The Printer Working Group

August 21, 2015
Candidate Standard 5100.19-2015

IPP Implementor's Guide v2.0 (IG)
(Updates RFC 3196)

Status: Approved

Abstract: This document updates and extends "Internet Printing Protocol/1.1: Implementor's Guide" (RFC 3196) for all IPP protocol versions.

This document is a PWG Candidate Standard. For a definition of a "PWG Candidate Standard", see: http://ftp.pwg.org/pub/pwg/general/pwg-process30.pdf

This document is available electronically at:

http://ftp.pwg.org/pub/pwg/candidates/cs-ippig20-20150821-5100.19.docx
http://ftp.pwg.org/pub/pwg/candidates/cs-ippig20-20150821-5100.19.pdf

Title: IPP Implementor's Guide v2.0 (IG)

About the IEEE-ISTO

The IEEE-ISTO is a not-for-profit corporation offering industry groups an innovative and flexible operational forum and support services. The IEEE-ISTO provides a forum not only to develop standards, but also to facilitate activities that support the implementation and acceptance of standards in the marketplace. The organization is affiliated with the IEEE (http://www.ieee.org/) and the IEEE Standards Association (http://standards.ieee.org/).

For additional information regarding the IEEE-ISTO and its industry programs visit:

http://www.ieee-isto.org

About the IEEE-ISTO PWG

The Printer Working Group (or PWG) is a Program of the IEEE Industry Standards and Technology Organization (ISTO) with member organizations including printer manufacturers, print server developers, operating system providers, network operating systems providers, network connectivity vendors, and print management application developers. The group is chartered to make printers and the applications and operating systems supporting them work together better. All references to the PWG in this document implicitly mean “The Printer Working Group, a Program of the IEEE ISTO.” In order to meet this objective, the PWG will document the results of their work as open standards that define print related protocols, interfaces, procedures and conventions. Printer manufacturers and vendors of printer related software will benefit from the interoperability provided by voluntary conformance to these standards.

In general, a PWG standard is a specification that is stable, well understood, and is technically competent, has multiple, independent and interoperable implementations with substantial operational experience, and enjoys significant public support.

For additional information regarding the Printer Working Group visit:

http://www.pwg.org

Contact information:

The Printer Working Group
c/o The IEEE Industry Standards and Technology Organization
445 Hoes Lane
Piscataway, NJ 08854
USA

About the Internet Printing Protocol Work Group

The Internet Printing Protocol (IPP) working group has developed a modern, full-featured network printing protocol, which is now the industry standard. IPP allows a print client to query a printer for its supported capabilities, features, and parameters to allow the selection of an appropriate printer for each print job. IPP also provides Job information prior to, during, and at the end of Job processing.

For additional information regarding IPP visit:

http://www.pwg.org/ipp/

Implementers of this specification are encouraged to join the IPP mailing list in order to participate in any discussions of the specification. Suggested additions, changes, or clarification to this specification, should be sent to the IPP mailing list for consideration.

Table of Contents

1. Introduction

IPP defines a rich set of operations and attributes to support the many tasks and features that exist in a networked printing ecosystem. With IPP, it is possible to perform some tasks in a few different ways. The quality of the user experience will be affected by how the tasks are performed. The goal of this specification is to provide IPP Client and Printer implementors with guidance on how to implement their system well from the start to ensure a quality user experience.

2. Terminology

2.1 Conformance Terminology

Capitalized terms, such as MUST, MUST NOT, RECOMMENDED, REQUIRED, SHOULD, SHOULD NOT, MAY, and OPTIONAL, have special meaning relating to conformance as defined in Key words for use in RFCs to Indicate Requirement Levels [RFC2119]. The following additional terms are defined:

CONDITIONALLY REQUIRED: A conformance requirement that applies when a specified condition is true.

DEPRECATED: An operation, attribute, or value that SHOULD NOT be used or supported in new implementations.

This specification defines a number of normative requirements, but they are all recommendations using SHOULD. There are no MUST requirements in this specification.

2.2 Imaging Terminology

Normative definitions and semantics of printing terms are imported from IETF Printer MIB v2 [RFC3805], IETF Finisher MIB [RFC3806], and IETF Internet Printing Protocol/1.1: Model and Semantics [RFC2911].

This document also defines the following protocol roles in order to specify unambiguous conformance requirements:

Client: Initiator of outgoing IPP session requests and sender of outgoing IPP operation requests (Hypertext Transfer Protocol -- HTTP/1.1 [RFC7230] User Agent).

Device: A Logical or Physical Device associated with one or more Printers; also see section 2.3 of [RFC2911].

Document: An object created and managed by a Printer that contains the description, processing, and status information. A Document object can have attached data and is bound to a single Job.

Logical Device: a print server, software service, or gateway that processes jobs and either forwards or stores the processed Job or uses one or more Physical Devices to render output.

Physical Device: a device that renders output (typically on paper.)

Printer: Listener for incoming IPP session requests and receiver of incoming IPP operation requests (Hypertext Transfer Protocol -- HTTP/1.1 [RFC7230] Server) that represents one or more Physical Devices or a Logical Device.

Imaging Device: A printer or other device that acts as a Printer.

Job: An object created and managed by a Printer that contains description, processing, and status information. The Job also contains zero or more Document objects.

2.3 Other Terminology

Direct Imaging: Printing, facsimile, and scanning performed by direct communication from the Client to an Imaging Device or local print server.

Directory Service: A Service providing query and enumeration of information using names or other identifiers.

Discovery: Finding Printers by querying or browsing local network segments or Enumeration of Directory or Name Services.

Report an Error: An Action taken by an IPP Client that includes informing the User that an error has been detected, and logging the condition in a robust manner.

Enumeration: Listing Printers that are registered with a Directory or other Service.

Indirect Imaging: Printing, facsimile, and scanning performed by communication from the Client and/or Imaging Device to an intermediary service in a different administrative domain, for example when the Client communicates with a third-party print service or when an Imaging Device communicates with a Cloud service.

Network Accessible Device: A Device that can be directly accessed by a Client.

Network Accessible/Accessibility: Refers to the ability of one device to communicate directly with another, for example a Client is able to connect to a Device, query for supported attributes, submit Job creation requests, and so forth.

Operator: A person or automata that typically oversees the Printer. The Operator is allowed to query and manage the Printer, Jobs and Documents based on site policy.

Secure Print: A print Job using the "document-password", "job-password", and/or "job-password-encryption" operation attributes to provide document and/or physical security. See [PWG5100.11], [PWG5100.13], and [PWG5100.16].

Service: Software providing access to physical, logical, or virtual resources assisting in the processing of queued Jobs.

User: A person or automata using a Client to communicate with a Printer.

Zombie Job: A Job that was created using a Job Creation operation but that never had a Document associated with it, and that remains in a state of limbo.

2.4 Acronyms and Organizations

IANA: Internet Assigned Numbers Authority, http://www.iana.org/

IEEE: Institute of Electrical and Electronics Engineers, http://www.ieee.org/

IETF: Internet Engineering Task Force, http://www.ietf.org/

ISO: International Organization for Standardization, http://www.iso.org/

PWG: Printer Working Group, http://www.pwg.org/

3. Requirements

3.1 Rationale

The "Internet Printing Protocol/1.1: Implementor's Guide" [RFC3196] was ratified in November 2001. Since that time many extensions to IPP have been ratified, and the scope of use of IPP has grown considerably. Given all these extensions to IPP, implementers will benefit from an updated best practices document that covers these extensions, as well as the core of IPP that has remained unchanged, to assist Client and Printer implementers in their efforts to provide the best possible user experience.

3.2 Use Cases

3.2.1 Developer Implementing IPP Client Support

Garrett is a software engineer developing a new client platform that is adding system-level printing support. Many printers support IPP Everywhere [PWG5100.14], so he plans to implement IPP Everywhere printing support in his client platform. But IPP Everywhere and its related standards don't describe how to best use IPP for the various tasks his software needs to perform to deliver a quality client user experience. He finds the "Internet Printing Protocol/1.1: Implementor's Guide" [RFC3196], but finds its Client recommendations insufficient. Using the IPP Implementor's Guide v2.0 (IG), he is able to avoid some common design pitfalls and quickly deliver a quality IPP client experience.

3.2.2 Developer Implementing IPP Printer Support

Duncan is a firmware engineer at a printer vendor creating a new printer, and that printer includes support for IPP Everywhere. In reading the IPP Implementor's Guide v2.0 (IG), he can more accurately anticipate how some segment of clients implemented according to these practices are likely to behave, and more rapidly understand how the various operations can be used with one another to achieve certain tasks.

3.3 Out of Scope

The following are out of scope for this specification:

- Definition of extensions or modifications to the Internet Printing Protocol
- Normative MUST statements regarding user experience
- Definitions of new file formats or modifications of existing file formats
- Definition of extensions or modifications to HyperText Transport Protocol [RFC7230]

3.4 Design Requirements

The design requirements for this specification are:

- Identify common tasks performed by Clients and Printers
- Enumerate a series of implementation alternatives
- Rank those alternatives according to a qualitative grading scheme
- Discuss proper use of IPP attributes related to the implementation alternatives and related implementation considerations, including security considerations

4. Client Tasks and Implementation Alternatives

A user needs to do certain tasks in order to engage the user's computer with a printer. If that printer is an IPP Printer and the user's computer has an IPP Client, the IPP Client needs to perform certain tasks to provide a basic level of service to the user. A well-implemented Client also performs additional tasks to provide a high quality user experience. A fully featured and well-implemented IPP Client will support all the following tasks:

- Find A Printer
- Validate Printer Reachability and User Access
- Identify a Printer
- Get Printer Capabilities
- Check Constraints Between Print Options
- Submit a Print Job
- Monitor Job Status
- Cancel a Print Job During Job or Document Creation
- Get Printer Supplies Status
- Error Handling

Each task is covered in this section, including descriptions of implementation alternatives. Each alternative will be labeled according to its perceived quality, using the following labels, which are listed in order from least desirable to most desirable:

- BAD
- POOR
- FAIR
- GOOD
- BETTER
- BEST

The most preferred alternative for each task will always be labeled "BEST". Options labeled as "BAD", "POOR" or "FAIR" SHOULD be avoided.

The task implementations are described using UML sequence diagrams, which provide methods for describing parallel activity, asynchronous activity, iteration, logical branching, and similar constructs.

4.1 Find A Printer

In order to use a Printer, the Client system needs to connect to it. For a connection to be established, the Client needs to first acquire the Printer's address.

Historically, setting up a printer on a client system included creating persistent local records for that printer to identify drivers needed to communicate at different levels. The number of printers or other devices the user's system would encounter was small and fairly static. Network discovery protocols were not broadly used.

Recently, the relationship between Client and Printer has become more "dynamic". Clients discover available standardized print services matched to "universal" drivers rather than model-specific ones. These universal drivers acquire that print service instance's capabilities and other attributes by interrogating the chosen print service using sophisticated protocols such as IPP. Based on that, the universal driver provides a set of features based on device information rather than pre-distributed model information (model-specific drivers). This set of capabilities can be acquired at each time the User wishes to print, rather than once when the "print queue" has been created on the Client. In this paradigm, a persistent binding between a driver and a print service (what has historically been referred to as a "print queue") is more like a Web browser bookmark, and need not be persisted except as a "favorite" or "recently used printer" as a convenience to the User.

Even with the recent driver / printer interface standardization and widespread implementation, there are various use cases where the user acquires the Printer's DNS host name or network address and wishes to contact the Printer directly rather than looking it up via a service discovery or directory protocol. Examples of this include: the target printer does not reside within the scope that the discovery protocol service, or when the user has acquired a URI, host name or network address from a banner near the printer. These use cases will continue to be important for the foreseeable future.

4.1.1 Discover and Select a Printer Using a Discovery Protocol

Discovery protocols such as mDNS / DNS-SD [RFC6762] [RFC6763], SLP [RFC2608], and WS-Discovery [WSDiscovery-1.1], as well as directory protocols such as LDAP [RFC4510], allow a user to find instances of print services or printers without having to perform a survey in physical space to acquire devices' addresses. Service discovery queries are sent via broadcast or multicast to all hosts within scope, or unicast to a directory server. The query specifies the desired service types or device types. Replies to those queries are processed and presented to the user.

Regardless of the actual discovery protocol used, the APIs driving the protocols generally can be used in either a synchronous or asynchronous fashion. Unfortunately, many legacy software systems (as well as developers) are accustomed to the synchronous model, which is easily identified by the presence of a "refresh button". The synchronous model is not as user friendly as the asynchronous model, but it is somewhat easier to write programs in a synchronous way than an asynchronous way.

Alternatives

- POOR:
  - Perform network discovery with a synchronous API; present static results after short discovery process period; provide "Refresh" button to restart the process

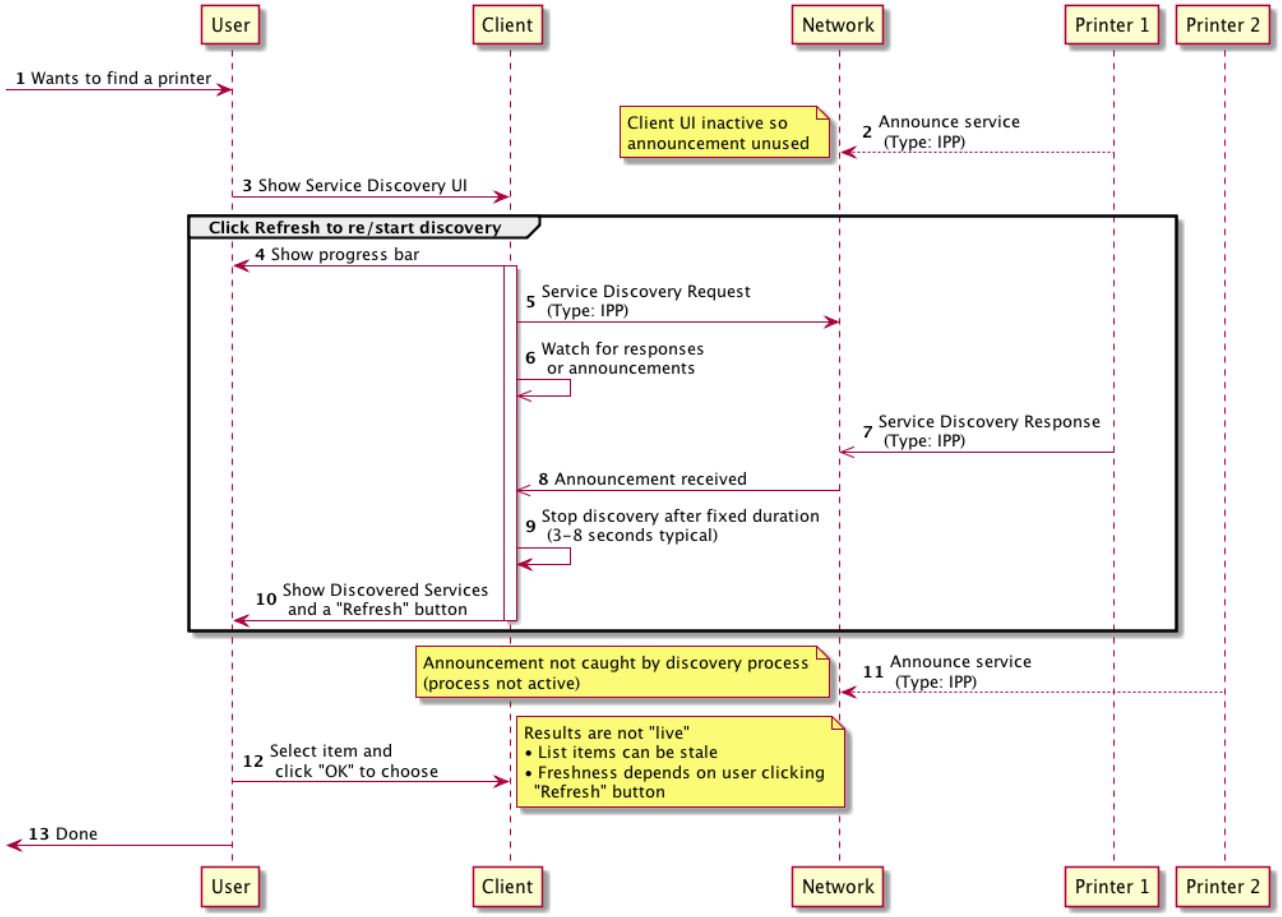

Figure 1: Discover and Select a Printer Using a Discovery Protocol - POOR

- BETTER:
  - Same as "POOR" above but user interface self-refreshes every few seconds in a worker thread that refreshes the UI owned by the main thread (or equivalent).

