation is persistently stored in the Job's "job-originating-user" attribute. This attribute can be used to match the requester's identity of subsequent operations on that Job in order to enforce the local authorization policy(ies), if any. There are operation status codes that allow an implementation to return information back to the operation requester about what has been forbidden, not allowed, or not authorized.

For example, a site security policy might be that only the job owner is allowed to cancel a job using the Cancel-Job operation.

3.1.3.2 Restricted Queries

In many of these IPP 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.

3.1.4 Versions

Each operation request carries with it a version number. Each version number is in the form X.Y where X is the major version number and Y is the minor version number.

By including a version number in the client request, it allows the client (the requester) to identify which version of IPP it is interested in using. If the Printer does not support that version, the Printer responds with a status code of 'server-error-version-not-supported'. There is no version negotiation per se. However, if after receiving a version not supported status code from the Printer, there is nothing that prevents a client from trying again with a different version number. For example, a client might be configured to support IPP version 3.0, 2.5, 2.0 and 1.0 (hypothetically speaking). A client might first try to speak version 3.0. If after receiving a version not supported, it might successively try version 2.5, then 2.0, then 1.0. All implementations MUST support version 1.0.

Items that might affect the changing of the major version number include any changes to the protocol specification itself such as:

- reordering of ordered attributes or attribute sets
- changes to the syntax of existing attributes
- changing OPTIONAL to MANDATORY and vice versa

Items that might affect the changing of the minor version number include any changes to the model objects and attributes but not the protocol specification itself, such as:

- grouping all extensions not included in a previous version into a new version
- formally adding in new attribute values
- changing any of the type1 attributes

3.1.5 Job Creation Operations

In order to create a new Job object, a client issues a create request. A create request is any one of following three operation requests:

- The Print-Job Request: A client that wants to create a Job with only a single Document uses the Print-Job operation. . The operation allows for the client to "push" the document data to

the Printer by including the document data in the request itself.

- The Print-URI Request: A client that wants to create a Job with only a single Document where the Printer "pulls" the document data uses the Print-URI operation. In this case, the client includes only a URI reference to the document data (not the document data itself).

- The Create-Job Request: A client that wants to create a Job with multiple Documents uses the Create-Job operation. . 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 in the request (the client "pushes" the document data to the printer), and the Send-URI operation includes only a URI reference to the document data in the request (the Printer "pulls" the document data from the referenced location). The last Send-Document or Send-URI request includes a "last-document" attribute set to 'true' indicating that this is the last Document for this Job.

Throughout this model specification, the term "create request" is used to refer to any of these three operation requests.

A Create-Job operation followed by 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. Also, Print-Job is a MANDATORY operation (all implementations MUST support it) whereas Create-Job is an OPTIONAL operation, hence some implemations might not support it.

Appendix E: Processing IPP Attributes (see Section 15) describes the rules and issues surrounding either the acceptance or rejection of a create request.

At job submission time, the Printer does not need to validate the document data or the actual contents of the URI reference to document data (in the case of a Print-URI). The Printer SHOULD check the syntax of the URI to make sure that it appears to be a valid URI. However, other checks are essentially useless,since they require actually parsing and interpreting all of the print data, or in the case of a URI, checking for availability at Job submission time does not guarantee availability at Job processing time. In other words, even if the checks were made a Job submission time, they MUST also be done later at Job processing time, since at job processing time, the Printer might discover any of the following conditions:

- additional runtime errors in the document data,
- some nested document data is in an unsupported format,

- the URI reference is no longer valid (i.e., the server hosting the data might be down), or
- any other run-time job processing error

At this point, the Printer is unable to return an error status code in the create response, since it had already sent the create response with a successful status code. In this case, the Printer, depending on the error, can set the "job-state", "job-state-reasons", or "job-state-message" attributes to the appropriate value(s) so that later queries can report the correct job status. Also, if the client has requested notification for 'job-problems', the Printer could notify the client via the address(es) in the "notify-addresses" attribute.

3.2 Printer Operations

3.2.1 Print-Job Operation

When an end user desires to submit a print job with only one Document, the client uses a Print-Job operation. 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. A Validate-Job operation contains no data, only a set of Job Template and Document attributes whose values can be validated for consistency. A Create-Job operation is similar as well, however the Create-Job Request is used to create a multi-Document Job.

3.2.1.1 Print-Job Request

The following sets of attributes are supplied as as part of the Print-Job Request:

Operation Attributes

"job-name" (name):
This OPTIONAL attribute identifies the client supplied Job name. If this attribute is supplied, its value it used for the "job-name" attribute of the newly created Job object. If this attribute is not supplied by the client, the Printer generates a name to use in the "job-name" attribute of the newly created Job object (see Section 4.3.4).

"ipp-attribute-fidelity" (boolean):
This is an OPTIONAL attribute. If not supplied, the Printer

assumes the value is 'false'. The value 'true' indicates that total fidelity to client supplied attributes and values is required. The value 'false' indicates that a reasonable attempt

to print the Job is acceptable. All implementations support both types of job processing. See section 15 for a full description of "ipp-attribute-fidelity" and its relationship to other attributes, especially the Printer's "pdl-override" attribute.

"document-name" (name):
This OPTIONAL attribute identifies the client supplied Document name. If this attribute is supplied, its value it used for the "document-name" attribute of the newly created Document object. If this attribute is not supplied by the client, the Printer optionally generates a name to use in the "document-name" attribute of the newly created Document object (see Section 4.4).

ISSUE: Since this is part of a Print-Job that creates only a single Document, should we
not allow this attribute?

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 Print-Job Request, the Printer uses the values set in its default value attributes at job processing time (not job submission time).

Document Attributes:

An optional set of Document Attributes as defined in section 4.4. If "document-format" is not supplied, the Printer assumes that the document data is in a format that this acceptable to the Printer. 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.

Document Content

The client MUST supply the document data to be processed.

Note: Since all Print-Job Request attributes are OPTIONAL, the simplest Print-Job Request consists of just the Document Content and

nothing else. In this case, the Printer creates a new Job object and stores a generated Job name in the "job-name" attribute. The Job contains a single Document. The Printer optionally generates a name

the Document and stores it in the "document-name" attribute. The Printer assumes that the document data is in a format supported by the Printer. When the Printer processes the Job, it uses all of its default values for the missing Job Template attributes.

3.2.1.2 Print-Job Response

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

Operation Attributes

"job-uri" (uri):
A URI which the client SHALL use for all other operations on this Job. This is the MANDATORY "job-uri" attribute.

ISSUE: Can this just be a "Job Status Attribute"?

Job Status Attributes

"job-name":
This is the Job's "job-name" attribute.

"job-state":
This is the Job's "job-state" attribute. The value of this 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.

"job-state-reasons":
This is the Job's "job-state-reasons" attribute. The value of this 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.

"job-state-message":
This is the Job's "job-state-message" attribute. Since the "job-state-message" attribute is OPTIONAL Job attribute, itis OPTIONALLY included in the response.

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:

> This is a set of attributes and attribute values that are unsupported. This output parameter is only returned in the response if the status code indicates that there was such an error.

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

3.2.2 Print-URI Operation

This operation is identical to the Print-Job operation (section 3.2.1) except that a client supplies a URI reference to the document data using the "document-uri" (uri) operation attribute rather than including 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.