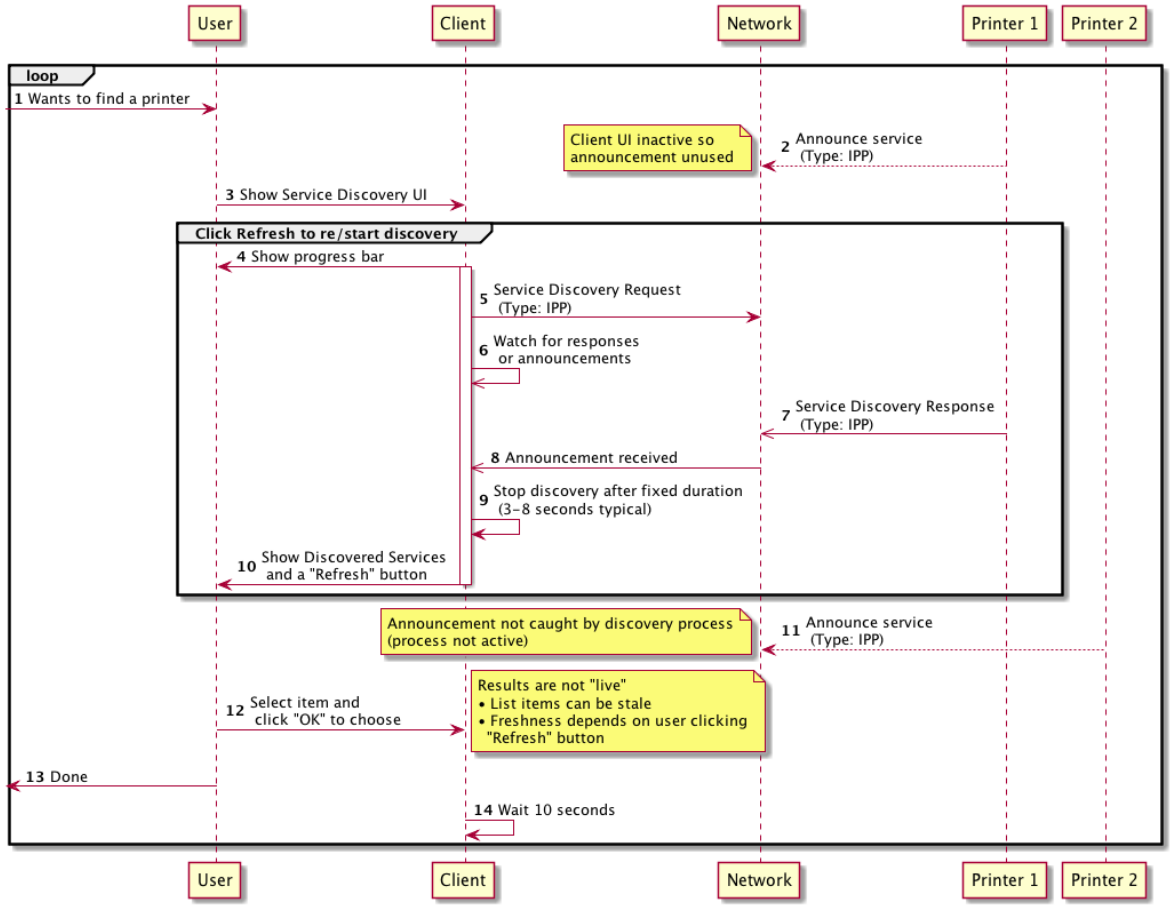

Figure 2: Discover and Select a Printer Using a Discovery Protocol - BETTER

- BEST:
  - Perform network discovery with an asynchronous API for "live" results and timely, responsive user experience

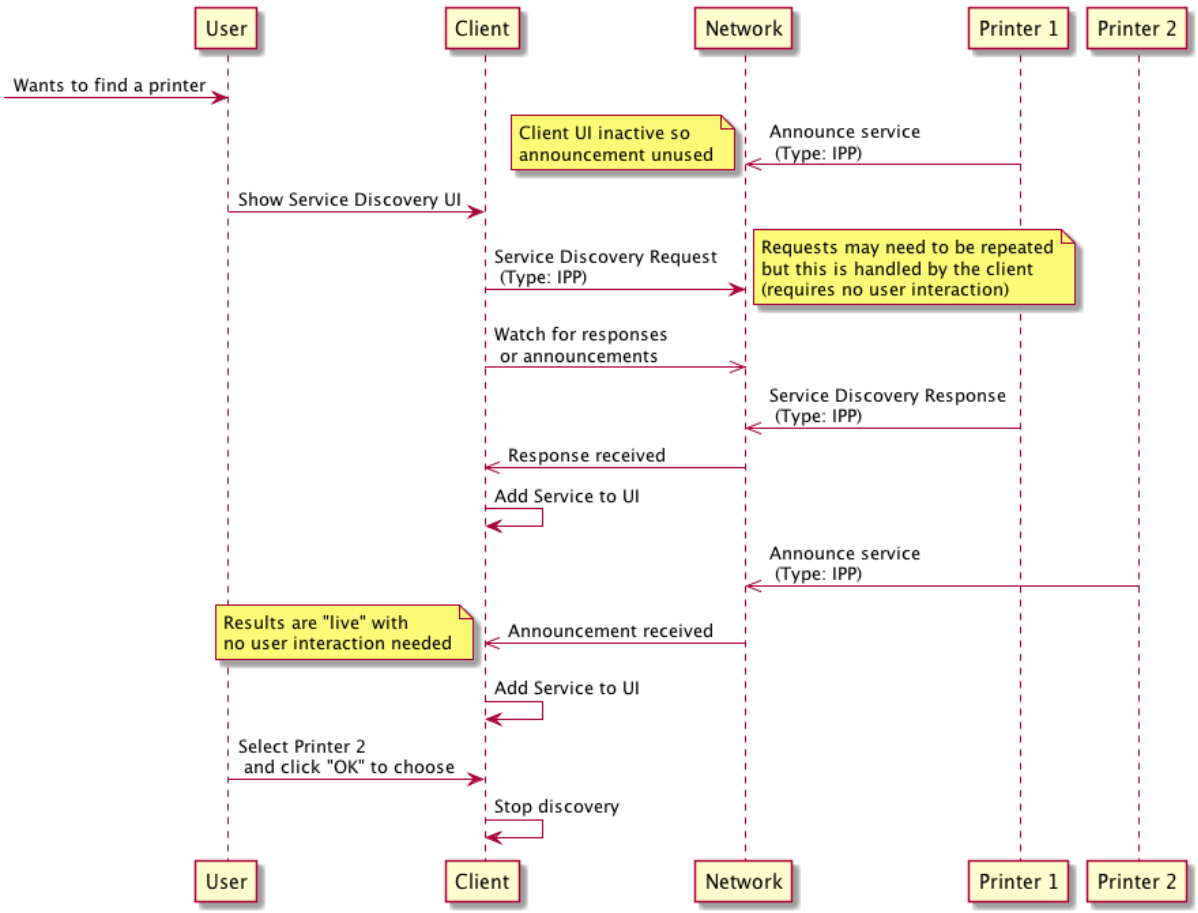

Figure 3: Discover and Select a Printer Using a Discovery Protocol - BEST

4.1.2 Select A Printer Using a Static Hostname or Address

A Printer is identified in IPP by a URI. The URI identifies a specific resource that is relevant only on the host where the Printer resides. A URI is not typically provided directly to a User; a simple host name or network address is more common.

The "printer-uri-supported" Printer attribute enumerates the allowed URIs on the host system. This attribute can be retrieved via a Get-Printer-Attributes operation. This can be a circular problem since the Client needs to have at least one of the allowed resource paths to perform a Get-Printer-Attributes operation. If all Printers implement well-known resource paths, this problem can be avoided.

Alternatives

- POOR: Each printer model chooses an arbitrary default resource path. The Client needs to depend on some method or protocol other than IPP to get the resource path. This exposes too many protocol details to the User. Users need not be aware of which print protocol is being used.

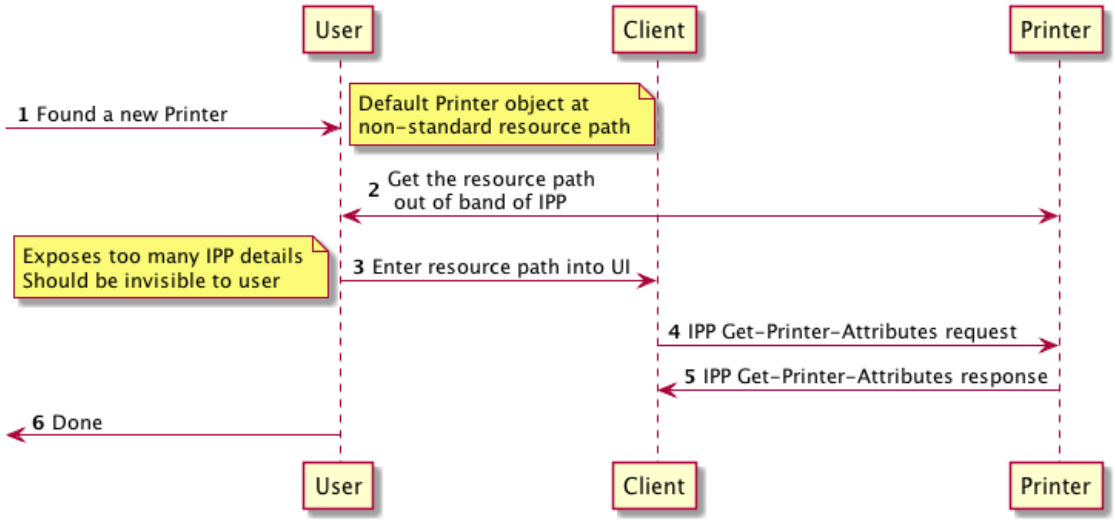

Figure 4: Select A Printer Using a Static Hostname or Address - POOR

- BETTER: Use a well-known, commonly specified resource path: "/ipp/print"
  - Not universally implemented
  - Recommendation to be added to IPP Everywhere [PWG5100.14]

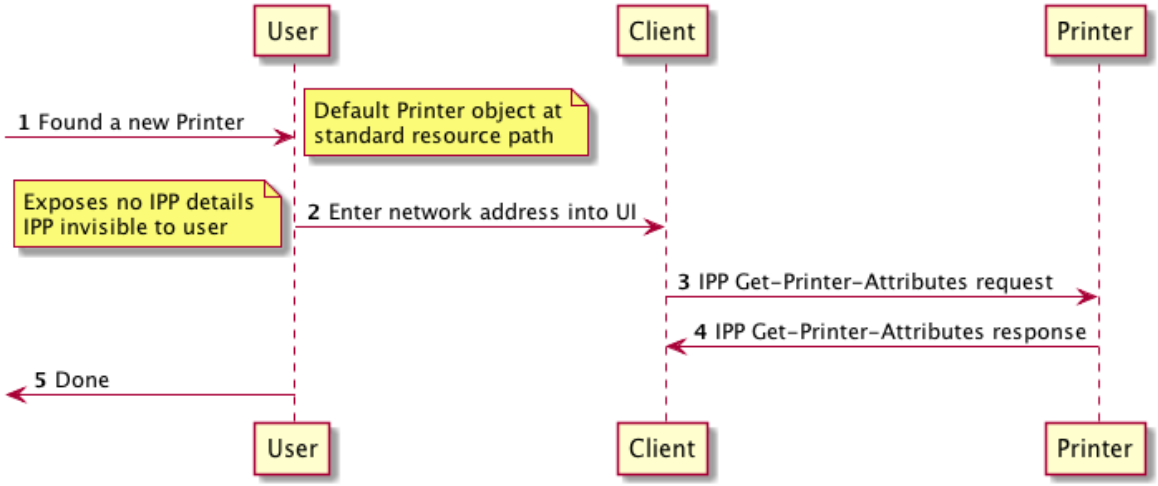

Figure 5: Select A Printer Via Static Hostname Or Address - BETTER

- BEST: Get a list of all URIs for the default Print service, from a common neutral resource path: "/"
    - Allows variance and different Printer Objects to be found
    - Not widely implemented, but recommended in IPP Everywhere [PWG5100.14]

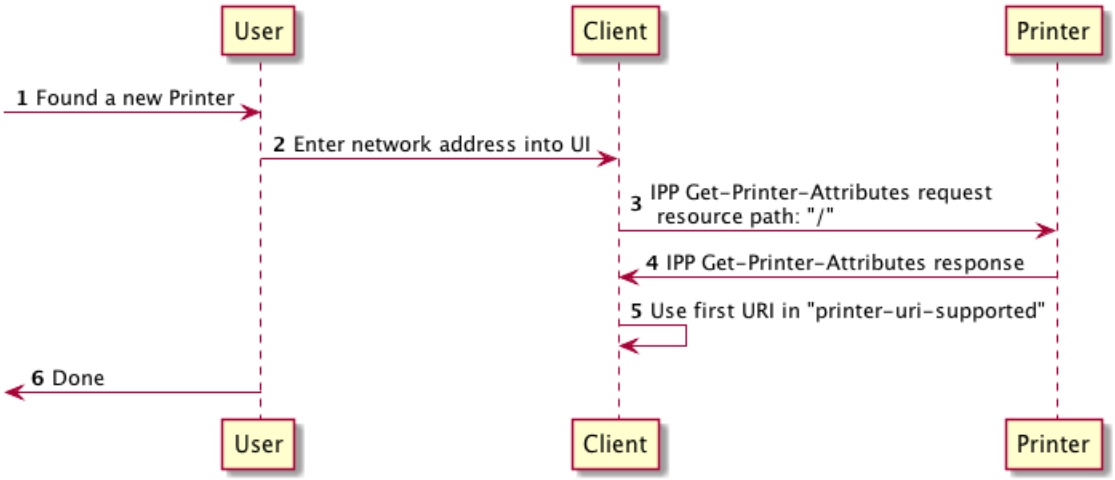

Figure 6: : Select A Printer Using a Static Hostname or Address - BEST

4.2 Validate Printer Reachability and User Access

Once a Printer's address has been acquired, the Client ought to communicate with the Printer to confirm that the Printer is reachable by the Client. The Client SHOULD validate that the Printer is reachable via one of the IPP transports supported by the Client system. Once reachability has been validated, the Client SHOULD validate that the Printer would allow the User operating the Client to use it. The Client SHOULD validate reachability and user access as a precondition to acquiring the Printer's capabilities, which can occur before submitting a Job or before creating a persistent queue or possibly both. If the User is allowed to set up the Client to use a particular Printer, but denied the use of that Printer, the User will most likely be disappointed. If the Printer implements IPPS [RFC7472] and a self-signed TLS certificate, the Client user experience ought to be managed in such a way that it allows the user to establish trust with unfamiliar devices or credentials, without scaring them away or concealing important details.

Alternatives

- BAD: Do Nothing
  - Reachability and access not checked; user could easily be disappointed

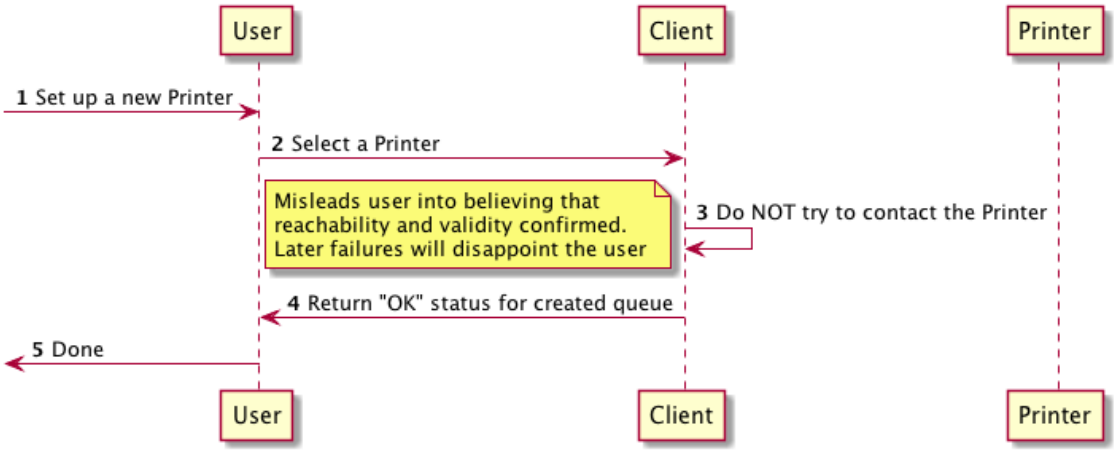

Figure 7: Validate Printer Reachability and User Access - BAD

- FAIR: Validate reachability using a non-IPP protocol (ICMP echo, SNMP GET, etc.)
  - Validates reachability
  - No validation that IPP service is available on target host
  - No validation that the user has access rights to the IPP Printer Object

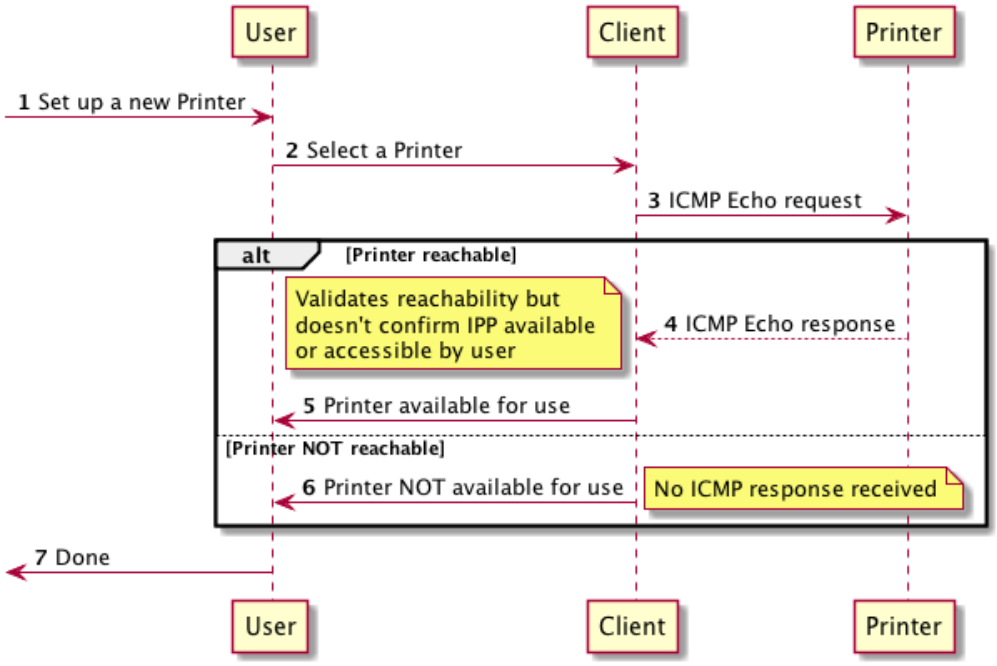

Figure 8: : Validate Printer Reachability and User Access - FAIR

- BETTER: IPP Get-Printer-Attributes
  - Validates reachability
  - Validates IPP Printer object is available on target host
  - No validation that the user has access rights to the IPP Printer Object

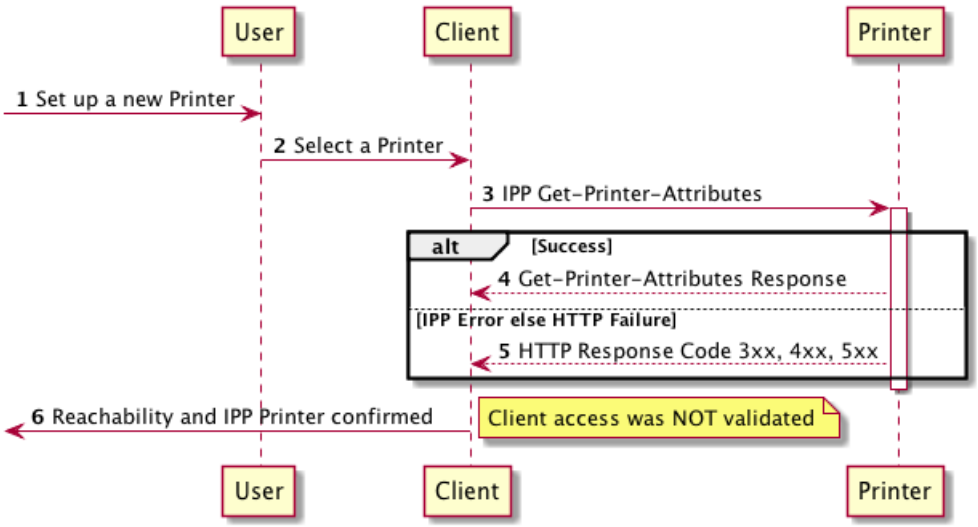

Figure 9: Validate Printer Reachability and User Access - BETTER

- BEST: IPP Get-Printer-Attributes + IPP Validate-Job
  - Validates reachability
  - Validates IPP service is available on target host
  - Validates user has access rights to the IPP Printer Object

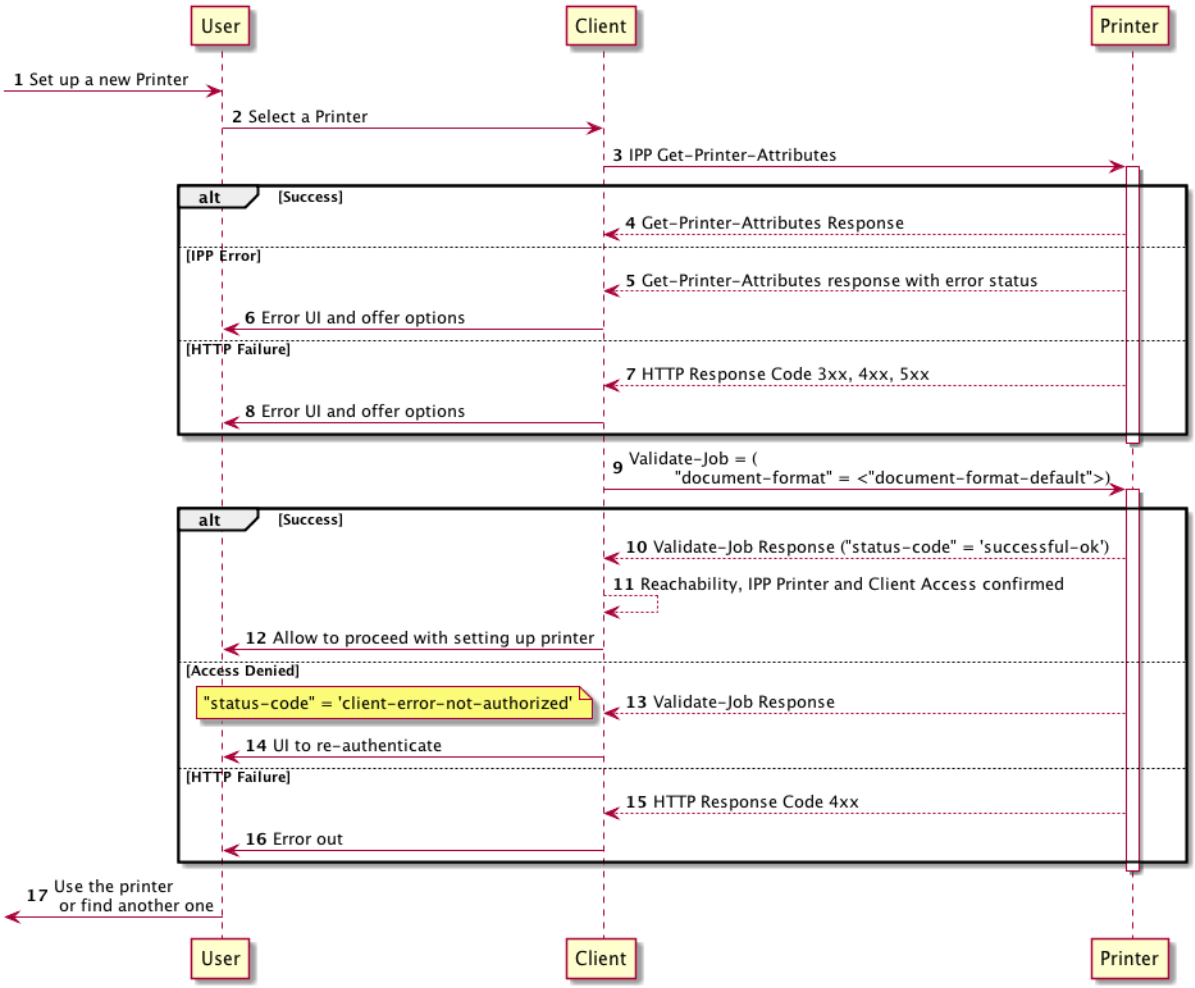

Figure 10: Validate Printer Reachability and User Access - BEST

4.3 Identify a Printer

There are situations where a Client has discovered a Printer on a network, and the User would like to determine its location in physical space, so they can find where to retrieve their printed job. Or the user might want to confirm that the printer they are standing in front of is the printer found on the network. This is increasingly important for users who are printing from highly portable devices such as smartphones or tablets.