3.2.3 Validate-Job Operation

This operation is similar to the Print-Job operation (section 3.2.1) except that a client supplies no document data and the Printer allocates no resources (i.e., it does not create a new Job object). This operation is used only to verify capabilities of a printer object against whatever attributes are supplied by the client in the Validate-Job request. There is no "job-uri" attribute returned in the Validate-Job Response neither are there any Job Status attributes returned in the response. The client MAY include a "document-uri" attribute in the request. In this case, the Printer SHOULD only validate the syntax of the URI rather than follow the reference and validate the contents of the reference. If all is well, the Printer returns the status code "successful-ok". Otherwise, the Printer returns a set of unsupported attributes and the appropriate error status code.

3.2.4 Create-Job Operation

This operation is similar to the Print-Job operation (section 3.2.1) 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. If a Printer object supports the Create-Job operation, it MUST also support either

the Send-Document operation or the Send-URI operation or both. Since a client can query the Printer's "operations-supported" attribute, a client SHOULD NOT attempt to use an unsupported optional operation.

3.2.5 Get-Attributes Operation

The Printer Get-Attributes operation allows a client to obtain information from a Printer object. In the operations attributes in a request, the client supplies the set of attribute names and/or attribute group names in which the requester is interested. In the operation attribute in the response, the Printer returns a corresponding attribute set with the appropriate attribute values filled in.

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 is also the special group 'all' which includes all supported attributes.

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

3.2.5.1 Get-Attributes Request

The following sets of attributes are part of the Get-Attributes Request when the request is directed to a Printer object:

Operation Attributes"requested-attributes" (1setOf keyword) :
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'.

"document-format" (mimeType) :
This input parameter is useful for determining the set of supported attribute values which relate to the requested 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 get the Printer to 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 'application/postscript' (for

PostScript) and 'text/plain' (for text) 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.

ISSUE: What about MIME types.

3.2.5.2 Get-Attributes Response

The Printer returns the following sets of attributes as part of the Get-Attributes Response:

Requested Attributes

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

3.2.6 Get-Jobs Operation

The Get-Jobs operation allows a client to retrieve the 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 possibly more than one object (see the description of attribute groups in section 3.3.4).

3.2.6.1 Get-Jobs Request

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

The following sets of attributes are part of the Get-Jobs Request:

Operation Attributes

"limit" (integer):
   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 spooled jobs, only the first 50 jobs are returned; the other 25 jobs are not returned.

"requested-attributes" (1setOf keyword):
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'.

3.2.6.2 Get-Jobs Response

The Printer returns zero or more Job objects each with zero or more attributes. There is a set of requested attributes for each Job. After each Job, there is a set of requested attributes for each Document in the Job. For example:

```
Job 1
     Document 1A
Job 2
     Document 2A
     Document 2B
     Document 2C
Job 3
     Document 3A
Job 4
Job 5
     Document 5A
     Document 5B
```

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). Next, 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). Note: Jobs

are returned in the following order:

deBry, Hastings, Herriot, Isaacson, Powell

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

Second, 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).

3.3 Job Operations

The target of Job operations are Job objects. Since Job objects are identified with both the Printer's URI and the "job-id" attribute, the following applies rules apply to all Job operations:

- The Printer's URI is encoded at the transport level the same as all Printer operations
- The "job-id" attribute is included as an operation attribute for all Job operations.

3.3.1 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 Request.

3.3.1.1 Send-Document Request

The client submits the request to a Job URI.

The following attribute sets are part of the Send-Document Request:

Operation Attributes:

"document-name" (name):
This OPTIONAL attribute identifies the client supplied Document name. If this attribute is supplied, its value it used for the "document-name" attribute of the newly created Document object. If this attribute is not supplied by the client, the Printer optionally generates a name to use in the "document-name"

attribute of the newly created Document object (see Section ??).

"last-document" (boolean):

This is a boolean flag that is set to 'true' if this is the last Document for the Job.

Document Attributes:

An optional set of Document Attributes as defined in section 4.4. If "document-format" is not supplied, the Printer assumes that the document data is in a format that this acceptable to the Printer. 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.

Document Content

The client MUST supply the document data to be processed.

3.3.1.2 Send-Document Response

The following sets of attributes are part of the Send-Document Response:

Job Status Attributes

"job-name":
This is the Job's "job-name" attribute.

"job-state":
This is the Job's "job-state" attribute. The value of this 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.

"job-state-reasons":
This is the Job's "job-state-reasons" attribute. The value of this 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.

"job-state-message":
This is the Job's "job-state-message" attribute. Since the "job-state-message" attribute is OPTIONAL Job attribute, itis OPTIONALLY included in the response.

Unsupported Attributes:

This is a set of attributes and attribute values that are unsupported. This output parameter is only returned in the

response if the status code indicates that there was such an error.

3.3.2 Send-URI Operation

This operation is identical to the Send-Document operation (see section 3.3.1) except that a client supplies a URI reference ("document-uri" operation attribute) 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.

3.3.3 Cancel Job Operation

This operation allows a client to cancel a Print Job any time after the print job has been submitted to the Printer. Since a Job might already be printing by the time a Cancel-Job is received, some pages may be printed before the job is actually terminated.

3.3.3.1 Cancel-Job Request

The client submits the request to a Job URI.

The following attribute sets are part of the Cancel Job Request:

Operation Attributes

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

3.3.3.2 Cancel-Job Response

There are no attributes in the Cancel Job Response other than the Status Code and optional Status Message. Once a successful response has been sent, the implementation guarantees that the Job will eventually end up in the 'cancelled' state. If the implementation is forced to leave the job in the some other state, the "job-state-reasons" SHOULD contain the 'processing-to-stop-point' value which indicates to later queries that although the Job may still be 'processing', it will eventually end up in the 'cancelled' state, not the 'completed' state.

3.3.4 Get-Attributes Operation

The Job Get-Attributes operation allows a client to obtain information from a Job object and it is almost identical to the Get-Attributes

operation for a Printer object (see section 3.2.5). The only differences are that the operation is directed at a Job rather than a Printer and there is no "document-format" operation attribute used when querying a Job object.

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 attributes specified in Section 4.3.

There is also the special group 'all' which includes all supported attributes.

There is another special group named 'document-attributes' which consists of the Document Attributes described in section 4.4. If any Document Attributes are requested, the response contains a separate set of Document attributes for each Document in the Job.

3.3.4.1 Get-Attributes Request

The following attribute sets are part of the Get-Attributes Request when the request is directed to a Job object:

Operation Attributes:

"requested-attributes" (1setOf keyword) :
An optional set of attribute names (without values) or attribute group names in whose values the requester is interested. If the client omits this attribute, the Printer SHALL respond as if this input parameter had been supplied with a value of 'all'.

3.3.4.2 Get-Attributes Response

The implementation returns the following response attributes as part of the Get-Attributes Response:

Job Attributes:

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

Document Attributes:

deBry, Hastings, Herriot, Isaacson, Powell

This is the set of requested attributes and their current values. The implementation ignores (does not respond with) any requested attribute which is not supported. One set is returned for each Document in the Job.

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 11.2.1). 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

This section defines the basic syntax types that a client and server SHALL be able to handle. These are considered to be a set of type2 enum values. The values are:

'1' 'text': a sequence of characters, length: 0 to 4095, where each character is a member of the ISO 10646 [??] coded character set using the UTF-8 character encoding scheme [28].. This syntax type is used for free form human readable text intended for human consumption.

'2' 'name': this is the same a "text" except that the sequence of characters is of length 1 to 255 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.

'3' 'keyword': a sequence of characters, length: 1 to 255,

containing only the characters ASCII lowercase letters ("a" - "z"), ASCII digits ("0" - "9"), hyphen ("-"), dot ("."), 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.

'4' '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.

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

'6' '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://...).

'7' 'human-language': a standard identifier for human language and optionally a country. The values for this syntax type are taken from RFC 1766 [26]. Independent of the human language, all text strings are strings of characters where each character is a member of the ISO 10646 [??] coded character set. Text strings are

encoded using the UTF-8 character encoding scheme [28]

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

'9' '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.

'10' '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).

'11' 'dateTime': a standard, fixed length representation of date and time as defined in RFC 1514 [32] and RFC 1903 [33].

'12' 'resolution': a special syntax used only for "printer-resolution" and "printer-resolution-supported" It consists of 3 parts: a cross feed direction resolution (positive integer value), a feed direction resolution (positive integer value) and a units value. All these are taken from the Printer MIB [1] suggested values.

'13' 'mimeType': MIME type values as defined by RFC 2045 [??].

'14' '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).

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

4.2 Job Template Attributes

Job Template attributes describe job processing behavior. Job Template attributes are OPTIONAL (see section 11.2.3 for a description of support for OPTIONAL attributes).

Job Template attributes conform to the following rules. For each Job Template attribute called "foo":

1. The Printer supports a "foo-supported" attribute that describes which job processing behaviors are supported by a Printer. A client can query the Printer and find out what behaviors are supported by inspecting the values of the "foo-supported" attribute.

2. The Printer also supports a default value attribute named "foo". 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). However, if the Printer supports either the "foo" default value attribute or the "foo-supported" attribute, the Printer MUST support both.

3. "foo" is also optionally supplied by the client in a create request. If "foo" is supplied, the client is specifying that the Printer SHALL apply thecorresponding job processing behavior to this Job while processing the Job. When "foo" is not supplied, the client is specifying that the Printer apply the default job processing behavior. Since an administrator could change the default value after the Job has been submitted (while it is waiting to be processed), the default value in affect a job processing time is used for processing a Job with no client supplied attribute.

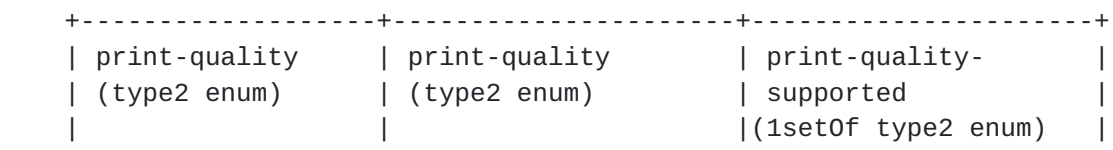
If an application wishes to present an end user with a list of supported values from which to choose, the application SHOULD query the Printer for the supported values. The application SHOULD also query the default value attributes. If the application then limits selectable values to only those value that are supported, application guarantees that the values supplied by the client in the create request all fall within the set of supported values at 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 "job-template" group in order to get the complete set supported attributes (both supported value and default value attributes).

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 which is equal to or smaller than the maximum range of 1 to 100. The default value attribute contains the job priority value that will be used for a new job if the client does not supply a value 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 and relationships for all Job Template attributes. The first column of the table (labled "Job") shows the name and syntax for each Job Template attribute in the Job object. These are the attributes that can optionally be supplied by the client in a create request. The last two columns (labled "Printer: Default Value" and "Printer: Supported Values") shows the name and syntax for each Job Template attribute in the Printer object (the default value attribute and the supported values attribute). A "No" in the table means the Printer SHALL NOT support the attribute (the attribute is simply not applicable). The second part of the table lists Document level attributes. Document level attributes have the same semantics as Job Template

Job Attribute	Printer: Default Value Attribute	Printer: Supported Values Attribute
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 uriScheme)
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)
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 (type2 keyword)	sides (type2 keyword)	sides-supported (1setOf type2 keyword)
printer-resolution (type2 enum)	printer-resolution (type2 enum)	printer-resolution-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 (2setOf integer 1 - MAX)	copies-supported (2rangeOf integer 1- MAX)
page-range (rangeOf integer)	No	page-range-supported (boolean)
Note: The following attributes are Document attributes, they have the same semantics a Job Template attributes, however they apply to each Document object rather than at the Job level as Job object attributes.		

Document Attribute	Printer: Default Value Attribute	Printer: Supported Values Attribute
document-format (mimeType)	document-format (mimeType)	document-format-supported (1setOf mimeType)
compression (type3 keyword)	No	compression-supported (1setOf type3 keyword)
document-k-octets (integer)	No	job-k-octets-supported (rangeOf integer)
document-impressions (integer)	No	job-impressions-supported (rangeOf integer)
document-media-sheets (integer)	No	job-media-sheets-supported (rangeOf integer)

4.2.1 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': one or more site specific standard job sheets are printed, e.g. a single start sheet or both start and end sheet 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.

Note: The effect of this attribute on jobs with multiple documents is controlled by the "multiple-document-handling" job attribute (section 4.2.6).

4.2.2 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 (i.e., when the job leaves the 'processing' state and enters the 'processing-stopped' state).
'job-started-processing': the Printer SHALL notify when the Printer starts processing the Job (i.e., when the job leaves the 'pending' state and enters the 'processing' state).
'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.

ISSUE: Need to add generic alerts from the Printer MIB.

4.2.2.1 Event Notification Content

The content of each event notification is:

```
*time: <US ASCII string representing the absolute time
            of the event>CRLF
event: <keyword for the event>CRLF
printer-uri: <Printer's URI>CRLF
printer-state: <keyword for printer state>CRLF
*printer-state-reason: <set of comma separated keywords>CRLF
*job-id: <US ASCII string representing the Job Id>CRLF
*job-state: <keyword for job state>CRLF
*job-state-reason: <set of comma separated keywords>CRLF
*message: <US ASCII string>CRLF
```

The contents of the event notification are intended for machine consumption more than human consumption, however the use of US ASCII enables humans to retrieve some semantics from the event report itself. Items above marked with and asterisk ('*') are optional. If the implementation does not support any notion of time, then field is not present in the notification body. If the event is Printer related, only the Printer related fields are included. If the event is Job related, all of the Job fields SHOULD be included.

4.2.3 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 uriScheme values are:

'mailto': the Printer sends a text message via email to the specified email address
'http': the Printer sends an HTML formatted message via an HTTP POST method to the specified URI
'ftp': the Printer sends a text message via an FTP `append' command to the specified remote file.

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

This attribute specifies a priority for scheduling the Job. A higher value specifies a higher priority. The value 1 indicates the lowest possible priority. The value 100 indicates 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.

4.2.5 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 that keep it in the 'pending-held' state remain, the Printer 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 specified time period is in effect has already started , the job SHALL be a candidate for processing immediately.

4.2.6 multiple-document-handling (type2 keyword)

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

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

4.2.7 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 that 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 that 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.

ISSUE: What should we do about "media-ready"?

Standard values are (taken from ISO DPA and the Printer MIB) and are

listed in section 14.

4.2.8 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 pages 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 pages 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.

4.2.9 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', and 'duplex' all work the same for portrait or landscape. However '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.