The Identify-Printer operation [PWG5100.13] provides a method for a Client to request that the Printer do something to identify itself. It is preferable for the Client to

Alternatives

- BAD: Client does nothing; User guesses which printer
  - Subpar user experience

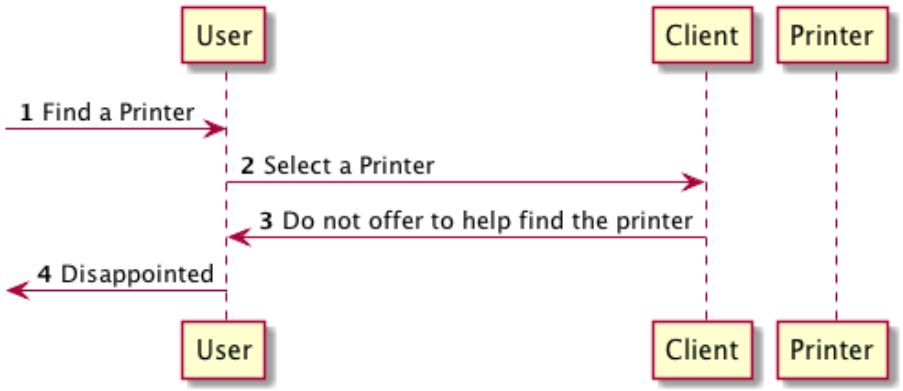

Figure 11: Identify a Printer - BAD

- POOR: IPP Print-Job with special "identify page"
  - Only useful to validate the printer nearby is the printer discovered
  - If it is the "wrong printer" the page will come out of an unexpected printer
  - Printing throwaway pages for this purpose can annoy the Printer's owner

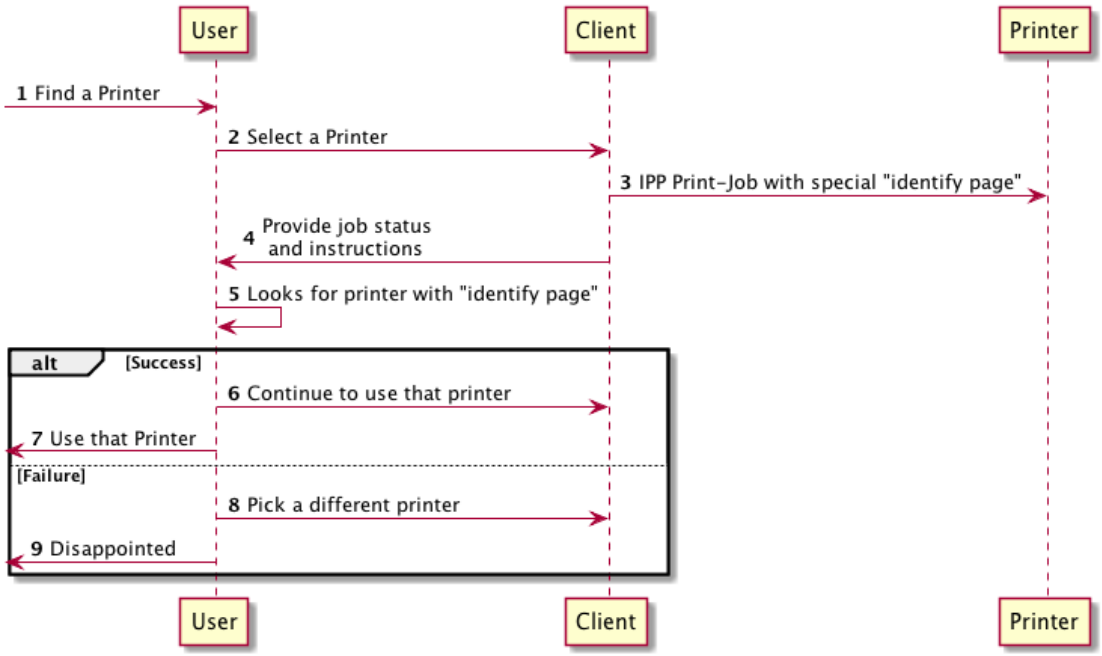

Figure 12: Identify a Printer - POOR

- BEST: IPP Identify-Printer
  - Add audio indicators only if the user cannot find it using visual means alone, to avoid causing audio disturbance to others nearby the Printer

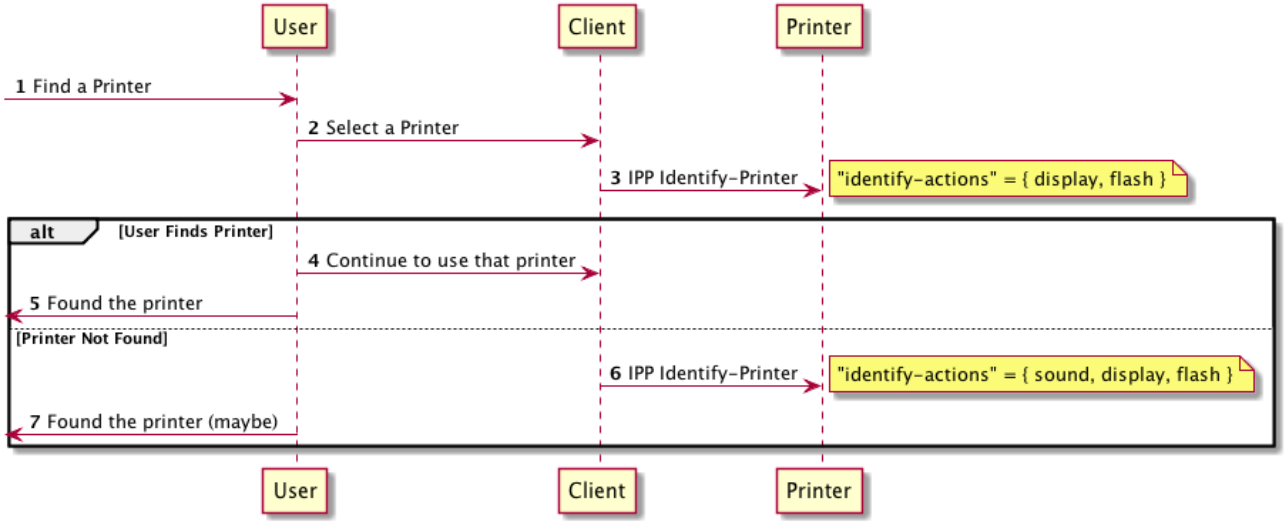

Figure 13: Identify a Printer - BEST

4.4 Get Printer Capabilities

Once the user has selected a printer, the Client needs to determine the Printer's capabilities in order to whether and how it can produce Jobs the Printer can process. The Printer's capabilities will include supported Document formats, Job processing alternatives such as "number of copies", "2-sided printing", and perhaps other features. Some capabilities can vary depending on the Document format. It also includes other information about the device itself, such as its model type, location, and so forth.

Traditionally, the Printer's capabilities would be acquired from the drivers bound to that Printer's print queue. A Client binds drivers by matching model identifiers in the driver with model identifiers acquired from the Printer. The Printer's capabilities were assumed to be relatively static; for the Client to get an up-to-date set of the Printer's capabilities the print queue would have to be destroyed and re-created. If driver matching fails because a driver was missing, the Client could not use the Printer because the queue would not get created. Deploying drivers before the user needed them was essential to avoid this problem.

Recently, "universal" printing systems such as IPP Everywhere have emerged that have enabled the Client / Printer relationship to become more dynamic. These systems define a base set of functionality along with methods to describe and use additional features that are agnostic to a particular Printer model. Clients that support these systems no longer depend on model-specific drivers to determine the Printer's capabilities if the Printer also supports the system. This also enables the Client to acquire a Printer's capabilities at the time the User indicates a desire to print. The Client can interrogate the Printer to

determine its capabilities and present the available printing options in a just-in-time fashion. This enables more dynamic connection workflows that were impractical with the traditional model.

Alternatives

- BAD: Assume all Printers have the same capabilities
  - Example: a print queue using a "generic PPD"

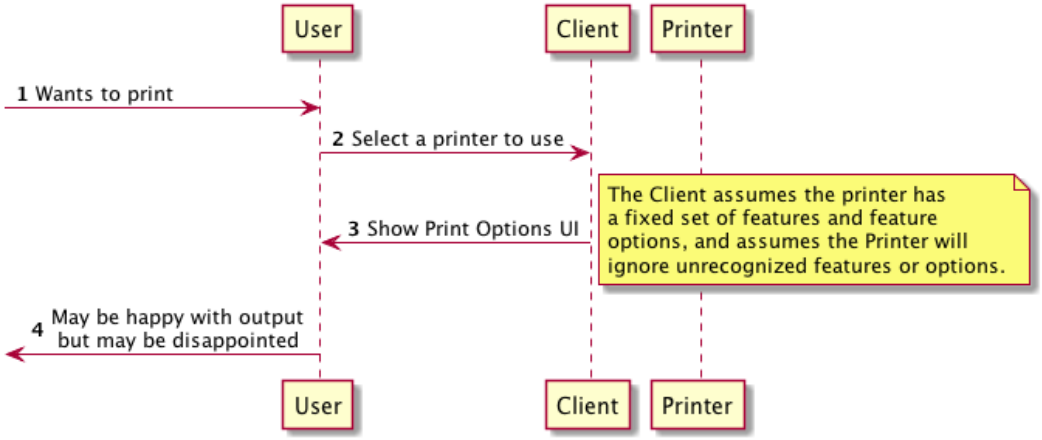

Figure 14: Get Printer Capabilities - BAD

- POOR: Match model to matching driver (e.g. model-specific PPD file) to acquire list of options supported by the target device; match using non-IPP protocol(s)
    - All "legacy style" model specific print drivers are an example of this alternative.
    - Printer capability detection using feature identifiers is preferable to detection keying off a model identifier, especially in the case where the source is a statically authored knowledge such as a PPD

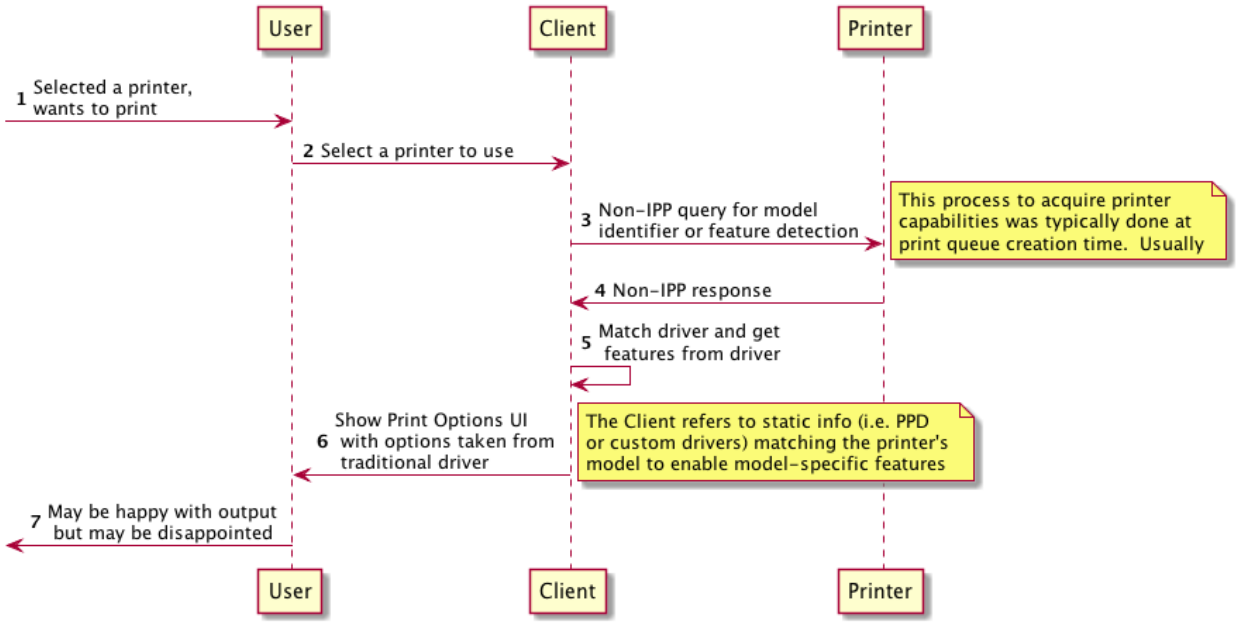

Figure 15: Get Printer Capabilities - POOR

- FAIR: Match model to matching driver (e.g. model-specific PPD file) to acquire list of options supported by the target device; match using IPP operations
    - Using IPP rather than another protocol means only one protocol is used for Printer capability evaluation and Job submission, even with traditional driver use models

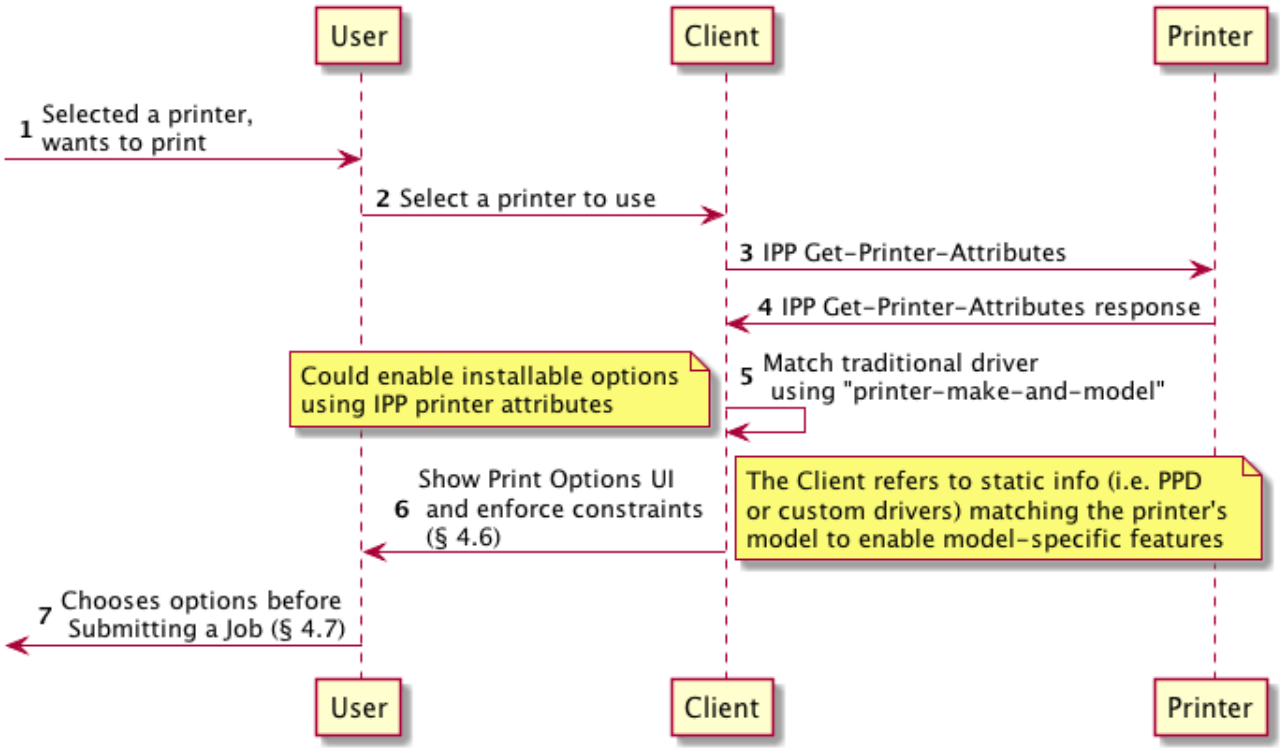

Figure 16: Get Printer Capabilities - FAIR

- BEST: Perform a Get-Printer-Attributes operation, sending "document-format" and other attributes that allow filtering. If needed, start with an initial Get-Printer-Attributes with no filtering attributes, or acquire Printer information via some other method, to get the set of filterable attributes from the Printer