4.2.10 printer-resolution (resoultion)

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

The value is a special type consisting of a pair of integers and a value which specifies the units for the two integers. The three values are the same as those specified in the draft-ietf-printmib-mib-info-02.txt as prtMarkerAddressabilityFeedDir (the resolution in the feed direction), prtMarkerAddressabilityXFeedDir (the resolution in the cross feed direction), and prtMarkerAddressabilityUnit (the units of the first two values, namely dots per inch and dots per centimeter).

4.2.11 print-quality (type2 enum)

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

The standard values are:

'3' 'draft': lowest quality available on the printer
'4' 'normal': normal or intermediate quality on the printer
'5' 'high': highest quality available on the printer

4.2.12 finishings (1setOf type2 enum)

This attribute identifies the finishing operations that the Printer uses for each copy of each printed document in a particular Job. For Jobs with multiple documents, the "multiple-document-handling" attribute determines what constitutes a _copy_ for purposes of finishing.

Standard values are:

'3' 'none': Perform no finishing
'4' 'staple': Bind the document(s) with one or more staples. The exact number and placement of the staples is site-defined.
'5' 'staple-top-left': Place one or more staples on the top left corner of the document(s).
'6' 'staple-bottom-left': Place one or more staples on the

bottom left corner of the document(s).

'7' 'staple-top-right': Place one or more staples on the top right corner of the document(s).

'8' 'staple-bottom-right': Place one or more staples on the bottom right corner of the document(s).

'9' 'saddle-stitch': 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.

'10' 'edge-stitch': 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.

'11' 'punch': 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.

'12' 'cover': 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.

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

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

This attribute specifies the number of copies to be printed. On many devices the supported number of collated copies will be limited by the number of physical output bins on the device, and may be different from the number of uncollated copies which can be supported. Therefore the copies-supported attribute specifies a set of ranges; the first defines the supported range of values for uncollated copies, and the second the supported range of values for collated printing. The set of default values for copies specify the default number of uncollated copies followed by the default number of collated copies.

The effect of this attribute is controlled by the "multiple-documents-handling" attribute (section 4.2.6).This attribute specifies the number of copies of the job to be printed.

4.2.14 page-range (rangeOf integer)

This attribute specifies the pages of a document which are to be printed. In most cases, the exact pages to be printed will be generated by a device driver and this attribute would not be required.

However, when printing an archived document which has already been formatted, the end user may elect to print just a subset of the pages contained in the document. In this case, if page-range = n.m is

specified, the first page to be printed will be page n. All subsequent pages of the document will be printed through and including page m.

Page-range supported is a boolean value indicating whether or not the printer is capable of supporting the printing of page ranges. This capability may differ from one PDL to another. The page-range default value is always zero (0) and indicates that all pages of the document will be printed if a page-range is not specified.

4.2.15 orientation (type2 enum)

This attribute specifies the orientation of the content on the output pages to be printed. In most cases, the orientation of the content is specified within the document format generated by the device driver at print time. However, some document formats (such as "text") do not support the notion of page orientation, and it is possible to bind the orientation after the document content has been generated. This attribute provides an end user with the means to specify orientation for such documents.

Standard values are:

'1' 'portrait': The content will be printed across the short edge of the media.
'2' 'landscape': The content will be printed across the long edge of the media.

4.2.16 document-format (mimeType)

This attribute defines the document format of the data to be printed. The standard values for this attribute are MIME types. Since the complete list is rather long, the full enumeration of standard values is found in section 13 APPENDIX C: "document-format" values.

If the "document-format" is unknown for a certain document, the client does 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 (not the operation data).

Standard values are :

'none': no compression is used.
'zip':ZIP (inflate/deflate) compression technology
`gzip' GNU zip compression technology described in RFC 1952.

'compress': UNIX compression technology

ISSUE: Look what HTTP defines.

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. For these three attributes, the Printer may try to compute the value if it is not supplied in the create request. Even if the client does supply a value for this attribute in the create request, the Printer may choose to change the value if the Printer is able to compute a value which is more accurate than the client supplied value. The Printer may be able to determine the correct value for this attribute either right at job submission time or at any later point in time. If the value of this attribute is unknown, the Printer may choose to respond with a value of '-2' (which signifies "unknown") rather than choose to not support the attribute at all.

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

This attribute specifies the total number of impressions for this job.

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

This attribute specifies the total number of media sheets used by this job.

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-id (this or job-uri)	32bit unsigned	MANDATORY
job-more-info	uri	
job-name	name	MANDATORY
job-originating-user	name	MANDATORY
job-originating-host	name	
user-human-language	human-language	
job-state	type1 enum	MANDATORY
job-state-reasons	1setOf type2 keyword	
job-state-message	text	
output-device-assigned	name	
time-at-pending	seconds	
time-at-processing	seconds	
time-at-completed	seconds	
number-of-intervening-jobs	integer	
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-id (32bit unsigned integer)

This attribute contains the ID of the job. The Printer, on receipt of a new job, generates an ID which identifies the new Job on that Printer. The Printer returns the value of the "job-id" attribute as part of the response to a create request.

ISSUE: job-uri or job-id?

4.3.3 job-uri-user (uri)

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

4.3.4 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. The Job's "job-name" attribute is set to the value supplied by the client in the "job-name" operation attribute in the create request. If, however, if it is not supplied by the client in the create request, the Printer, on creation of the Job, SHALL generate a name. The Printer can generate the name using any method convenient to the application. The Printer MAY choose to use the value of the "document-name" attribute of the first (or only) Document (or any other piece of Job specific information) as a basis for generating a Job name.

ISSUE: There has been some suggestion to add a "job-user-label". The idea for this attribute was to allow a client to supply some meaningful label to be applied to the Job independent of the Job URI or the Job ID. However, after reviewing the semantics of "job-name", the two sound identical.

4.3.5 job-originating-user (name)

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

4.3.6 job-originating-host (name)

This attribute identifies the originating host of the job. The Printer sets this attribute to the most authenticated host name it can obtain

from the protocol over which the operation was received from the client.

4.3.7 user-human-language (human-language)

This attribute identifies the human language and optionally 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 human language it SHOULD use for translating ALL text strings that it sends back to the end user. The Printer has a "human-languages-supported" supported values attribute and a "human-language" default value attribute.

ISSUE: This should be a Job Template attribute.

4.3.8 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.

ISSUE: Suggested change to (but this must be synchronized with JMP)

'processing'(5): One of:

1. the job is using, or is attempting to use hardware devices that are making marks on a medium, and/or performing finishing, such as stapling OR
2. the job is using, or is attempting to use software processes that are analyzing or interpreting a PDL without making marks on a medium.
3. 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.

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.

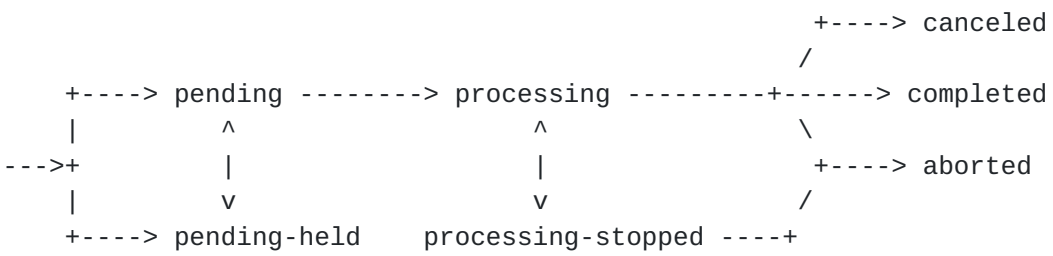
'aborted'(8): The job has been aborted by the system, usually

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.