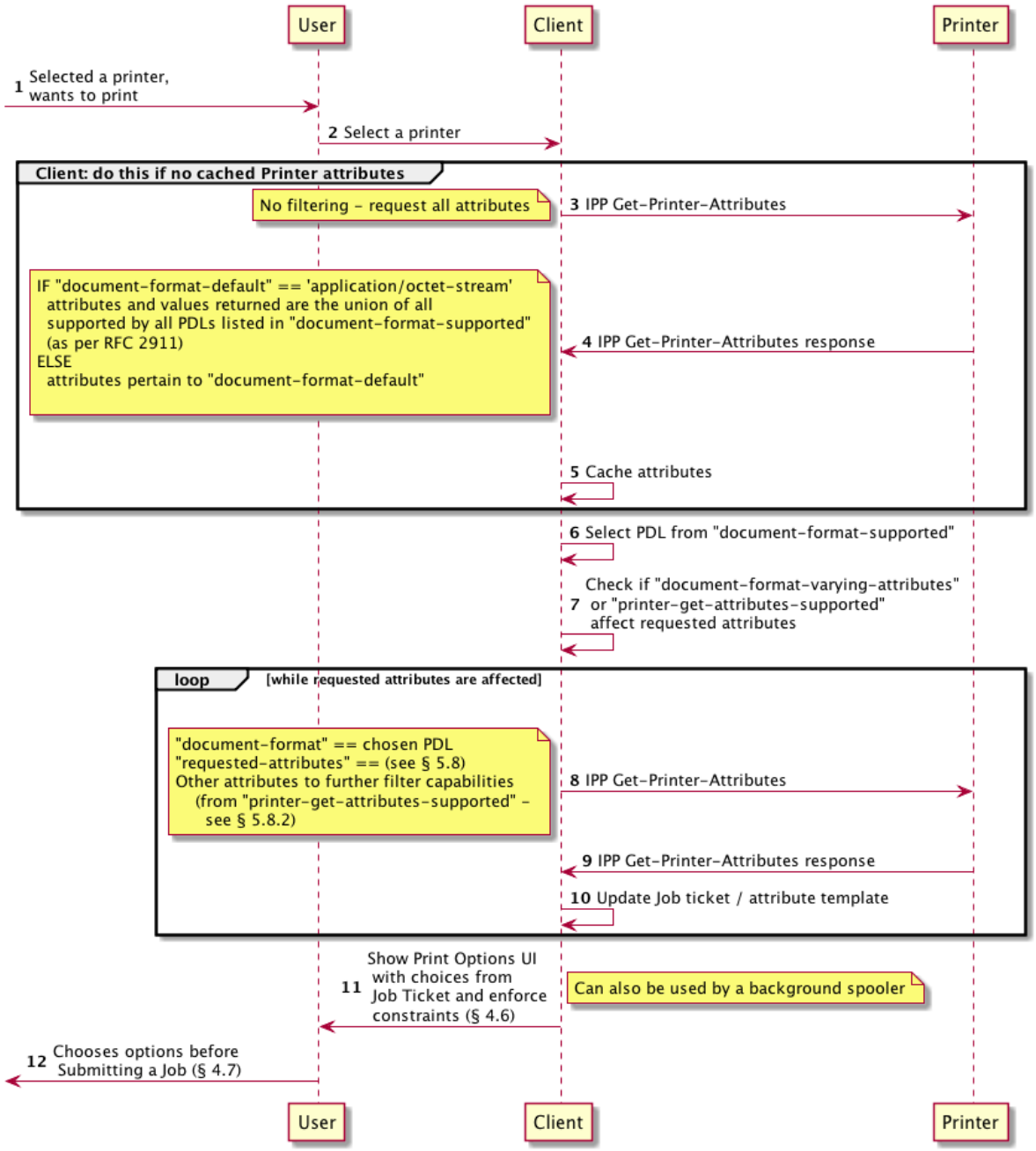

Figure 17: Get Printer Capabilities - BEST

4.5 Check Constraints Between Print Options

Printer features and options are presented typically in a print dialog. Some of these have states that have relationships with other options' states, where one cannot be in a particular state if another one is too. These are known as "constraints". Constraints need to be re-evaluated any time a control changes state. There are a few different ways this can be done via IPP.

Alternatives

- BAD: Do nothing

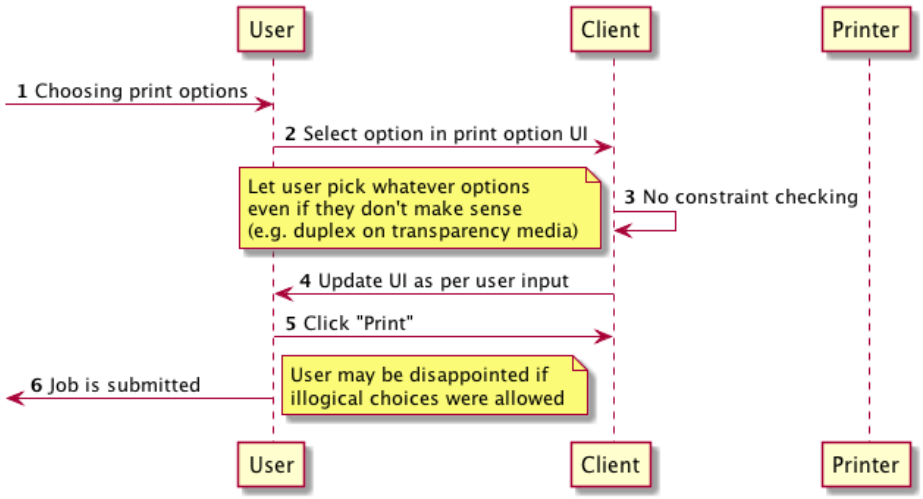

Figure 18: Check Constraints Between Print Options - BAD

- POOR: Perform IPP Validate-Job every time a control changes
  - Uses IPP but badly thrashes the network

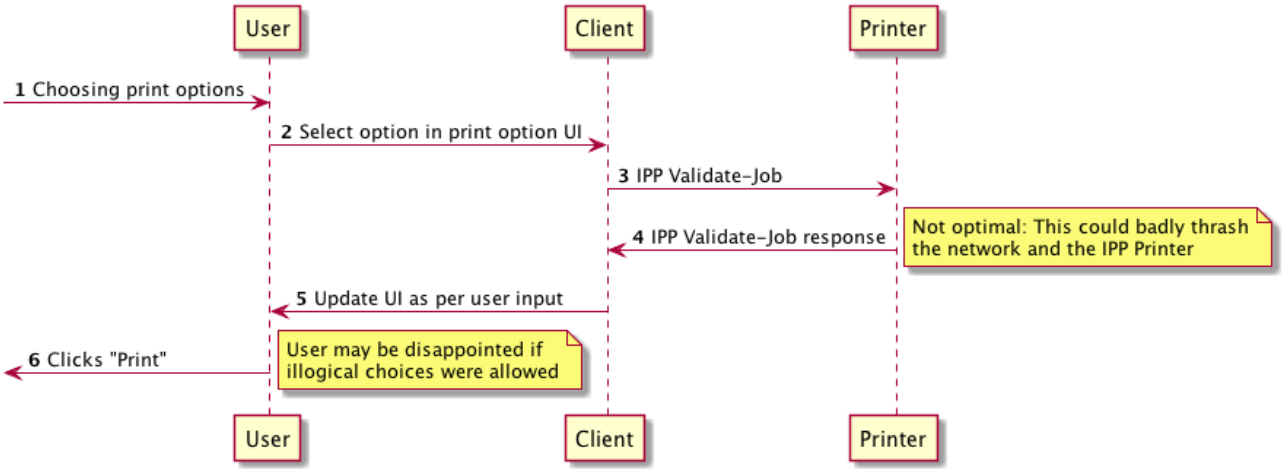

Figure 19: Check Constraints Between Print Options - POOR

- FAIR: Use local model-specific data (i.e. PPDs)
  - This is preferable because at least it doesn't thrash the network

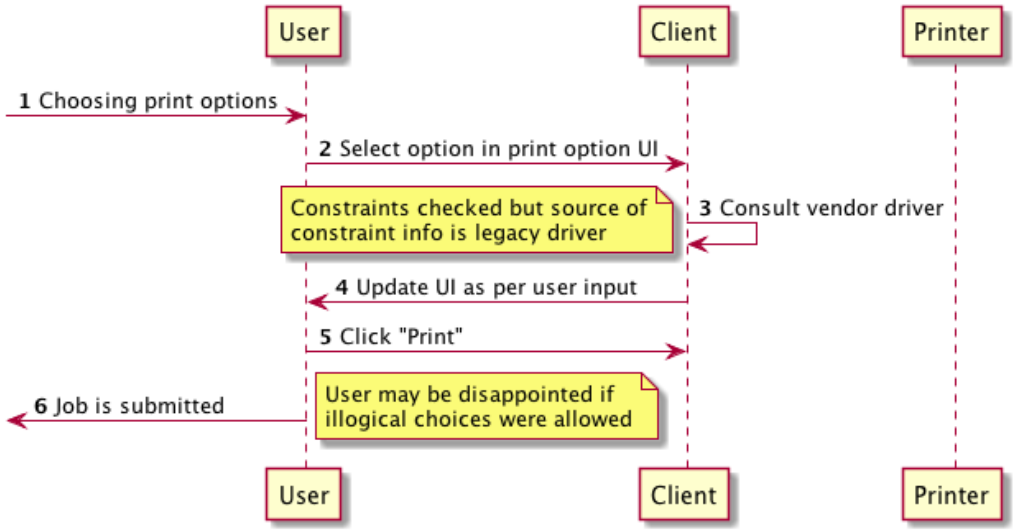

Figure 20: Check Constraints Between Print Options - FAIR

- GOOD:
  - IPP Validate-Job
    - When "Print" button is pressed, confirms the Job creation / submission will succeed (authentication, etc.)
    - Client depends on this operation to perform constraints validation printer-side, checking for errors along with "preferred-attributes"

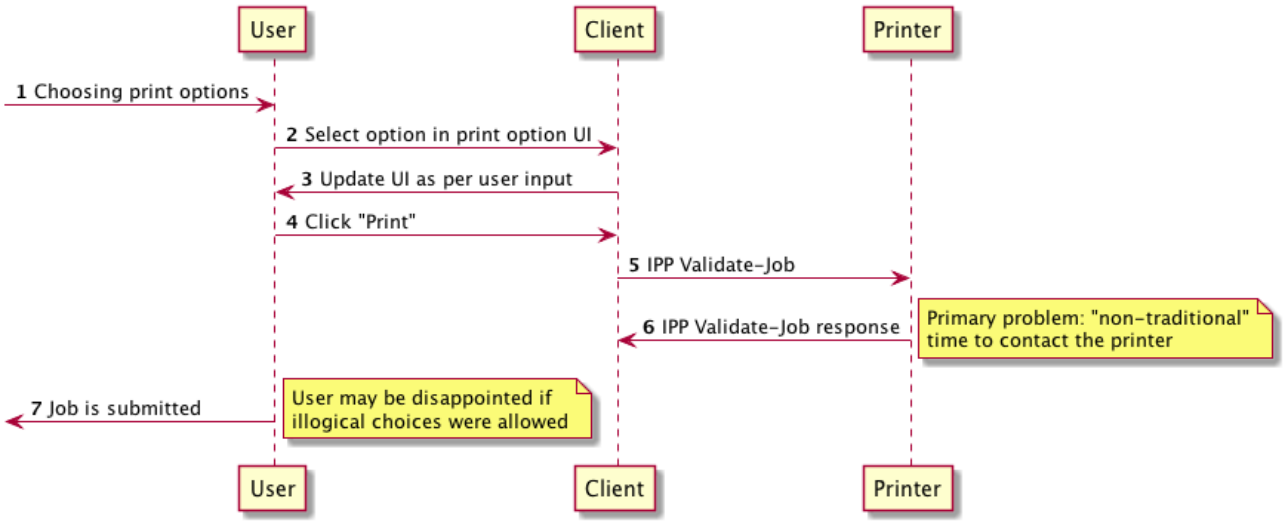

Figure 21: Check Constraints Between Print Options - GOOD

- BETTER: Use "job-constraints-supported" and "job-resolvers-supported"

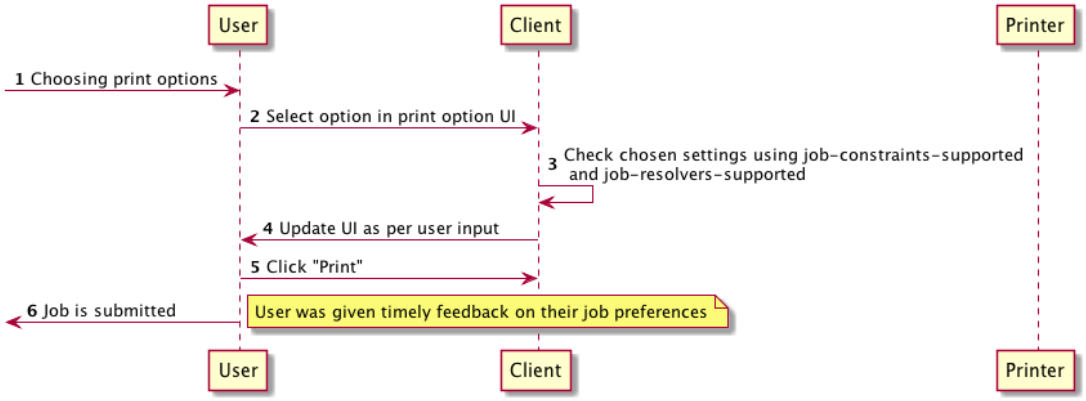

Figure 22: Check Constraints Between Print Options - BETTER

- BEST: Use "job-constraints-supported" and "job-resolvers-supported" as well as IPP Validate-Job

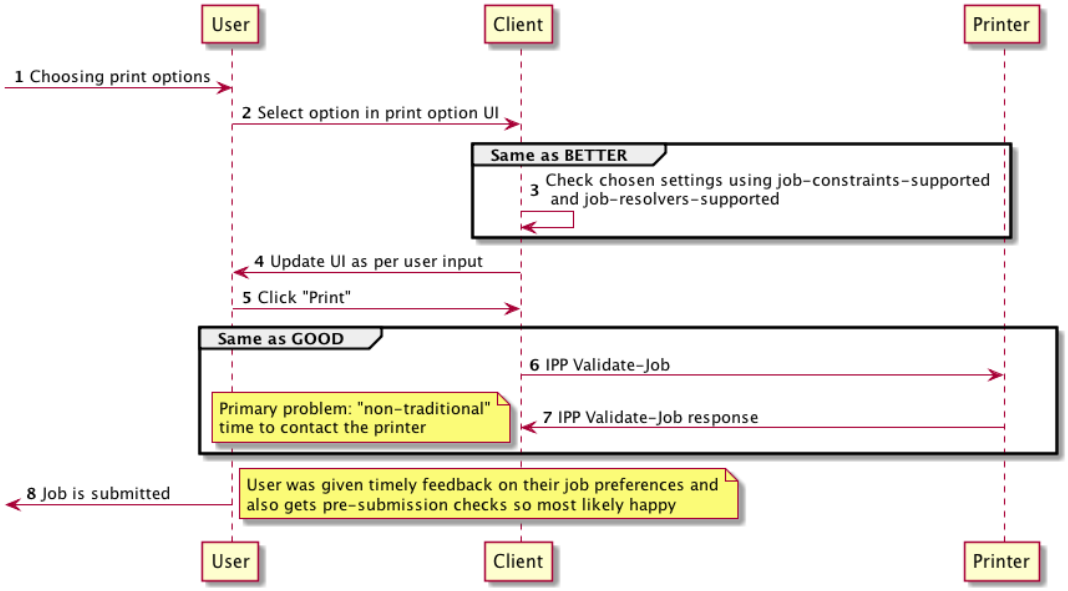

Figure 23: Check Constraints Between Print Options - BEST

4.6 Submit a Print Job

Once the user has selected the desired Job options, the print content is generated, and both the Job options and the content is made available to the Printer so that it can be printed. There are several different ways this can be done with IPP.

The first set of alternatives a Client needs to consider involves how the Document content is conveyed to the Printer. The Client can submit document content via an IPP operation. Alternately, the Client can submit to the Printer a URI reference to a Document, with the expectation that the Printer will retrieve the Document itself at Job processing time.

IPP also provides two methods for requesting that a Printer create a new Job. They differ in the number of operations that are needed. The first method, supported by Print-Job and Print-URI, create the Job and provide the document content or reference all in one operation. The second method creates the Job using the Create-Job operation, submits Document content to the Job via separate Send-Document or Send-URI operations, and might conclude with a Close-Job operation. This second more verbose method is preferred because it provides more opportunities for the Printer to provide control over

4.6.1 Submit a Job with Document Data

This is the classical way that a print Job is sent from the Client to the print service: first a Job is created, and then the Job information and payload content are sent from the Client to the print service.

Alternatives

- POOR: Send the Job with IPP Print-Job
  - The Printer can reject it but only after it has been transmitted. IPP would never respond early; HTTP could respond early but that would be effectively a transport level exception outside the scope of IPP. Better to check ticket and content types first.

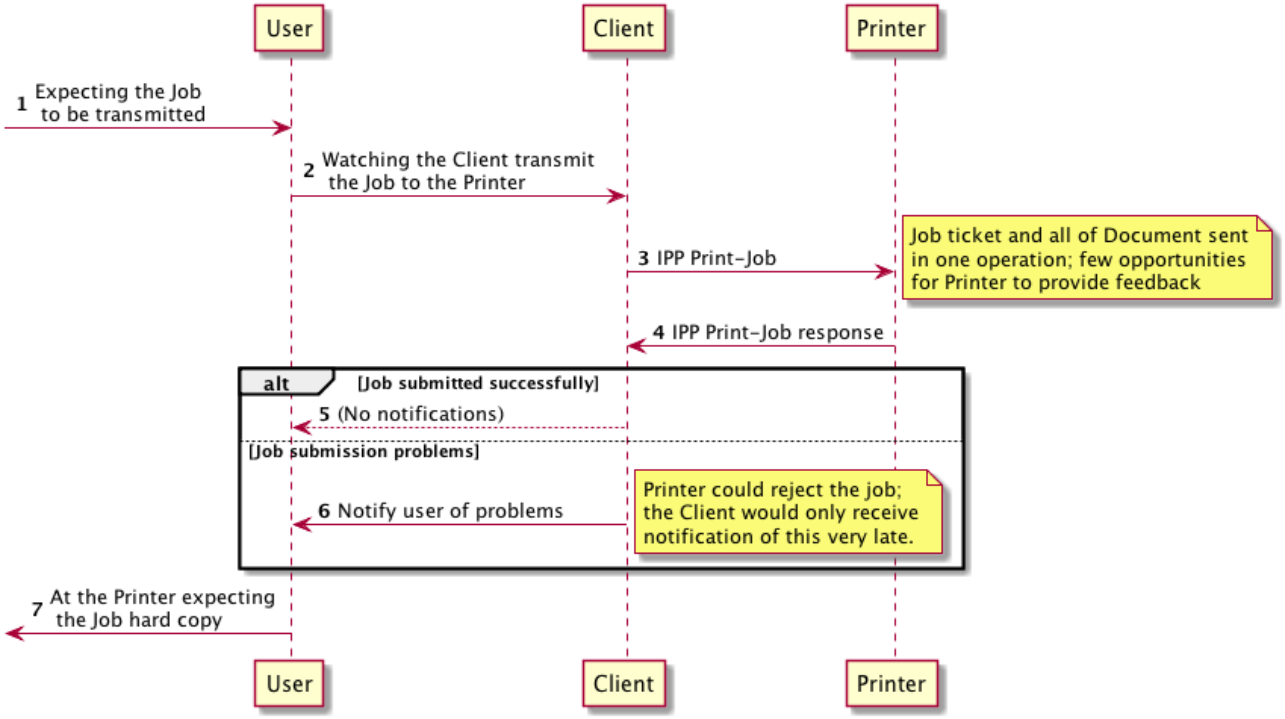

Figure 24: Submit a Job with Document Data - POOR

- GOOD: Pre-qualify the job ticket information with Validate-Job, then submit the Job using Print-Job
    - Doesn't work well with flow-controlled (low-end) printers