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.

4.3.9 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:

'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-time': 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.
'job-hold-until-resources-are -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).
'job-interpreting': Job is in the 'processing' state, but more

specifically, the Printer is interpreting the data.
'job-printing': Job is in the 'processing' state, but more specifically, the Printer is actually printing (making marks on the media).

'job-queued': Job is in the 'processing' state, but more specifically, the Printer is queuing the data.
'job-transforming': Job is in the 'processing' state, but more specifically, the Printer is transforming the data.

ISSUE: Show a partitioning of which "job-state-reasons" are valid or expected for each "job-state" value.

4.3.10 job-state-message (text)

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

4.3.11 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.

4.3.12 time-since-pending (integer)

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

4.3.13 time-since-processing (integer)

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

4.3.14 time-since-completed (integer)

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

4.3.15 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.

4.3.16 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.

4.3.17 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 described in the Job Monitoring MIB [27]). That is, if the "job-state" is 'processing' or 'processing-stopped', this value is intended to contain the amount of the job that has been processed to the time at which the attributes are requested. For any of these three attributes, the Printer may choose to return the value '-2' (which represents "unknown") rather than choose to not support the attribute at all.

4.3.18 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.

4.3.19 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 applies to Document objects. If there are multiple Documents in a Job, then there is a different set of Document attributes associated with Document in the Job.

When using the Get-Attributes or Get-Jobs operations, the group named "document-attributes" includes all of the attributes in this section,

including the Document level Job Template attributes (attributes that have the same characteristics as Job Template attribute, yet apply to

Documents, not Jobs). These are identified in the second part of the table in section 4.2. Specifically they are:

"document-format"
"compression"
"document-k-octets"
"document-impressions"
"document-media-sheets"

Other Document attributes include:

Document Attribute	Syntax	MANDATORY?
document-name	name	NO
document-uri	uri	NO

4.4.1 document-name (name)

This attribute contains the name of the document. The name is optionally supplied by the client in the "document-name" operation attribute in the create request. If it is not supplied by the client in the create request, the Printer MAY generate a Document name. The "document-name" attribute contains the name whether supplied or generated.

4.4.2 document-uri (uri)

If a "pull" operations is used to create the Document object (such as the Print-URI or Send-URI operations which include only a URI reference to the document data, not the document data itself), then this "document-uri" attribute contains the URI reference. This attribute is populated from the "document-uri" operation attribute supplied by the client in the Print-URI or Send-URI operation.

If a "push" operation is used to create the Document object (such as the Print-Job or Send-Document operations which include the document data rather than just a reference to the data) then this "document-uri" attribute is not associated with the Document object.

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-name	name	MANDATORY
printer-location	text	
printer-description	text	
printer-more-info	uri	
printer-driver-installer	uri	
printer-make-and-model	text	
printer-more-info-manufacturer	uri	
printer-state	type1 enum	MANDATORY
printer-state-reasons	1setOf type2 keyword	
printer-state-message	text	
operations-supported	1setOf type2 enum	MANDATORY
printer-is-accepting-jobs	boolean	MANDATORY
queued-job-count	integer	
printer-message-from-operator	text	
printer-language	human-language	MANDATORY
printer-language-supported	1setOf human-language	MANDATORY
color-supported	boolean	
pdl-override	type2 keyword	
message-protection-	keyword	

supported		
authentication-author ization-supported	keyword	

printer-up-time	seconds	MANDATORY
printer-current-time	dateTime	

4.5.1 printer-uri (uri)

This attribute contains the URI for the printer. An administrator determines a printer's URI and sets this attribute to that URI. The precise format of a printer URI is implementation dependent.

4.5.2 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 determines a printer's name and sets 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.

4.5.3 printer-location (text)

This attribute identifies the location of this printer. This could include things like: _in Room 123A, second floor of building XYZ_.

4.5.4 printer-description (text)

This attribute identifies the descriptive information about 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".

4.5.5 printer-more-info (uri)

This attribute contains a URI used to obtain more information about this specific printer. For example, this could be an HTTP type URI referencing an HTML page accessible to a Web Browser. 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 specific services (e.g. job pricing, services offered, end user assistance).

4.5.6 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.

4.5.7 printer-make-and-model (text)

This attribute identifies the make and model of the printer.

4.5.8 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.

4.5.9 printer-state (type1 enum)

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 keep this attribute set in a timely manner to the value in the table below which most accurately reflects the state of the Printer. If the printer has jobs that are requesting notification of printer-problems or job-problems, then _timely manner_ means continually.. Otherwise, _timely manner_ means whether the Printer receives a query for this attribute. 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.

4.5.10 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.
'marker-supply-low': The Printer is low on marker supply.
'spool-area-full': The limit of persistent storage allocated for spooling has been reached.

ISSUE: Show a partitioning of which "printer-state-reasons" are valid or expected for each "printer-state" value.

4.5.11 printer-state-message (text)

This attribute specifies the additional information about the printer state and printer state reasons in human readable text.

4.5.12 operations-supported (1setOf type2 enum)

This attribute specifies the set of supported operations for this Printer.

The following standard values are defined:

0x00 reserved, not used
0x01 Print-Job
0x02 Print-URI
0x03 Validate-Job
0x04 Create-Job
0x05 Get-Jobs
0x06 Get-Attributes
0x07 Send-Document

```
  0x08    Send-URI
  0x09    Cancel-Job
  0x0A-0x3FFF       reserved for future operations
```

0x4000-0xFFFF reserved for private extensions

In order for IPP to be extensible, this set of operations is defined to be a 1setOf type2 enum (see section 6.1 for a description of typed extensions). Since the base type is enum, the possible full range of values is 0x0000-0xFFFF, however a special range of 0x4000-0xFFFF has been reserved for private extensions. This allows for certain vendors to implement private extensions that are guaranteed to not conflict with future registered extensions. However, there is no guarantee that two or more private extensions will not conflict.

4.5.13 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.

4.5.14 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 is set by the Printer.

4.5.15 printer-message-from-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-human-language (human-language)

This attribute specifies the current human-language that the Printer is operating in.

4.5.17 printer-human-language-supported (1setOf human-language)

This attribute specifies the supported human languages 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)

This attribute expresses the ability for a particular Printer implementation to either attempt to override print data instructions with IPP attributes or not.

This attribute takes on the following values:

- 'attempted': This value indicates that the Printer attempts to make sure that IPP attribute values take precedence over embedded instructions in the Print data, however there is no guarantee.
- 'not-attempted': This value indicates that the Printer makes not attempt to ensure that IPP attribute values take precedence over embedded instructions in the print data.

This is a MANDATORY attribute.

Appendix E: Processing IPP Attributes (see Section 15) contains a full description of how this attribute interacts with and affects other IPP attributes, especially the "ipp-attribute-fidelity" attribute.

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.

4.5.21 printer-up-time (seconds)

This attribute is a MANDATORY attribute. It indicates the amount of time (in seconds) that this instance of this Printer implementation has been up and running. This value is used to populate the Job attributes "time-at-pending", "time-at-processing", and "time-at-completed". These time values are all measured in seconds and all have meaning only relative to this attribute, "printer-up-time".

ISSUE: Does this need to be MANDATORY?

4.5.22 printer-current-time (dateTime)

This attribute is an OPTIONAL attribute. It indicates the current

absolute wall-clock time. If an implementation supports this attribute, the a client could calculate the absolute wall-clock time each Job's "time-at-pending", "time-at-processing", and "time-at-

completed" attributes by using both "printer-up-time" and this attribute, "printer-current-time". If an implementation does not support this attribute, a client can only calculate the relative time of certain events based on the MANDATORY "printer-up-time" attribute.

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 Client Conformance Requirements

A conforming client SHALL send operations that conform to the protocol defined in _Internet Printing Protocol/1.0: Protocol Specification_ [23]. For each parameter or attribute included in an operation request, a conforming client SHALL send a value whose type and value syntax conforms to the requirement of this document

Otherwise, 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 parameters, 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 parametes, attributes, or values that it does not understand.

5.2 Printer Object Conformance Requirements

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

5.2.1 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

5.2.2 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:

Print-Job (section 3.2.1)	MANDATORY
Print-URI (section 3.2.2)	OPTIONAL
Validate-Job (section 3.2.3)	MANDATORY
Create-Job (section 3.2.4)	OPTIONAL
Get-Jobs (section 3.2.6)	MANDATORY
Get-Attributes (section3.2.5)	MANDATORY

For a Job object:

Send-Document (section 3.3.1)	OPTIONAL
Send-URI (section 3.3.2)	OPTIONAL
Cancel-Job (section 3.3.3)	MANDATORY
Get-Attributes (section 3.3.4)	MANDATORY

Conforming Printer implementations SHALL support all request and response parameters and all values of such parameters, except for parameters which are collections of attributes. The following section on attributes specifies the support required for attributes.

5.2.3 Attributes

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

If a Printer supports an attribute, it SHALL support only those values specified in this document or through the extension mechanism

described in the next section. It MAY support any non-empty subset of these values. That is, it SHALL support at least one of the specified values and at most all of them.

5.2.4 Printer extensions

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

A conforming Printer SHALL send responses that conform to the protocol defined in _Internet Printing Protocol/1.0: Protocol Specification_ [23]. For each parameter or attribute included in an operation response, a conforming printer SHALL send a value whose type and value syntax conforms to the requirement of this document

5.2.5 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.

5.3 Security Conformance Requirements

ISSUE: The "why" and "what" needs to be moved from the Security document to this document. The "how" from the Security document needs to be moved to the Protocol Specification document.

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 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 list 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.

By definition, each of the four types above assert some sort of registry or review process in order for extensions to be considered valid. Each higher level (1, 2, 3, 4) tends to be decreasingly less stringent than the previous level. Therefore, 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) fully qualified 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, the 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. Labels are separated by the "." character.

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

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

The "document-format" attribute's syntax is "mimeType". This means that valid values are MIME types. RFC 2045 [??] defines the syntax for valid MIME types. Also, IANA is the registry for all MIME types.

6.3 Attribute Extensibility

Attribute names are considered to be set of type2 keywords. In order to be extended, the same rules as type2 keywords apply.

6.4 Attribute Syntax Extensibility

Attribute syntaxes are considered to be set of type2 enums. In order to be extended, the same rules as type2 enums apply.

7. Internationalization Considerations

This model describes attributes whose values can be text strings intended for human understanding rather than machine understanding. These attributes are:

Printer Attributes:

- printer-name
- printer-location
- printer-description
- printer-make-and-model
- printer-state-message
- printer-message-from-operator

Job Attributes

- job-name
- job-state-message
- job-message-from-operator

document-name

These attributes MUST contain characters for the coded character set defined in ISO 10646 [??] and be encoded using the UTF-8 character encoding rules [28]. No other coded character sets or encoding rules are allowed.

However, since these strings are intended for human understanding, the strings can be in any human language identifiable with tags defined in RFC 1766 [??]. A client that queries the above mentioned text attributes needs to know what human language is being used for these text attributes. A Printer optionally supports a "human-languages-supported" supported values attribute and a "human-language" default value attribute. Since these are optional, if the printer does not support these attributes, the client can assume that the text strings are either US English "en-US" or any other language that is specific to a given site. Whatever the value, the same value applies to ALL text based attributes for a given Printer and its contained Jobs.

In the HTTP mapping for IPP/1.0, these attributes are mapped in to the HTTP headers that control negotiation of human language between client and server.

8. 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.

9. 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.

[31] ?, " ", RFC 2130, ?.

[32] ?, " ", RFC 1514, ?.

[33] ?, " ", RFC 1903, ?.

[34] ?, " ", RFC 1808, ?.

10. 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.
901 San Antonio.Road, MPK-17
Palo Alto, CA 94303

Phone: 415-786-8995 (Area code change to 650 in August 1997)
Fax: 415-786-7077 (Area code change to 650 in August 1997)

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 - Dataproducts Corp.
Sylvan Butler, HP
Keith Carter, IBM Corporation
Jeff Copeland - QMS
Andy Davidson - Tektronix
Mabry Dozier - QMS
Lee Farrell - Canon Information Systems
Steve Gebert - IBM
Babek Jahromi, Microsoft
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
Paul Moore, Microsoft

Pat Nogay - IBM
Bob Pentecost - HP
Rob Rhoads - Intel

David Roach - Unisys
Stuart Rowley, Kyocera 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.

11. APPENDIX A: Terminology

This specification uses the terminology defined in this section.

11.1 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]. The sections below reiterate these definitions and include some additional ones.

11.1.1 MUST

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

11.1.2 MUST NOT

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

11.1.3 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.

11.1.4 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.

11.1.5 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.)

11.1.6 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.

11.2 Model Terminology

11.2.1 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 ("-"), period ("."), and underscore ("_"). A keyword starts with a lower-case letter.

11.2.2 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.

11.2.2.1 Attribute Name

Each attribute is uniquely identified in this document by its

attribute name. An attribute name 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 (").

11.2.2.2 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 3.

11.2.2.3 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.

11.2.2.4 Attribute Syntax

Each attribute is defined using an explicit syntax type. 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 rules for each syntax type. Attribute syntax types are defined in section 4.1.

11.2.3 Supports

By definition, an implementation (an instance of an IPP object) supports 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. However, even for MANDATORY attributes, 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 for

that attribute. This limited set of values represents the Printer's set of supported document formats. Supporting an attribute and some set of values for that attribute enables IPP end users to be aware of

and make use of those features associated with that attribute and those values. If an implementation chooses to not support an attribute or some specific value, then IPP end users would have no ability to make use of that feature within the context of IPP itself. However, due to existing practice and legacy systems which are not IPP aware, there might be some other mechanism outside the scope of IPP to control or request the "unsupported" feature (such as embedded instructions within the print data itself, for example).

For example, consider the "finishings-supported" attribute. If a Printer is not physically capable of stapling, the "finishings-supported" attribute MUST NOT be populated with the value of 'staple'. If a Printer is physically capable of stapling, an implementation MAY choose to support the "finishings-supported" attribute and the value of 'staple'. Doing so, would enable end users to be aware of and make use of the stapling feature. Without support for the value 'staple', an IPP end user would have no means within the protocol itself to request that a Job be stapled. However an existing print data formatter might be able to request that the document be stapled directly within the print data itself. In this case, the IPP implementation does not "support" stapling, however the end user is still able to have some control over the stapling of the completed job.