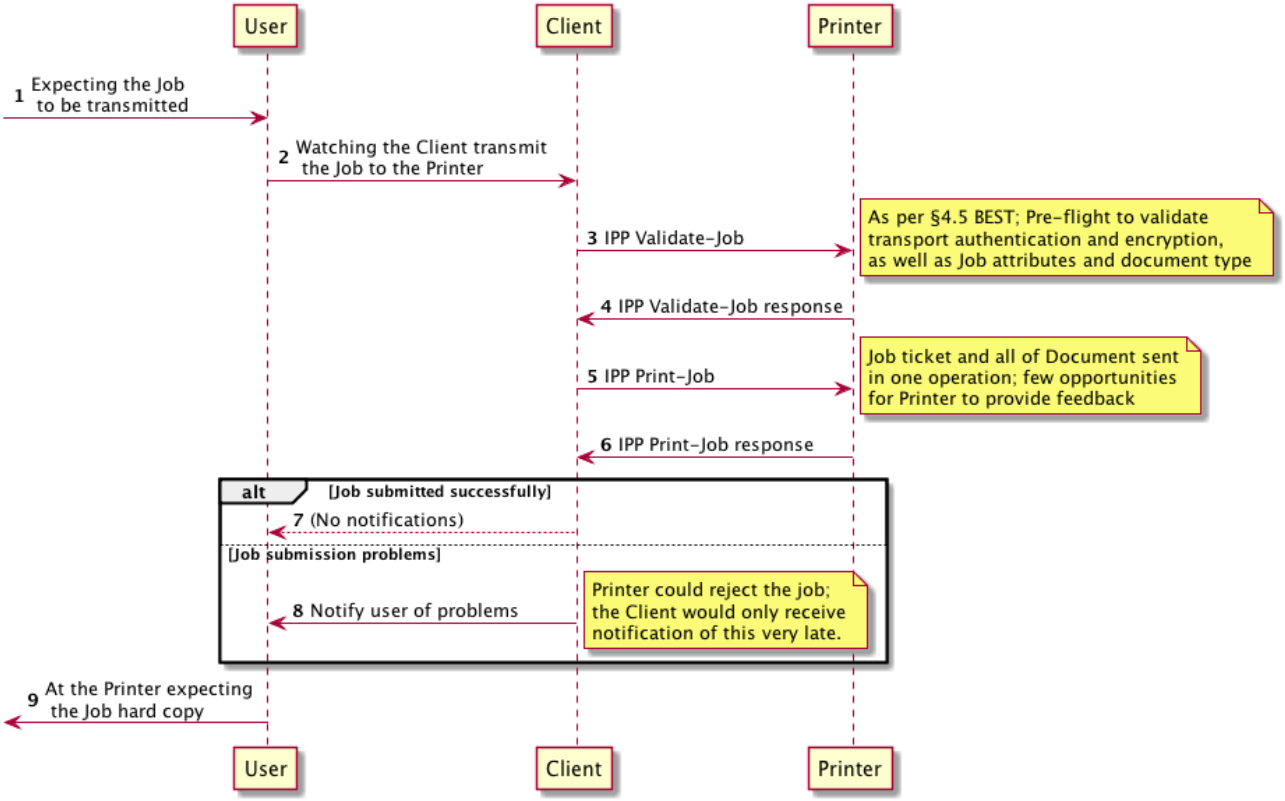

Figure 25: Submit a Job with Document Data - GOOD

- BETTER: IPP Create-Job / IPP Send-Document

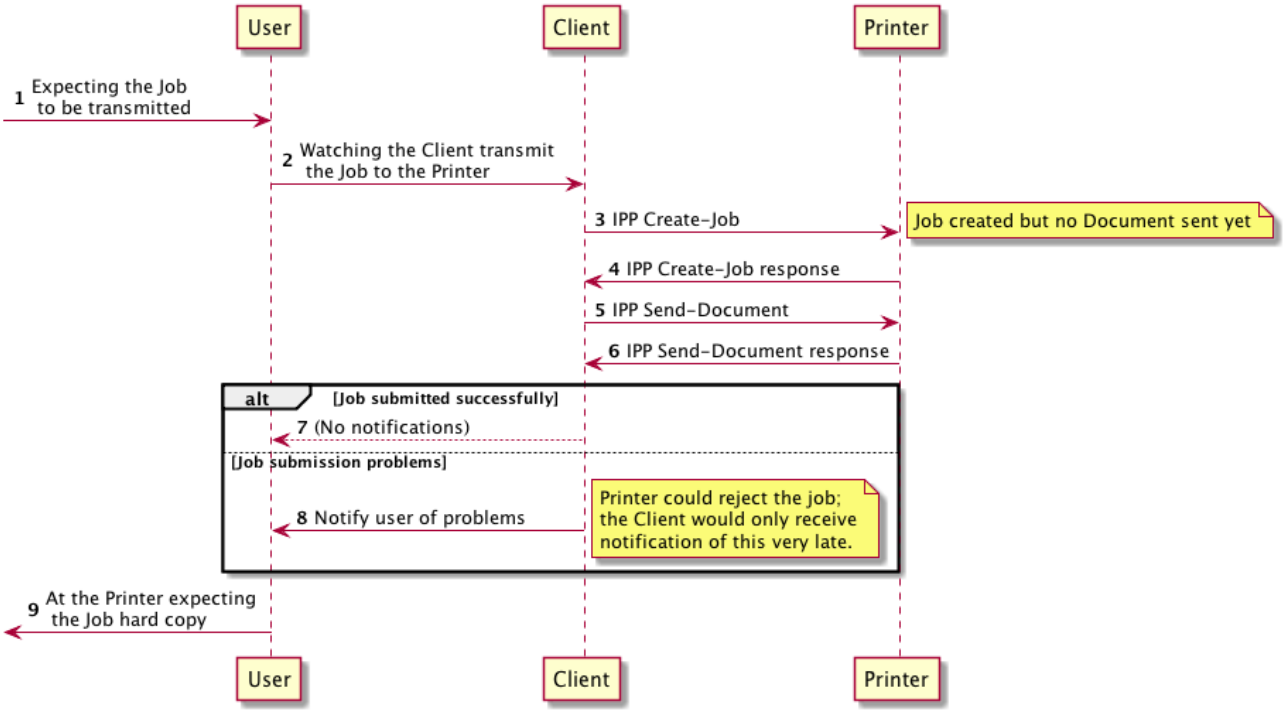

Figure 26: Submit a Job with Document Data - BETTER

- BEST: IPP Validate-Job / IPP Create-Job / IPP Send-Document / IPP Close-Job

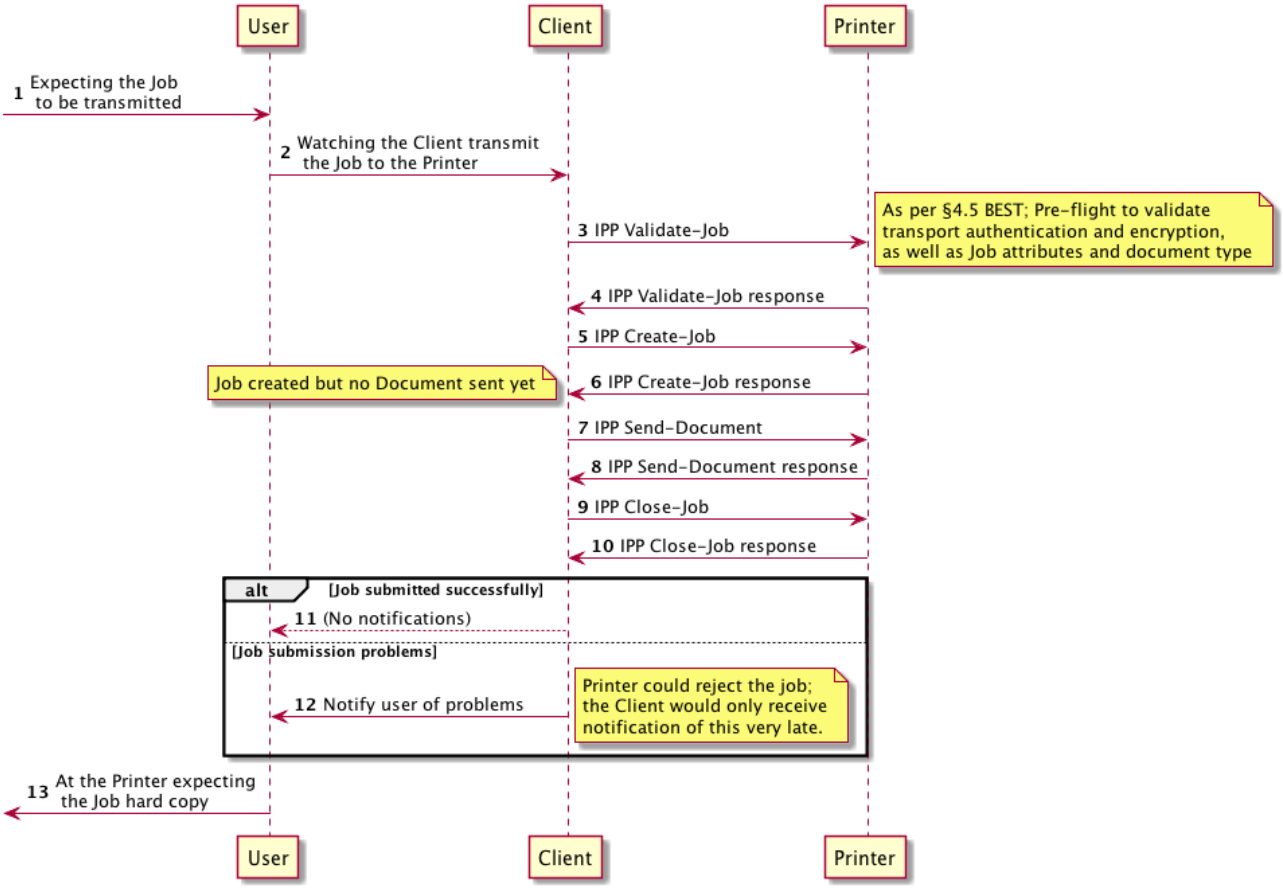

Figure 27: Submit a Job with Document Data - BEST

4.6.2 Submit a Job with Document References

This is a slightly different method for the Document content to be available to the Printer. The notion generally consists of creating a Job and then including in the Job references to documents that the Printer will itself "pull" into the Job. The documents might be on the Client but don't have to be.

This method can be potentially useful for working around certain network topology scenarios or limiting the need for the Client to be transmitting the content directly to the Printer across an expensive link. But there are risks to using this method. If the document is reachable by the Client system but not the Printer, the Job will fail. Reachability can be affected by network topology, authorization, or other factors. And since there can be a time delay between Job creation and Job processing, it could be that reachability problems are not discovered in a timely manner. Therefore, this Implementor's Guide generally advises against using this method unless robust mechanisms are in place to avoid such reachability issues; the Client SHOULD provide URIs that are stable for the life of the Job.

Alternatives

- POOR: IPP Print-URI
  - No pre-flight checks
  - Printer can reject it but only after it has been transmitted
  - Better to check ticket and content types first

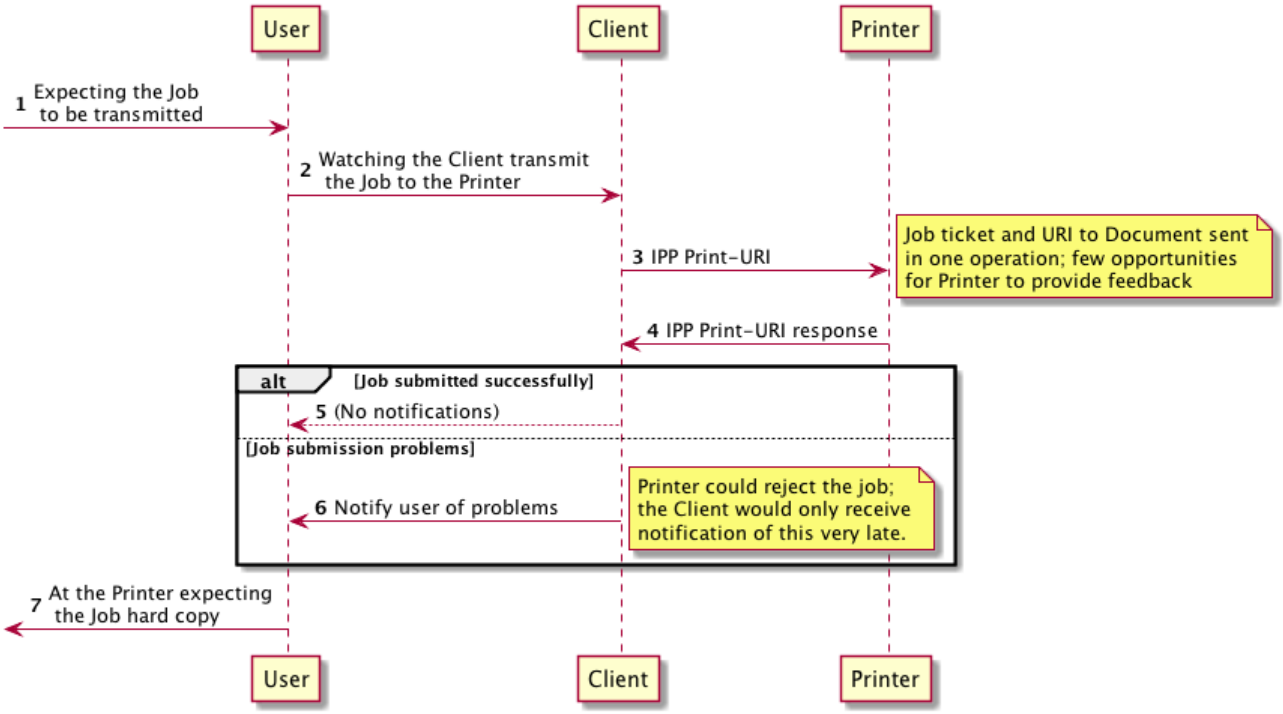

Figure 28: Submit a Job with Document References - POOR

- GOOD: IPP Validate-Job / IPP Print-URI
  - URI might not be accessible at time of processing

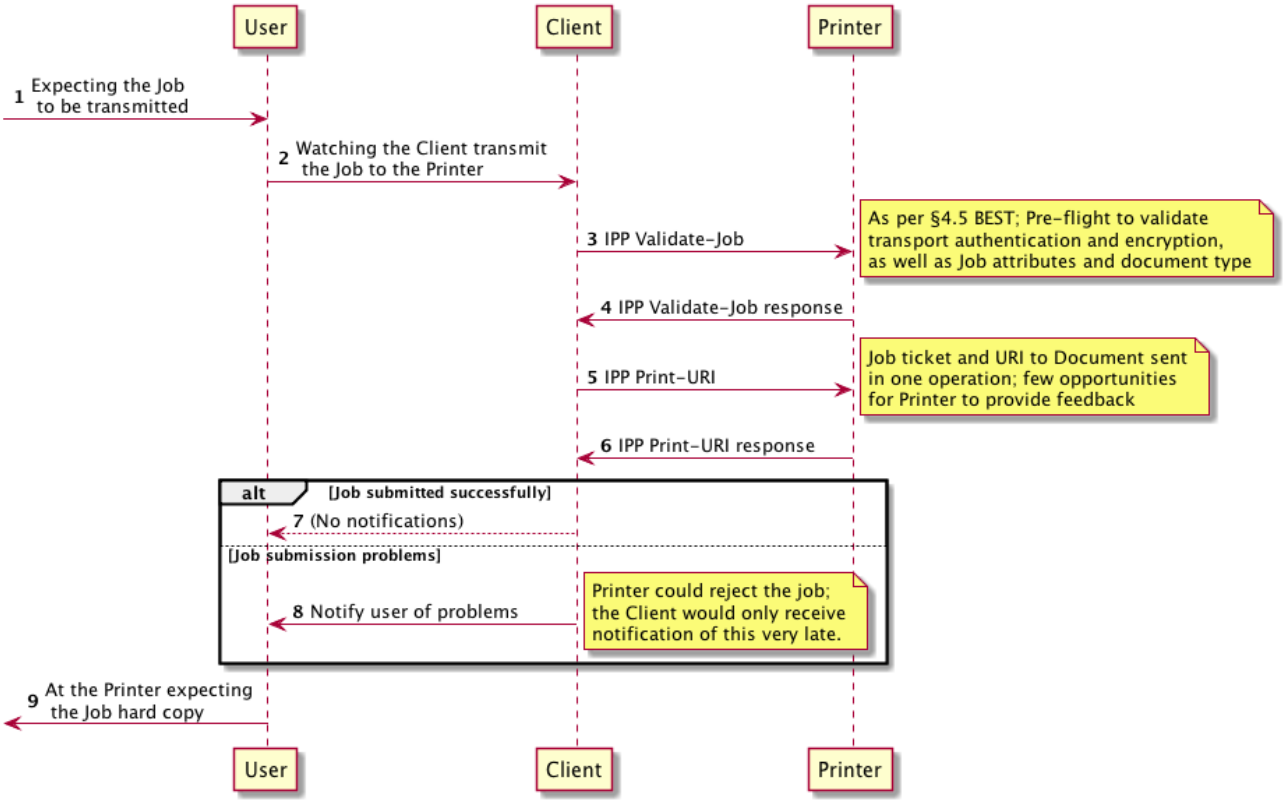

Figure 29: Submit a Job with Document References - GOOD

- BETTER: IPP Create-Job / IPP Send-URI
  - URI might not be accessible at time of processing

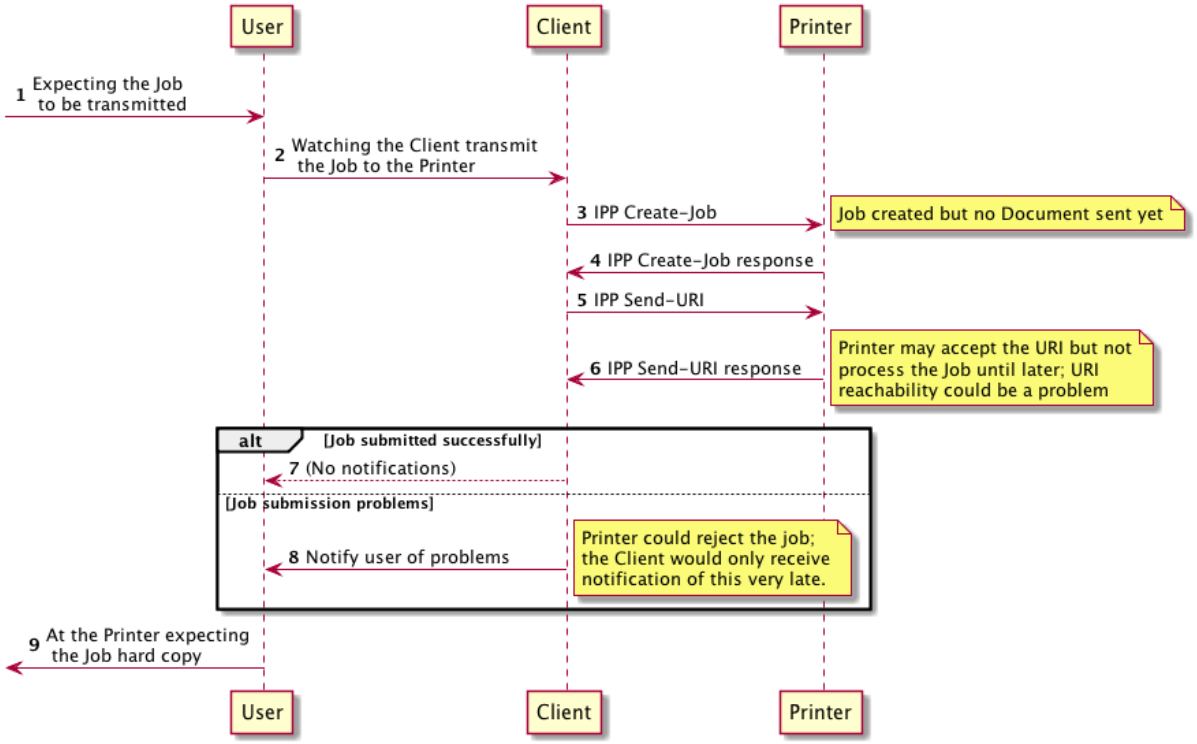

Figure 30: Submit a Job with Document References - BETTER

- BEST: IPP Validate-Job / IPP Create-Job / IPP Send-URI / IPP Close-Job
  - URI equivalent to §4.6.1; better to submit document data to avoid potential URI accessibility issues

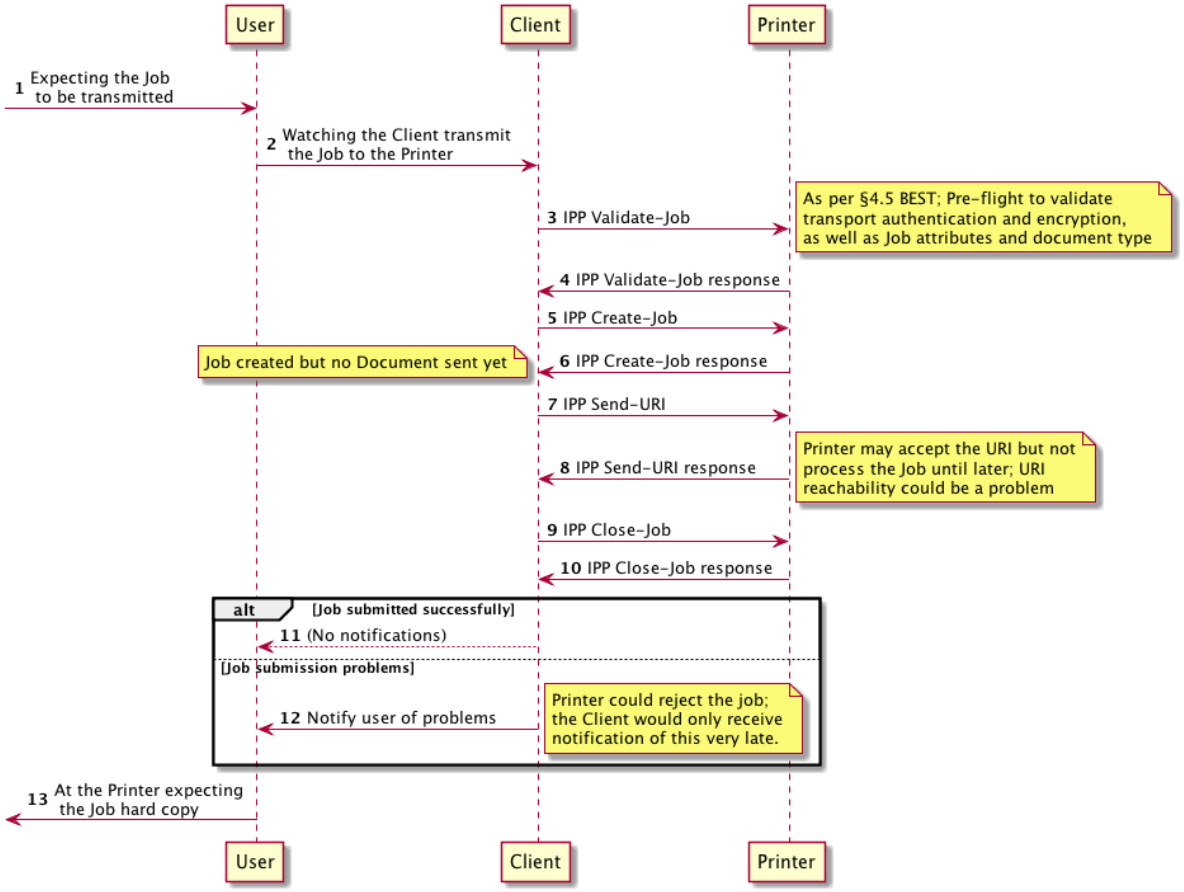

Figure 31: Submit a Job with Document References - BEST

4.6.3 Handle Print Job Creation Errors

If the IPP Printer reports an error during the process of creating the Job or submitting the Documents to that job, it is best if the Client takes action to ensure that the malformed Job doesn't turn into a "Zombie Job", which is a form of digital pollution.

Alternatives

- POOR: Do nothing; leave the malformed Job on the Printer

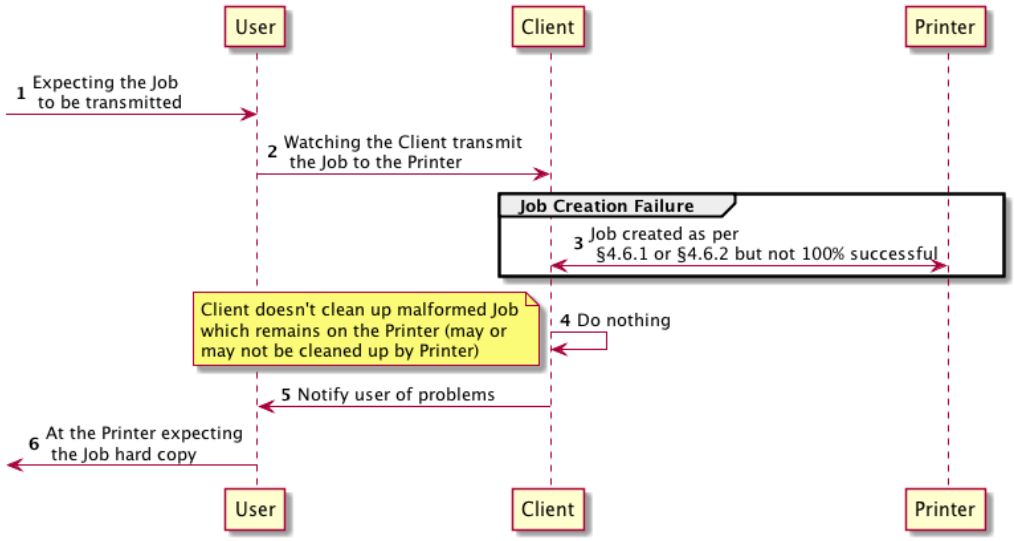

Figure 32: Handle Print Job Creation Errors - POOR

- BEST: Use IPP Cancel-Job to cancel the Job to clean up the malformed Job

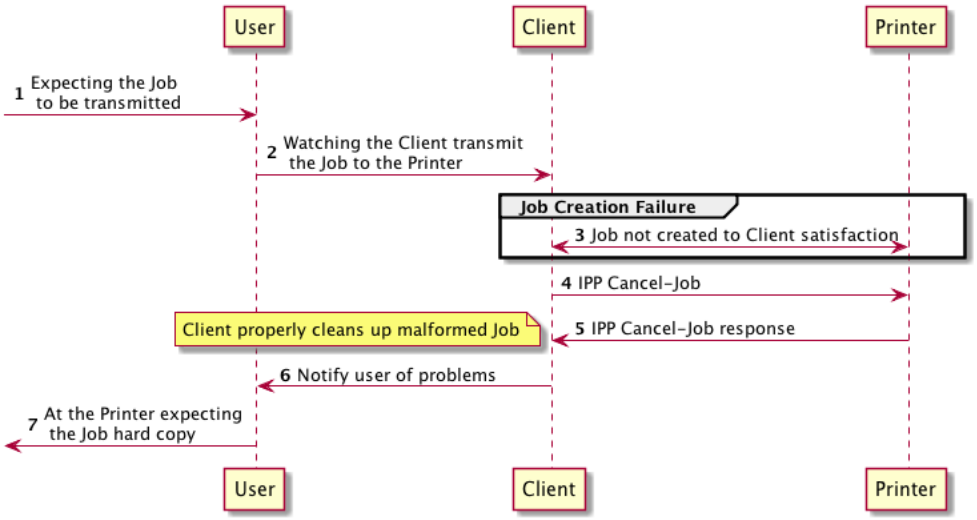

Figure 33: Handle Print Job Creation Errors - BEST

4.7 Monitor Job Status

While the Job is being processed, users might wish to know whether it is proceeding successfully, or whether there are conditions that they need to handle that are preventing

processing from proceeding, such as a media jam, open covers, marking agents depleted, and so forth.

To provide this using IPP, a Client SHOULD monitor the status of the various IPP objects involved, including the Printer, the Job, and Documents within that Job. A Client SHOULD monitor the status of a Job submitted to a Printer and SHOULD continue to monitor it until it has been successfully printed.

The basic system for monitoring IPP object status is to periodically submit one or more IPP operations to retrieve these objects' status. Obviously, this is "polling", and as with most polling based systems, this doesn't scale well and causes frequent unnecessary updates to be communicated. Ideally, this would be avoided and replaced with a system based on asynchronous event notifications.

IPP has an event notification system ([RFC3995], [RFC3996], [RFC3997]). Unfortunately, this system has been largely overlooked. Both IPP Clients and Printers can support IPP notifications to minimize network traffic and monitoring overhead. Section 5.11 discusses this in more detail.

For those options below that involve polling the Printer Object, the degree to which the option is better or worse is due in no small part to the polling frequency. The interval SHOULD be tuned so that the frequency of queries is not so great that it burdens the Printer Object or Job Object or the network, but not so small that there is an undesirable lag between when an event occurs and when the user is notified. It is always a bad practice in any case if a Client is polling as fast as the network can handle traffic. When polling the Printer Object, the Client SHOULD NOT poll as fast as the network can handle traffic; it SHOULD instead tune its polling interval to achieve a suitable balance between user responsiveness and resource overuse.

In some cases, the Client will cease monitoring the Job once the Job Creation or Document Creation operations have concluded. There are many scenarios where the User would prefer to be notified about Job status until the Job has finished being processed. For example, with a Job requesting 500 copies of a document containing 4 pages, the Job Creation operations most likely will conclude minutes before the Job has concluded being processed by the Printer. A well-implemented Client SHOULD monitor Job status until the Job has reached a terminal state. If IPP Subscriptions are used, these notifications can be monitored and received quite a long time after the Job Creation operations have been performed.

Alternatives

- BAD: Poll Printer for general Printer status instead of precise Job status
  - Polling all status information needlessly uses a large number of resources (network bandwidth, CPU)
  - Polling for status without the actual Job ID is imprecise

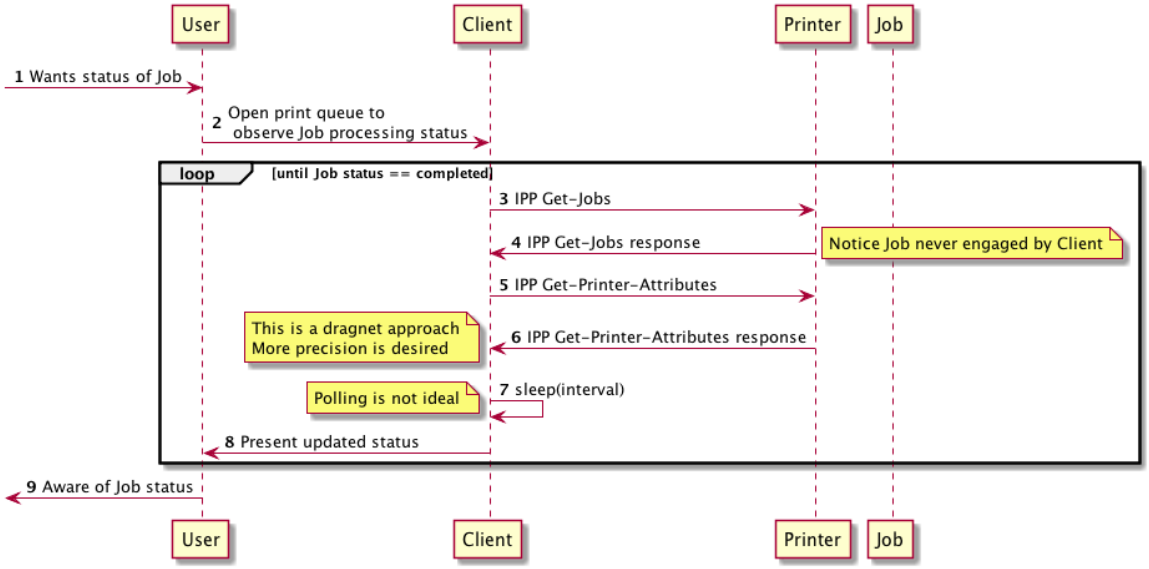

Figure 34: Monitor Job Status - BAD

- POOR: Poll for Job status using Get-Job-Attributes
  - Monitor the Job with ID acquired earlier, according to a polling interval

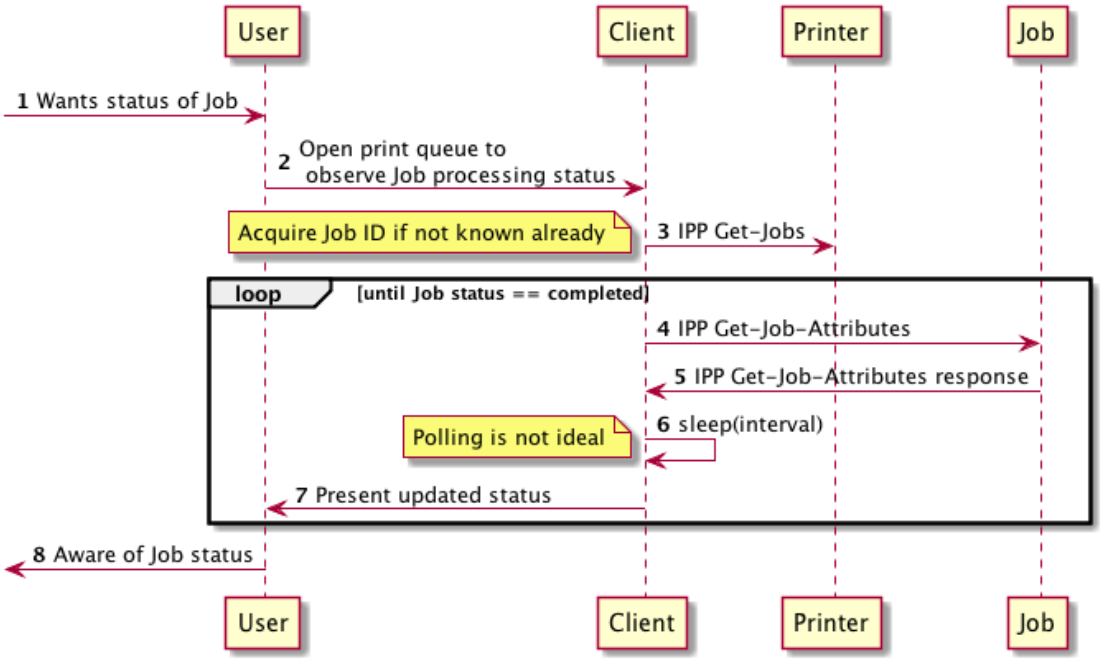

Figure 35: Monitor Job Status - POOR

- GOOD: Monitor both Job and Printer status, but restricting Printer status to only attributes germane to status monitoring
  - Monitor the value of "printer-state" attribute as well as targeted monitoring of a specific Job's status
  - Only ask for the attributes you need, to minimize the bandwidth needed for the information being used.

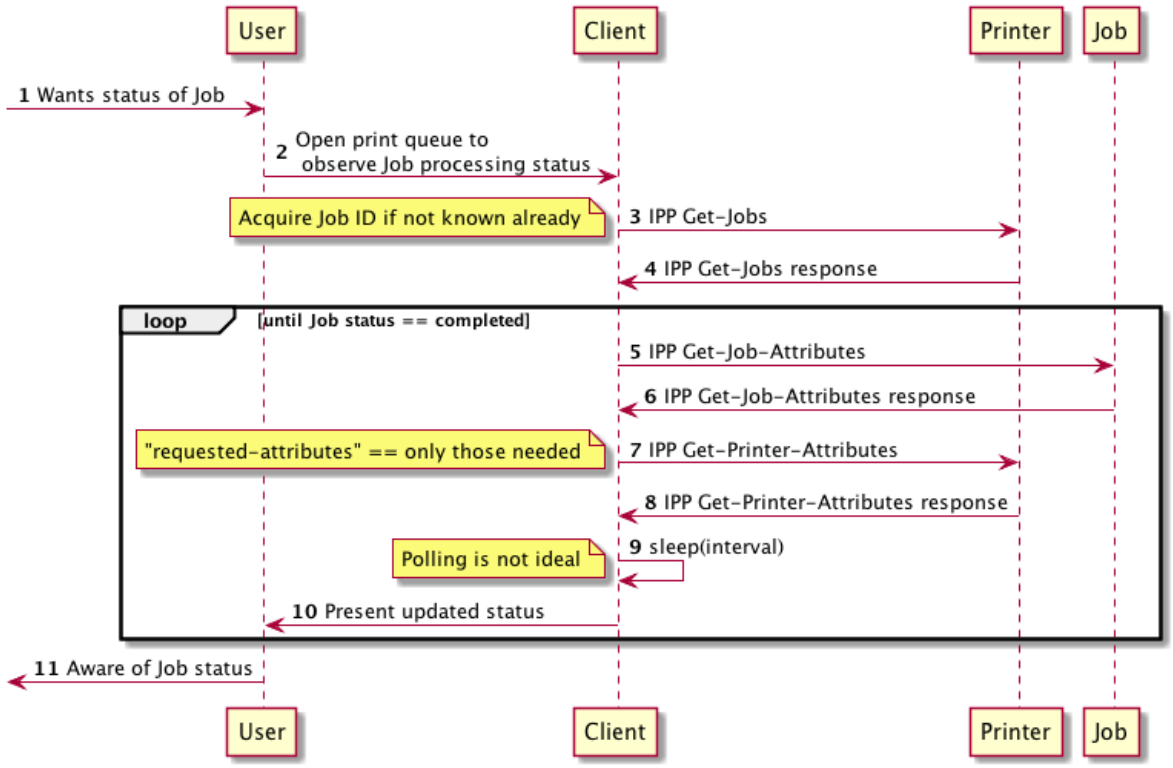

Figure 36: Monitor Job Status - GOOD

- BETTER: Use IPP Event Notifications and Subscriptions [RFC3995]
  - Asynchronous / long running queries for notifications that don't require polling
  - When Job has completed, query the state of that Job
  - Printer state changes will be provided by subscribing to the printer; subscribing to the Job will provide less information and not be as useful