In order to ease the implementation burden, many IPP attributes are OPTIONAL. Therefore, it is an implementation choice whether or not to support OPTIONAL attributes and values, even if those attributes and values correspond to features and functions that are realizable in the implementation. However, it is RECOMMENDED that if an IPP implementation that is capable of realizing any feature or function that corresponds to an IPP attribute and some associated value, then that implementation support that IPP attribute and value.

Note: The set of values in any of the supported value attributes is set (populated) by some administrative process or automatic sensing mechanism that is outside the scope of IPP. For administrative policy and control reasons, an administrator may choose to make only a subset of possible values visible to the end user. In this case, the real output device behind the IPP Printer abstraction may be capable of a certain feature, however an administrator is specifying that access to that feature not flow back to the end user through the IPP protocol.

12. 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 status message that provides a short textual description of the status. The status code is intended for use by automata, and the status message is intended for the human end user. Since the status 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 status 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

Since IPP status codes are type2 enums, they 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.

12.1 Status Codes (type2 keyword)

Each status code is described below. Section 12.2 contains a table

that indicates which status codes apply to which operations.

12.1.1 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.

12.1.2 Successful Status Codes

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

12.1.2.1 successful-ok (0x00)

The request has succeeded.

12.1.2.2 successful-ok-ignored-or-substituted-attributes (0x01)

The request has succeeded, but since ignoring or substituting of values was requested ("ipp-attribute-fidelity" set to 'false' in the create request), some attributes were ignored or unsupported values were substituted with supported values in order to process the job without rejecting it.

12.1.3 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.

12.1.4 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.

12.1.4.1 client-error-bad-request (0x400)

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

12.1.4.2 client-error-forbidden (0x401)

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.

12.1.4.3 client-error-not-authenticated (0x402)

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".

12.1.4.4 client-error-not-authorized (0x403)

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".

12.1.4.5 client-error-not-possible (0x404)

This status code is used when the request is for something that can not 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.

12.1.4.6 client-error-timeout (0x405)

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.

12.1.4.7 client-error-not-found (0x406)

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.

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.

12.1.4.8 client-error-gone (0x407)

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.

12.1.4.9 client-error-request-entity-too-large (0x408)

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.

12.1.4.10 client-error-request-URI-too-long (0x409)

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.

12.1.4.11 client-error-unsupported-document-format (0x40A)

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

12.1.4.12 client-error-attribute-not-supported (0x40B)

For a Create-Job, Print-Job or Validate-Job operation, if the IPP Printer does not support one or more attributes or attribute values supplied in the request, the Printer shall return this status. For example, if the request indicates 'iso-a4' media, but that media type is not supported by the Printer. Or, if the client supplies an optional attribute and the attribute itself is not even supported by the Printer. If "ipp-attribute-fidelity" it set to false, the Printer can ignore or substitute values for unsupported attributes and values rather than reject the request and return this status code. .

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.

12.1.5 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.

12.1.5.1 server-error-internal- error (0x500)

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.

12.1.5.2 server-error-operation-not-supported (0x501)

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.

12.1.5.3 server-error-service-unavailable (0x502)

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.

12.1.5.4 server-error-version-not-supported (0x503)

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.

12.1.5.5 server-error-device-error (0x504)

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).

ISSUE: This doesn't seem like a good error. If the create request succeeded, an OK should be returned. The job-status of printer-stopped gives information about the printer being stopped.

12.1.5.6 server-error-temporary-error (0x505)

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.

12.2 Status Keywords for IPP Operations

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

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

13. APPENDIX C: "document-format" 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 symbols for each 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. MIME types corresponding to the integer enum value is used as the value the IPP "document-format" attribute.

This APPENDIX lists the document formats that are currently registered with IANA.

The standard values registered as of the data of this document 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.

ISSUE: What are we going to do with 'langAutomatic (37)'?.

14. 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

15. APPENDIX E: Processing IPP Attributes

When submitting a print job to an IPP Printer, the IPP model allows a client to supply operation and Job Template attributes along with the print data. These Job Template attributes in the create request 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, embedded within the print data itself. In addition, the Printer has a set of attributes that describe what rendering and finishing options which are supported by that Printer. This model, which allows for flexibility and power, also introduces the potential that at job submission time, these client-supplied attributes may conflict with either:

- what the implementation is capable of realizing (i.e., what the Printer supports), as well as
- the instructions embedded within the print data itself.

The following sections describe how these two types of conflicts are handled in the IPP model.

15.1 Fidelity

If there is a conflict between what the client requests and what a Printer supports, the client may request one of two possible conflict handling mechanisms:

1) either reject the job since the job can not be processed exactly

as specified, or

2) allow the Printer to make any changes necessary to proceed with processing the Job the best it can.

In the first case the client is indicating to the Printer: "Print the job exactly as specified with no exceptions, and if that can't be done, don't even bother printing the job at all." In the second case, the client is indicating to the Printer: "It is more important to make sure the job is printed rather than be processed exactly as specified; just make sure the job is printed even if client supplied attributes need to be changed or ignored."

The IPP model accounts for this situation by introducing an "ipp-attribute-fidelity" attribute.

In a create request, "ipp-attribute-fidelity" is a boolean attribute that is OPTIONALLY supplied by the client. The value 'true' indicates that total fidelity to client supplied attributes and values is required. The client is requesting that the Job be printed exactly as specified, and if that is not possible then the job must be rejected rather than processed incorrectly. The value 'false' indicates that a reasonable attempt to print the Job is acceptable. If a Printer does not support some of the client supplied Job Template attributes or values, the Printer may ignore them or substitute any supported value for unsupported values. The Printer may choose to substitute the default value associated with that attribute, or use some other supported value that is similar to the unsupported requested value. For example, if a client supplies a "media" value of 'na-letter', the Printer may choose to substitute 'iso-a4' rather than a default value of 'envelope'. Since this is an OPTIONAL attribute, if the client does not supply a value, the Printer assumes a value of 'false'.

Each Printer implementation MUST support both types of "fidelty" printing (that is whether the client supplies a value of 'true' or 'false'). This is possible across all types of implementations, since there is a broad range of acceptable actions when substituting or ignoring unsupported attributes and values. Also, even if the client supplies a value of 'false', a Printer might still reject the Job for any reason including an unsupported attributes and/or values. In the other case, where the client requests a value of 'true', it is expected that the Printer support this type of printing since the Printer is already indicating functional support corresponding to all advertised supported attributes and values.

Since a client can always query a Printer to find out exactly what is and is not supported, "ipp-attribute-fidelity" set to 'false' is useful when:

1) The End-User uses a command line interface to request attributes that might not be supported.

2) In a GUI context, if the End User expects the job might be moved to another printer and prefers a sub-optimal result to nothing at all.
3) The End User just wants something reasonable in lieu of nothing at all.

15.2 Page Description Language (PDL) Override

If there is a conflict between the value of an IPP Job Template attribute and a corresponding instruction in the print data, it is desirable that the value of the IPP attribute take precedence over the document instruction. Consider the case where a previously formatted file of print data is sent to an IPP Printer. In this case, if the client supplies any attributes at job submission time, the client desires that those attributes override the embedded instructions. Consider the case were a previously formatted document has embedded in it commands to load 'iso-a4' media. However, the document is passed to an end user that only has access to a printer with 'na-letter' media loaded. That end user most likely wants to submit that document to an that IPP Printer with the "media" Job Template attribute set to 'na-letter'. The job submission attribute should take precedence over the embedded PDL instruction. However, until companies that supply interpreters for print data allow a way for external IPP attributes to take precedence over embedded job production instructions, a Printer might not be able to support the semantics that IPP attributes override the embedded instructions.