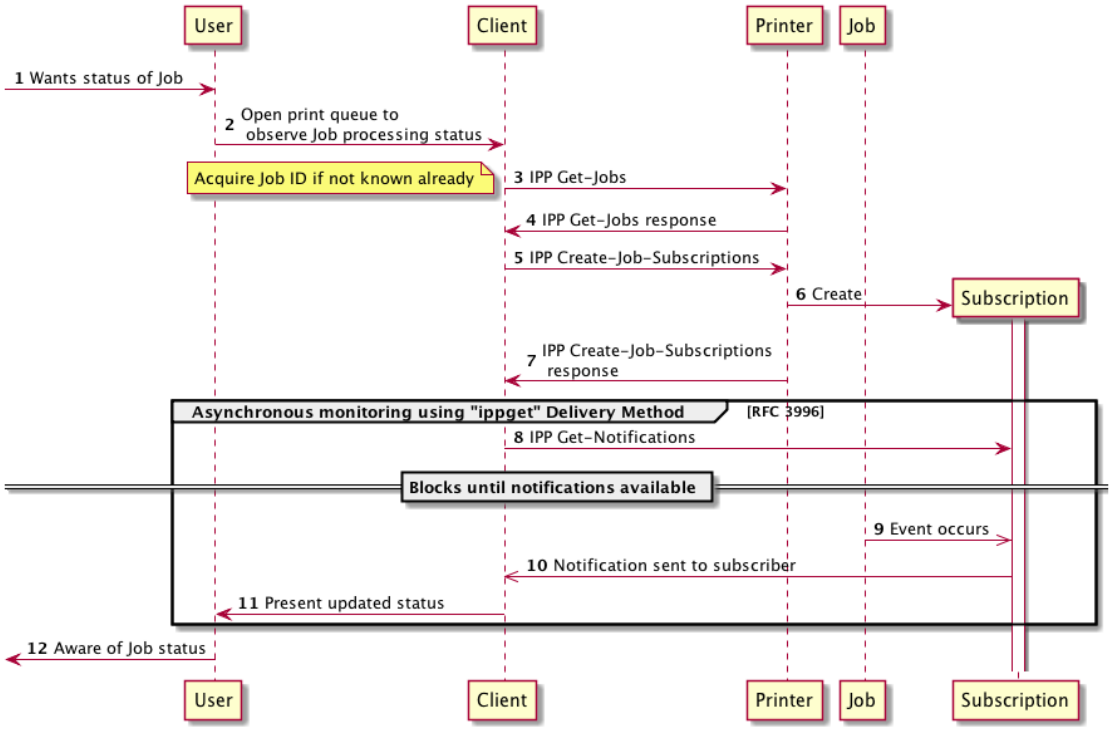

Figure 37: Monitor Job Status - BETTER

- BEST: Create the Job using methods in section 4.6 and include the Subscriptions group in the Job Creation operation request; then monitor using the IPP Event Notifications and Subscriptions method [RFC3995]
    - Most optimal if IPP Create-Job / IPP Send-Document is used so that the Job URI is positively sent by the Printer Object

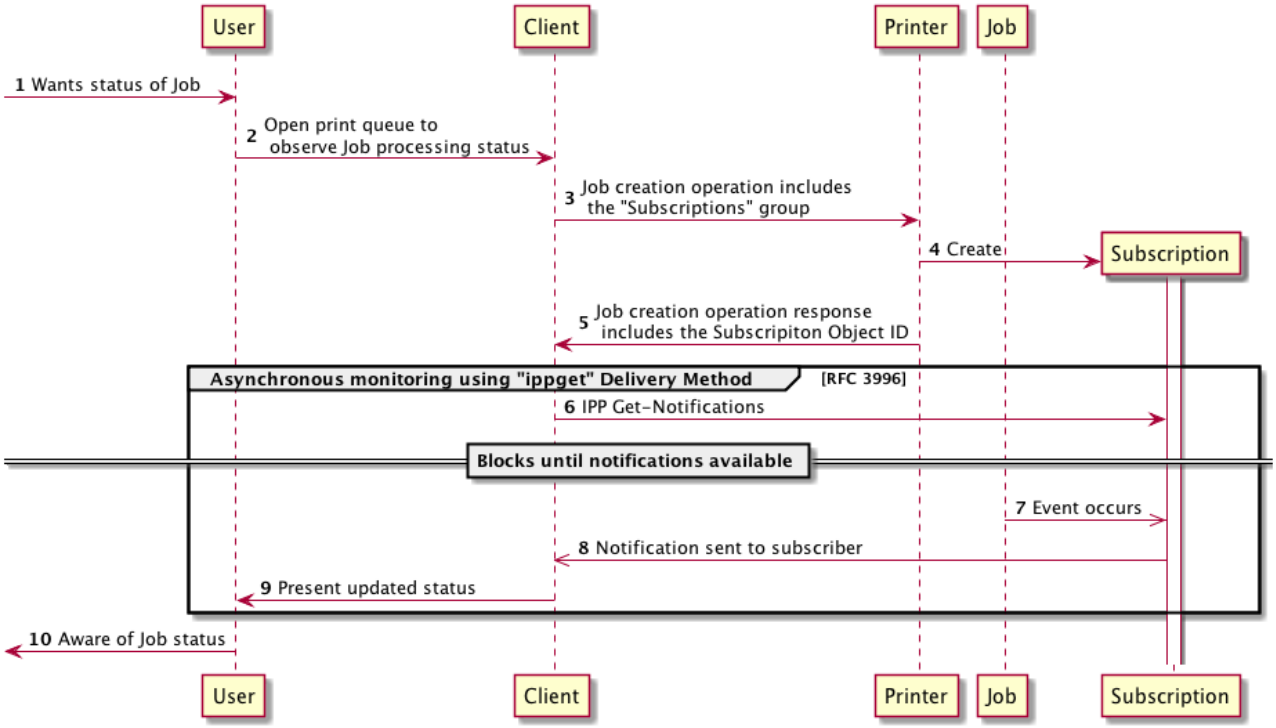

Figure 38: Monitor Job Status - BEST

4.8 Cancel a Print Job During Job or Document Creation

Situations arise where the user wants to terminate a Job before it is processed. This can occur before the Job's Documents have been completely transmitted to the Printer, or it might occur after the Job has been created but is awaiting processing. If the Job has already been completely created successfully, the Job can be terminated simply using an IPP Cancel-Job operation.

If the Job hasn't been fully created yet there is the risk that artifacts of that incomplete Job might remain. A well-implemented Client SHOULD clean up such artifacts if the Job is not completely created or it's Documents not all submitted; leaving broken Job objects abandoned on the Printer is bad form.

There is some a dependency between the options below and how the Job was submitted. Adopting a high quality option from those listed below will likely depend on implementing a high quality option from §4.6.

Alternatives

- BAD: Stop sending octets and abandon the connection
  - Abandoning the connection after partial successful data transmission could cause incomplete Job objects to linger

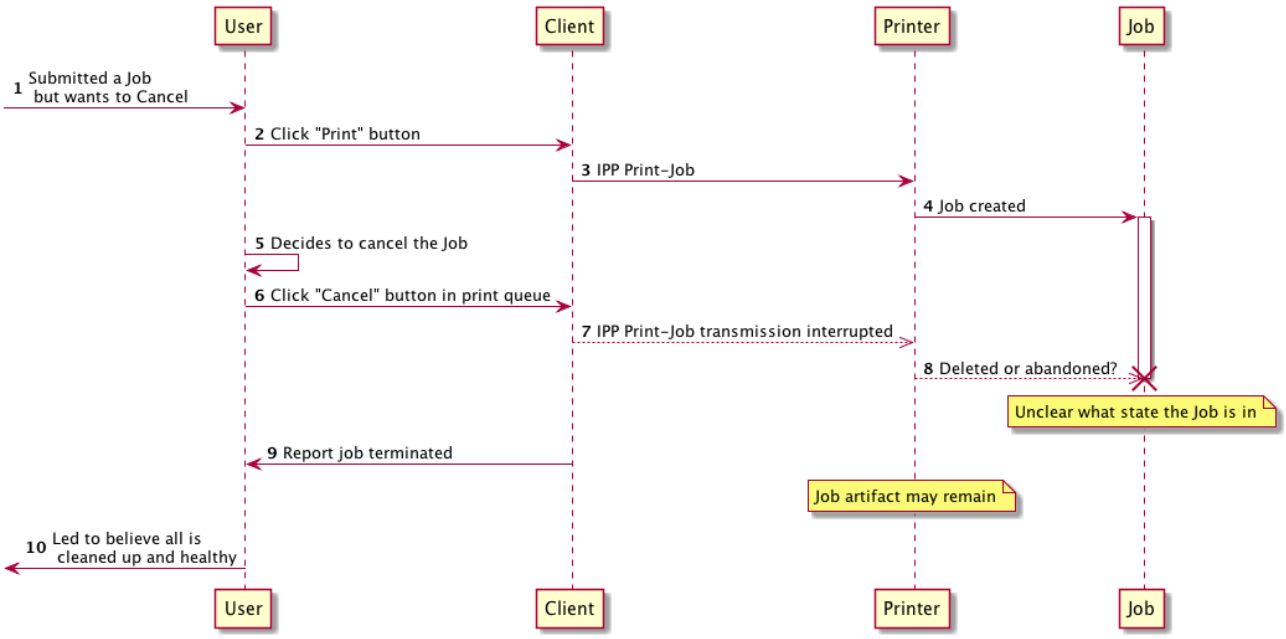

Figure 39: Cancel a Print Job During Job or Document Creation - BAD

- GOOD: Stop sending on a page boundary; properly terminate Document during IPP Print-Job operation; IPP Cancel-Job to cancel the Job object

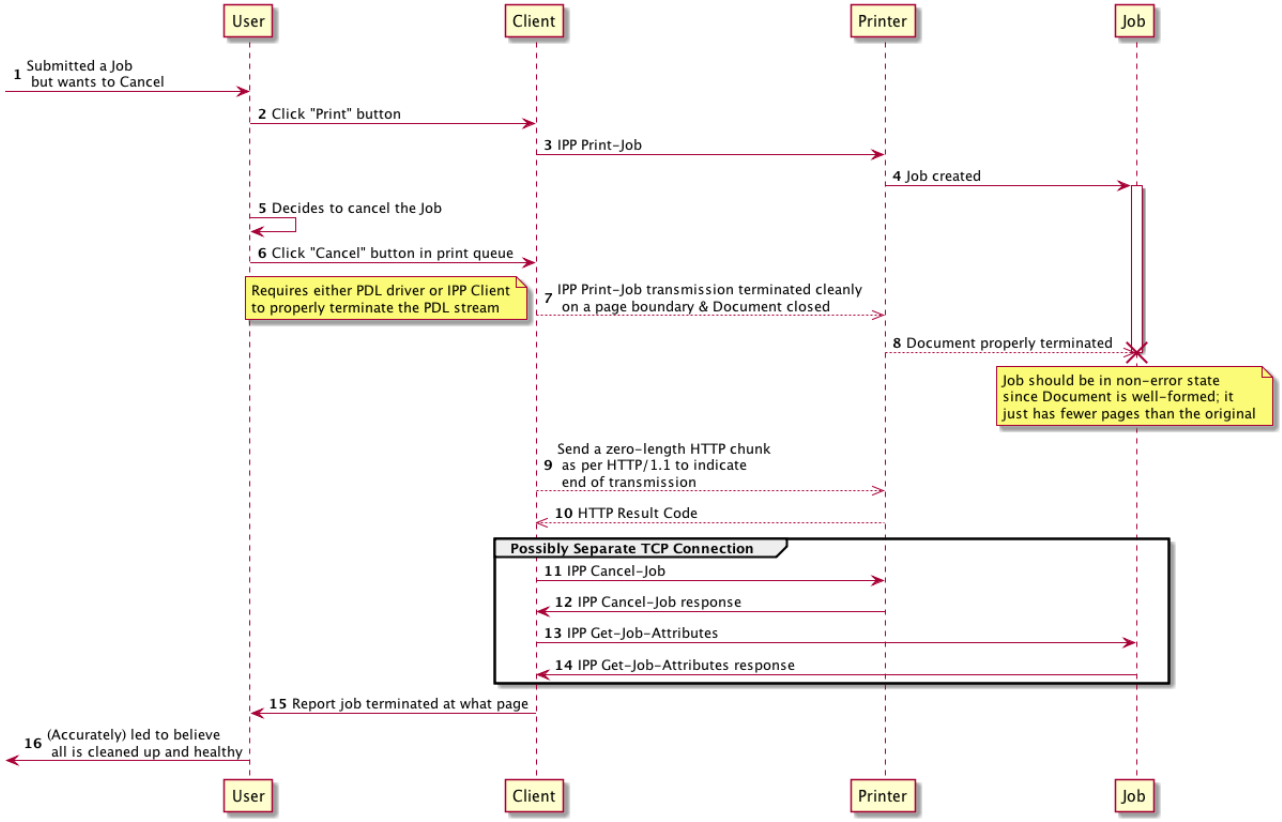

Figure 40: Cancel a Print Job During Job or Document Creation - GOOD

- BEST: Stop sending on a page boundary; properly terminate Document during IPP Send-Document operation; IPP Cancel-Job to cancel the Job object

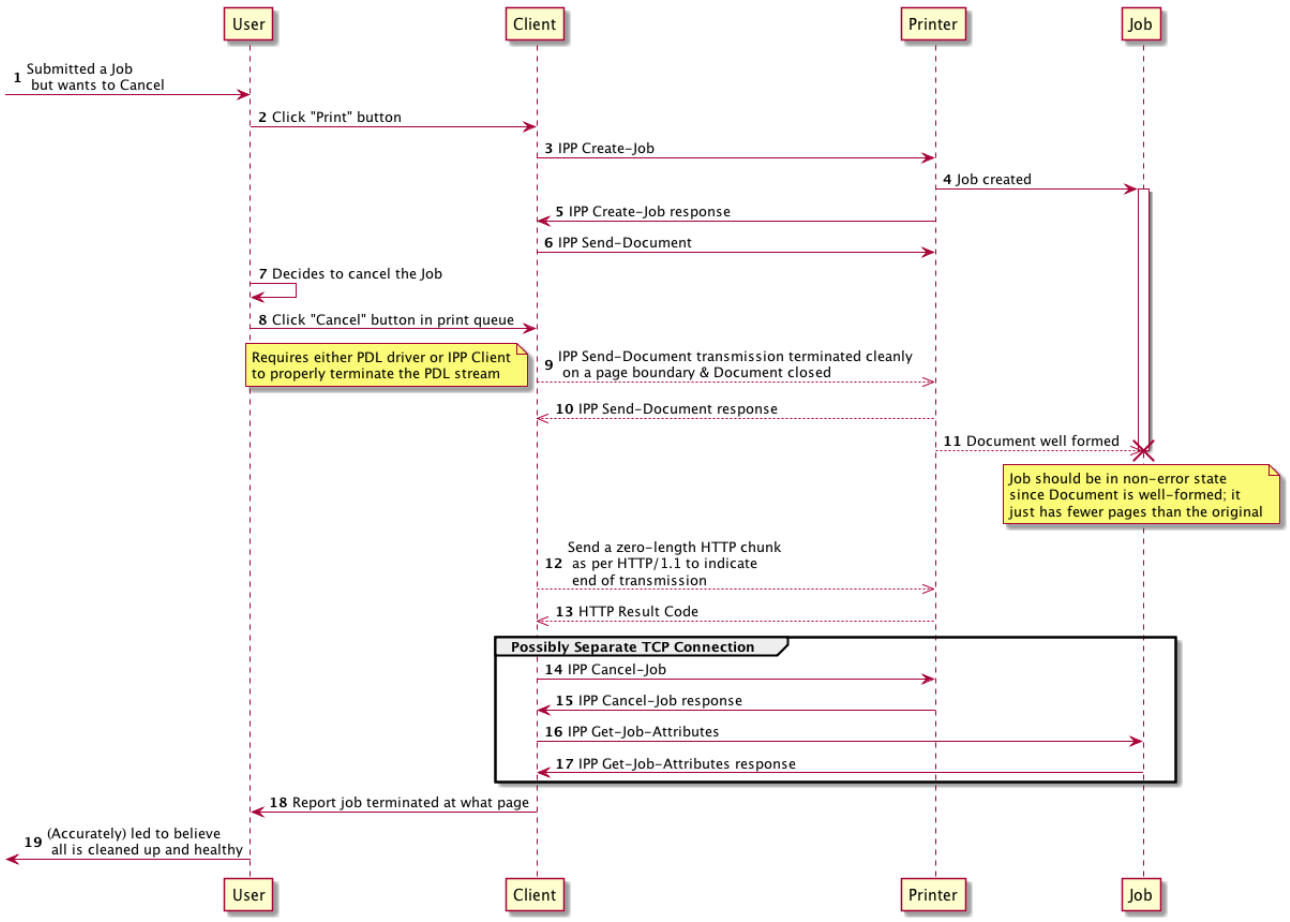

Figure 41: Cancel a Print Job During Job or Document Creation - BEST

4.9 Get Printer Supplies Status

Some administrative tasks, like checking or monitoring consumables levels, are presented to Users in print dialogs or Job status dialogs. It could also be monitored by printer management software or device status monitoring software on User systems. A well implemented Client SHOULD provide supplies status. Ideally, the Client would check supplies status using publicly specified extensions to IPP.

Alternatives

- BAD: Don't provide supplies status
  - Not good for users or printer vendors

- POOR: Use some proprietary protocol or (possibly closed) platform-specific extension to IPP
  - This falls short of the ideal that all devices implement a publicly specified and open extension to IPP
  - Private extensions to IPP are less desirable than a publicly specified and open protocol other than IPP

- GOOD: SNMP and standard printer MIBs
  - Public standard but not IPP, so not ideal

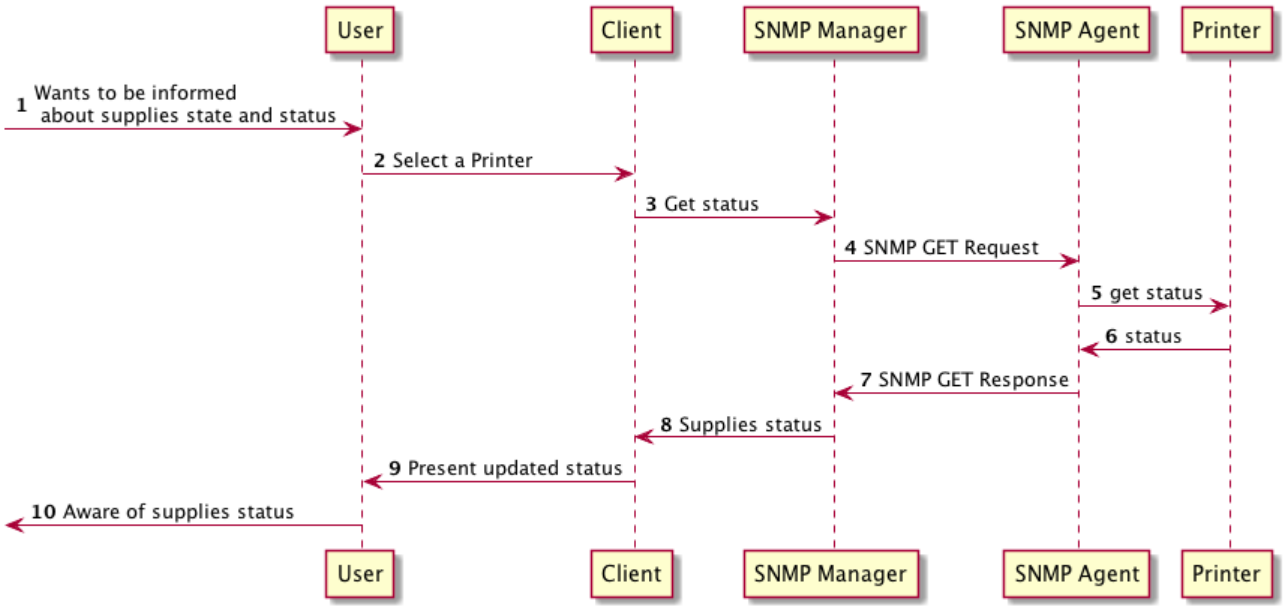

Figure 42: Get Printer Supplies Status - GOOD

- BETTER: IPP Get-Printer-Attributes to retrieve "printer-supply" attribute
  - Printer needs to implement "printer-supply" attribute
  - Supplies events in polling model can monitor "printer-state-reasons" for 'marker-toner-low' and similar states

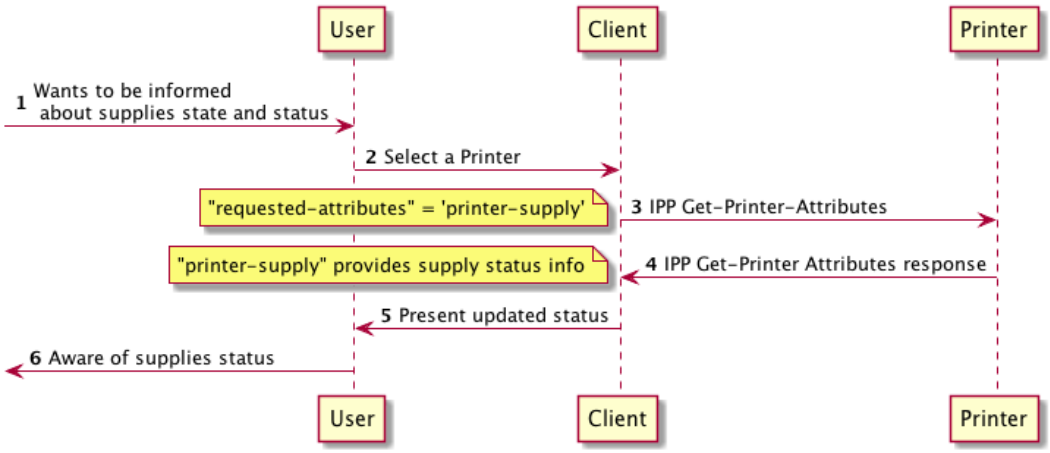

Figure 43: Get Printer Supplies Status - BETTER

- BEST: Monitor supplies status using the IPP Event Notifications and Subscriptions method [RFC3995]
  - Printer needs to implement "printer-supply" attribute to support this scenario
  - Printer MAY send affected attributes, e.g. "printer-supply", "printer-input-tray", "printer-output-tray", etc.