The IPP model accounts for this situation by introducing a "pdl-override-supported" attribute.

This attribute takes on the following values:

- 'attempted': This value indicates that the Printer attempts to make sure that IPP attribute values take precedence over embedded instructions in the Print data, however there is no guarantee.
- 'not-attempted': This value indicates that the Printer makes not attempt to ensure that IPP attribute values take precedence over embedded instructions in the print data.

This is a MANDATORY Printer attribute.

At job processing time, an implementation that supports the value of 'attempted' might try to do one of several different actions:

1) generate an output device specific command sequence to realize
   the feature represented by the IPP attribute value

2) parse the print data itself and replace the conflicting embedded instruction with a new embedded instruction that matches the intent of the IPP attribute value
3) indicate to the Printer that external supplied attributes take precedence over embedded instructions and then pass the external IPP attribute values to the print data interpreter
4) anything else that allows for the semantics that IPP attributes override embedded print data instructions.

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

Note: The "ipp-attribute-fidelity" attribute applies to the Printer's ability to either accept or reject other unsupported attributes. In other words, if "ipp-attribute-fidelity" is set to 'true', a Job is accepted if and only if the client supplied attributes and values are supported by the Printer. Whether these attributes actually affect the processing of the Job depends on the ability of the Printer to override the instructions embedded in the print data with the semantics of the IPP attributes. If the print data attributes can be overridden ("pdl-override-supported" set to 'attempted'), the Printer makes an attempt to use the IPP attributes when processing the Job. If the print data attributes can not be overridden ("pdl-override-supported" set to 'not-attempted'), the Printer makes no attempt to use the IPP attributes when processing the Job, and hence, the IPP attributes may fail to affect the Job processing and output in any manner whatsoever.

15.3 Suggested Attribute Processing Algorithm

When a Printer receives a create request, the Printer either accepts or rejects the request. The Printer accepts the create request and creates a Job object if it is able to accept all Job Template and Document attributes in the request. The Printer rejects the request and does not create a Job object if the Printer rejects any Job Template or Document attribute in the request. In order to determine whether or not to accept or reject the request, the Printer SHOULD use the following algorithm:

1. The implementation checks to see if the operation is supported. If not, the Printer rejects the request and sets the appropriate

status code in the response.

2. The implementation checks to see if the requested version is supported. If not, the Printer rejects the request and sets the appropriate status code in the response.

3. The implementation checks to see if the client supplied an "ipp-attribute-fidelity" attribute. If the attribute is missing (not supplied by the client), the Printer assumes that the value is 'false'.

4. The Printer loops through all other attributes, checking to see if the requested values are supported (e.g., the value of "foo" in the request is one of the values in the Printer's "foo-supported" attribute). If the attribute or its value is unsupported, the Printer flags it as unsupported.

5. Once all attributes have been checked individually, the Printer checks for any inconsistent values among all the supported values. For example a Printer might be able to staple and to print on transparencies, however due to physical stapling limitations, the Printer might not be able to staple transparencies. Any inconsistent values are flagged as unsupported.

6. Once all attributes have been checked and validated, if "ipp-attribute-fidelity" is set to true and there are any attributes flagged as unsupported, the Printer rejects the request and returns all unsupported attributes and values in the response and sets the appropriate status code.

7. If "ipp-attribute-fidelity" is set to 'false' (nor it was not supplied by the client) and there are any attributes that are flagged as unsupported, the Printer, chooses to either ignore the unsupported attributes or change the requested value to some supported value. If, for some reason, it is not possible for the implementation to ignore or substitute values and is unable to "just print the job", the Printer is still able to reject the request and return all unsupported attributes and values in the response. In doing so, the Printer sets the appropriate status code.

8. If the Printer is able to accept the request (either as is or by making changes and the "ipp-attribute-fidelity" attribute is set to 'false'), the Printer creates a new Job object with the remaining valid Job Template attributes. Initially it sets the Job's state to 'pending'. These Job Template attributes are

associated with the Job object. All attributes which are associated with the Job object are intended to be override values that take precedence over whatever other embedded instructions

might be in the print data itself. That is, IPP allows for submission time attributes to take precedence over static instructions embedded in the print data. These submission time attributes are persistently stored with the Job. However, it is not possible for all implementations to realize the semantics of "override". End users may query the Printer's "pdl-override" attribute to determine if the Printer either attempts or does not attempt to override print data instructions with IPP attributes.

9. There are some cases, where a Printer supports a Job Template attribute and has an associated default value set for that attribute. In the case where a client does not supply the corresponding attribute, the Printer does not use its default values to populate Job attributes when creating the new Job object. The Printer's default values are used at Job processing time where no other IPP attribute or instruction embedded in the print data is present. Suppose the Printer were to associate the Printer's default value with the Job at creation time for an attribute not supplied by the client. This would change it from a default value to an override value. A later query of the Job object would return a set of attributes. Neither the Printer nor the end user would be able to tell the difference between an attribute that is an "override PDL" attribute supplied by the client or a "default value" attribute supplied by the printer.

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

11. Once the Job object has been created, the Printer responds back to the client with a successful response including Job status attributes that indicate the initial state of the Job ('pending', 'processing', etc.). The Printer uses its own configuration and implementation specific algorithms for scheduling the Job in the correct processing order. Once the Printer begins processing the Job, the Printer changes the Job's state to 'processing'. The Printer monitors all Jobs and notifies the intended recipients for each event by processing the all of the "notify-events" and "notify-addresses" Job attributes. If the Printer supports PDL override (the "pdl-override" attribute set to 'attempted'), the implementation does its best to see that IPP attributes take precedence over embedded instructions in the print data.

12. The implementation of the Printer object continues to process the Job until it can move the Job into the 'completed' state. If an Cancel-Job operation is received, the implementation eventually moves the Job into the 'cancelled' state. If the

system encounters errors during processing that do not allow it to progress the Job into a completed state, the implementation halts all processing, cleans up any resources, and moves the Job into the 'aborted' state.

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

Some Printer implementations may support "ipp-attribute-fidelity" set to 'true' and "pdl-override" set to 'attempted' and yet still not be able to realize exactly what the client specifies in the create request. This is due to legacy decisions and assumptions that have been made about the role of job instructions embedded within the print data and external job instructions that accompany the print data and how to handle conflicts between such instructions. The inability to be 100% precise about how a given implementation will behave is also compounded by the fact that the two special attributes, "ipp-attribute-fidelity" and "pdl-override", apply to the whole job rather than specific values for each attribute. For example, some implementations may be able to override almost all Job Template attributes except for "number-up". It would only make the IPP model more complex (with relatively little added value) to allow for additional special attributes that apply uniquely to each Job Template attribute or even specific values of each attribute.

16. APPENDIX F: Relationship to SNMP MIBs

The IPP model is related to the Printer MIB [??] in the following ways:

ISSUE: TBS.

The IPP model is related to the Job Monitoring MIB [??] in the following ways:

ISSUE: TBS