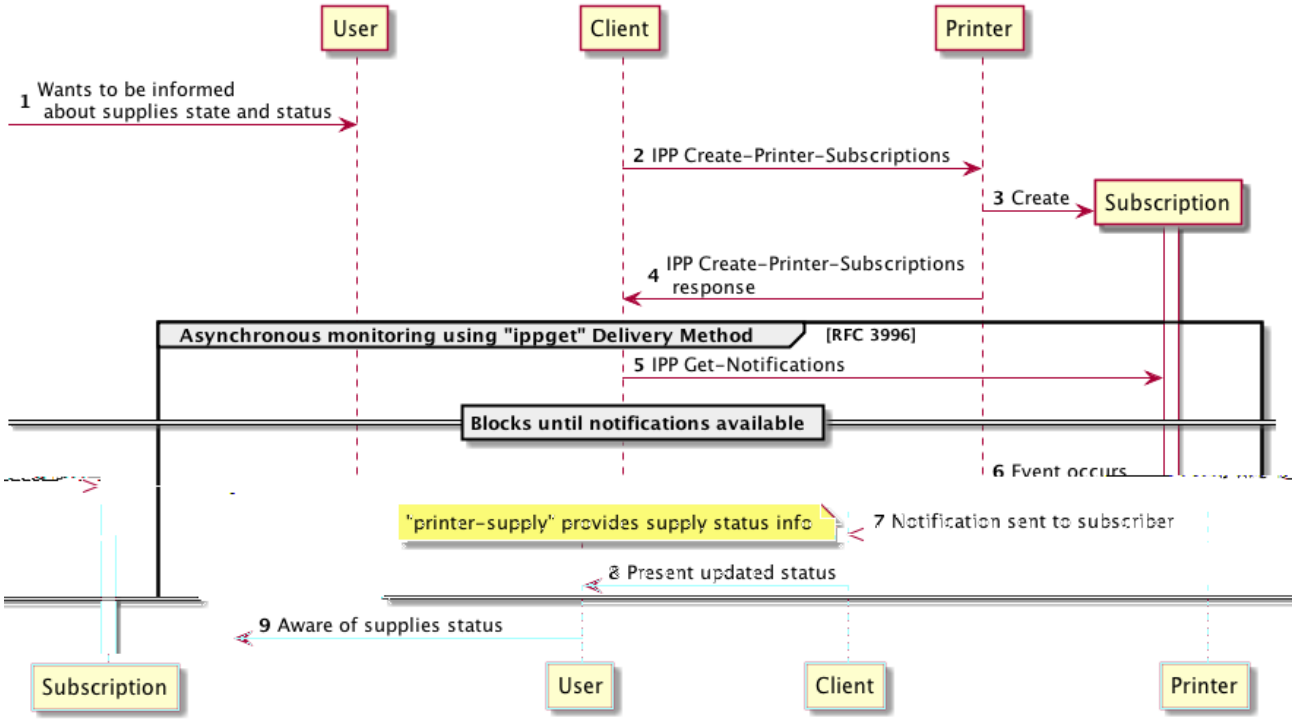

Figure 44: Get Printer Supplies Status - BEST

4.10 Error Handling

When a Client receives notice of an error, there are various ways to present that to the Client's user. Possible response actions include: presenting messages to the user; flagging or filtering those messages to indicate their severity and recoverability; and logging the event in an appropriate way.

A Client SHOULD log errors using the PWG Common Log Format [PWG5110.3].

5. Using and Evaluating IPP Attributes

There are many IPP attributes that are crucial to providing a quality user experience. In a number of cases, there are nuances about their use that need to be properly understood for them to be used properly. Below are some of these attributes and how they can best be used in various contexts, to help ensure an optimal user experience.

5.1 Job Template Attributes and Value Binding

A Printer conveys a Job Template attribute's default value using the corresponding "xxx-default" attribute. These attributes indicate the value the Printer will use if the Client does not provide that Job Template attribute at Job Creation time. The "xxx-actual" attributes reported by the IPP Printer in its operation response will report the values that are bound to the Job.

When a Client performs job submission operations, the Client can choose to not include Job Template attributes. If the Client doesn't provide the attribute, the Printer will use the "xxx-default" attribute's value. However, while it is technically possible for a Client to only send those Job Template attributes whose values do not match the default values, this strategy has undesirable side effects. If the value of the "xxx-default" attribute were to change between Job submission time and Job processing time, the resulting output from the Job might not meet user expectations. Attribute default values SHOULD be used only for setting the initial state of choice options for the User. This not only ensures the user's expectations are met; it also makes implementing the client simpler because the Client doesn't need to maintain any knowledge of whether the value chosen by the user matches the default provided by the Printer when the Job Template attributes were fetched to populate the user selection choices.

Using the "sides" Job Template attribute as an example, if the Printer reports that "sides-default" is 'one-sided' and "sides-supported" is {'one-sided', 'two-sided-long-edge', 'two-sided-short-edge'}, a well-implemented Client SHOULD populate its "sides" Job option choice with the value from "sides-default": 'one-sided'. If the user makes some other choice, then the "sides" attribute would be updated to hold that new value. Once the user was satisfied with the choices and had requested the Job be printed, the Client would include the "sides" attribute and the value it holds in the Job Creation operation. Changes on the Printer to "sides-default" between the time the Job was submitted and the time it is processed will have no effect on the Job.

To help eliminate these pitfalls even for Clients that are poorly implemented and don't submit attributes that match the "xxx-default", a well-implemented Printer SHOULD bind Job Template attribute default values to each Job Template attribute at Job Creation time rather than waiting to do so at Job processing time, to help ensure user expectations.

5.1.1 IPP Client Recommendations

A Client SHOULD plan to submit all Job template attributes at Job creation time. A Client SHOULD set the initial state of each Job Template attribute to the corresponding Job Template attribute default value, if one exists.

5.1.2 IPP Printer Recommendations

A Printer SHOULD make that value binding visible to Clients as soon as possible by setting the xxx-actual Job Status attribute using that value as well.

5.2 Document Format Selection

A Client uses the "document-format" attribute in several ways. The Client evaluates the Printer's document-format-supported attribute to decide on the most appropriate document format type. Once the Client has selected a format, it SHOULD perform a second Get-Printer-Attributes operation including "document-format" in the request, to request the

Printer send the set of Job Template attributes and attribute values that correspond to that document format. This is discussed in detail in §4.4.

Several factors, including time delays between Job receipt and Job processing as well as implementation challenges, can prevent an IPP Printer from receiving robust and timely feedback from its document format interpreter and rendering engine about the document content submitted with a Job. It is important for the Client to be as descriptive as possible about the construction of the Job, including not only that Job's IPP attributes but also accurately reporting the document format for each Document associated with that Job.

5.2.1 IPP Client Recommendations

As with all other Job Template attributes, a Client SHOULD include the "document-format" attribute with all Job creation operations, with the value set to the actual MIME Media type of the document format to be submitted to the Printer. A Client SHOULD NOT submit a Job with the "document-format" attribute specifying "application/octet-stream" as the document format.

5.2.2 IPP Printer Recommendations

A Printer SHOULD NOT implement 'application/octet-stream' as the only document format. A Printer SHOULD list all the document formats it supports (even vendor-specific ones) via the "document-format-supported" attribute. A Printer SHOULD NOT use 'application/octet-stream' as its "document-format-default".

A Printer SHOULD send the "document-format-actual" attribute in relevant IPP responses to report the detected MIME media type. If a request contains a document payload, the Printer SHOULD send the "document-format-actual" attribute value in the IPP operation response. The value of "document-format-actual" SHOULD be the detected format of the payload, not simply the value of the "document-format" attribute sent by the Client in the request.

5.3 Prefer "media-col" Attribute to "media" Attribute

The "media-col" attribute provides finer and more reliable control over media selection. As an example, the "media-col.media-size" sub-attribute is defined using integer value types rather than floating point value types, which avoids floating point calculation problems when converting between PWG size names and dimensions.

5.3.1 IPP Client Recommendations

Given a Printer Object that supports both "media" and "media-col" attributes, a Client SHOULD prefer to use the "media-col" attribute with operations that accept it. This is true for when "media" and "media-col" are top-level attributes as well as when "media" or "media-col" might be included within other collection attributes, such as "job-sheets", "job-error-sheet", "job-accounting-sheets", and others.

5.3.2 IPP Printer Recommendations

The Printer SHOULD allow for a limited degree of inaccuracy even with "media-col.media-size". If the dimensions are off by a small amount (1-3 units) this SHOULD NOT be regarded by the Printer as a mismatch.

For more discussion of the "media-xxx" attributes, see §5.8.11.

5.4 Extended Finishings Options with "finishings-col-XXX"

The IPP "finishings", "finishings-supported" and "finishings-default" attributes [RFC2911] define a basic mechanism for describing finishings capabilities. The "finishings-col-XXX" attributes [PWG5100.1] [PWG5100.3] extend the "finishings-XXX" attribute set by providing a mechanism to convey detailed description of finishing settings. In certain contexts, it is advantageous to use "finishings-col" rather than "finishings" to describe the finishings options, if the Printer supports both. A Printer can precisely describe the implementation underlying each "finishings" keyword using corresponding "finishings-col-database" and "finishings-col-ready" entries. A Client can use these detailed descriptions to provide an accurate print preview, and also provide a scalable user experience that can be simple but allow more detailed user control if desired.

5.4.1 IPP Client Recommendations

A Client SHOULD support "finishings-col-XXX" [PWG5100.1] to allow it to acquire from the Printer detailed descriptions of each value listed in "finishings-supported". This is especially important if the Client needs to map between IPP and JDF or other rich Job ticket formats.

5.4.2 IPP Printer Recommendations

A Printer SHOULD support IPP Finishings 2.0 [PWG5100.1].

5.5 Controlling Intended Output Using "ipp-attribute-fidelity", "job-mandatory-attributes", and "pdl-override-supported"

There are two attributes that might be used by a Client to control whether a print Job will print EXACTLY as requested, or alternately print with "best effort" but might not comply with all requested Job attributes. These attributes are listed below, each with a synopsis of their meaning and purpose.

ipp-attribute-fidelity [RFC2911] - Attribute provided by IPP Client in Job submission operations. If absent the IPP Printer will assume the value is ‘false’. If present and value is ‘true’ then the IPP Printer is required to support all Job attributes and attribute values included in the Job submission operation or else the IPP Printer is required to reject the operation.

job-mandatory-attributes [PWG5100.7] - Lists the Job Template attributes the Printer is required to support in a Job Creation request in order to accept the Job. This allows the IPP Client to specify at a finer level of granularity those attributes that need to be supported. If "job-mandatory-attributes" is included and it lists all the Job attributes, this is equivalent to including the ipp-attribute-fidelity attribute with value set to ‘true’.

pdl-override-supported [RFC2911] - Specifies whether the Printer is capable of overriding Job attributes embedded in the Job document(s) with IPP Job attributes, and with what level of reliability: 'attempted" [RFC2911], 'not-attempted' [RFC2911], 'guaranteed' [PWG5100.11]. Although 'attempted' is supposed to convey that the Printer will try to perform overrides, best effort results are variable. Also, for any PDL other than simple raster document formats, "guaranteed" is impractical to implement.

5.5.1 IPP Client Recommendations

A Client SHOULD:

- Only specify "ipp-attribute-fidelity" = 'true' when unconditional override is required, to avoid unintended side effects, such as output scaling, etc.;
- Use "job-mandatory-attributes" (if available) instead of "ipp-attribute-fidelity";
- Use 'guaranteed' only when "document-format" is specifying a raster document format such as JPEG or PWG Raster;
- Provide document data that is the correct size, etc. to avoid overrides and not depend on the IPP Printer to perform these overrides unless it can guarantee it.

5.5.2 IPP Printer Recommendations

A Printer SHOULD:

- Support "job-mandatory-attributes";
- Support 'guaranteed' for "pdl-override-supported" for raster document formats such as JPEG or PWG Raster.

5.6 Use "multiple-document-handling" to Collate Copies

Although there was a time where specifying multiple copies of a Job would by default produce an uncollated collection of copies, when a User asks for multiple copies the User's expectation is almost always that the Job will produce a collated collection of copies. Despite this User expectation, for historical reasons the print systems have assumed un-collated to be the norm.

The "sheet-collate" attribute [RFC3381] "specifies whether or not the media sheets of each copy of each printed document in a Job are to be in sequence, when multiple copies of the

document are specified by the ’copies’ attribute". The IPP Client likely doesn't want to collate sheets, but rather collate copies of the Job. The "multiple-document-handling" Job Template attribute provides these semantics. Although ([RFC2911] §4.2.4) asserts that the "multiple-document-handling" attribute is relevant only if a Job consists of two or more documents, this attribute provides the desired semantics outside that context as well.

[PWG5100.13] §10 provides a detailed discussion of the relationship between Impressions, Pages and Sheets, and attributes that influence how the various Job Template attributes affect these metrics.

5.6.1 IPP Client Recommendations

A Client SHOULD request collation for multiple copies of a Job using the "multiple-document-handling" attribute with its value set to "separate-documents-collated-copies" if the Printer supports the "multiple-document-handling" attribute.

5.6.2 IPP Printer Recommendations

A Printer SHOULD implement "multiple-document-handling" to allow a Client to request collated copies of a Job.

5.7 Specify "orientation-requested" For Supported Document Types

A Client uses the "orientation-requested" attribute [RFC2911] §4.2.10 to convey the desired orientation of the pages for a Job or Document in a Job. Although some document type formats provide a mechanism to express page orientation, others do not.

5.7.1 IPP Client Recommendations

If "orientation-requested-supported" is in the filtered list of Job Template attributes for the chosen document format, a Client SHOULD provide the "orientation-requested" attribute with the Job or with each Document submitted to an IPP Job.

5.7.2 IPP Printer Recommendations

A Printer SHOULD implement "orientation-requested" for the PDLs it supports that lack a mechanism to specify page orientation.

5.8 Evaluating Printer Capability Attributes

Before creating a Job, a Client SHOULD query the IPP Printer to evaluate its capabilities. Techniques for doing this using different sequences of operations are described in §4.3. When responding to an IPP Get-Printer-Attributes request, the Printer filters the set of attributes and values [RFC2911] to those that are acceptable to a Job Creation, Validate-Job, and Validate- Document operation..

IPP Client implementations SHOULD evaluate the attributes listed in Table 1, which lists the Printer attributes that are discussed in the following subsections.

Table 1: Printer Capability Attributes A Client Should Evaluate

Attribute Name	Reference	Description
document-format-supported	RFC 2911	Lists the set of supported document formats
printer-get-attributes-supported	PWG 5100.13	Lists the set of attributes a Client can send to the Printer in a Get-Printer-Attributes operation request that the Printer can employ to filter the set of Job Template attributes it sends back in its response
document-format-varying-attributes	RFC 3380	Lists the attributes whose defaults or range of supported values vary by document format
ipp-features-supported	PWG 5100.13	Lists the set of optional functionality the Printer supports
printer-config-change-date-time	PWG 5100.13	Helps a Client or operator to determine how recently any of the Printer description attributes has been changed; useful if the Client is caching Printer defaults and capabilities
printer-config-change-time	PWG 5100.13	Helps a Client or operator to determine how recently any of the Printer description attributes has been changed; useful if the Client is caching Printer defaults and capabilities
job-creation-attributes-supported	PWG 5100.11	Lists the keyword names of the Job Template and operation attributes that the Printer will accept in Job Creation operations or a Validate-Job operation
document-creation-attributes-supported	PWG 5100.5	Lists the keyword names of the Document Template and operation attributes that the Printer will accept in Document Creation operations (Send-Document, Send-URI) or a Validate-Document operation

5.8.1 document-format-supported / document-format

One of the most elemental tasks that a Client performs is choosing a print Job document format. The Printer's "document-format-supported" attribute enumerates the document formats supported by the Printer. How the Client chooses the particular document format is beyond the scope of this specification.

Once the Client has chosen a format, it SHOULD send a second Get-Printer-Attributes operation request, including the "document-format" attribute. As is described in §4.4, the Client does this to receive the list of Printer and Job Template attributes filtered to correspond to that specific document format. In this way, the Client can get a set of Printer capabilities the accurately describe the possible options supported by the Printer for that document format, with unsupported attributes and values filtered out [RFC2911]. Further filtering can be done using attributes listed in the Printer's "printer-get-attributes-supported" attribute, described in §5.8.2.

5.8.2 printer-get-attributes-supported

The "printer-get-attributes-supported" attribute [PWG5100.13] enumerates those attributes that can be included in a Get-Printer-Attributes request to influence the attributes and attribute values sent by the Printer in its Get-Printer-Attributes operation response. This attribute is useful to a Client because it allows it to further filter the Printer's capabilities so that the options provided to a user can be more tightly constrained. Ideally a user will be presented as many options as are supported but no more.

If a Printer supports the "printer-get-attributes-supported" attribute, a Client SHOULD include only attributes whose names are listed in that attribute when performing follow-up Get-Printer-Attributes operations as described in §4.4; otherwise, the Printer might reject the operation or the operation response might contain unfiltered information, which the Client would likely not expect.

5.8.3 document-format-varying-attributes

The "document-format-varying-attributes" attribute [RFC3380] is a Printer attribute that lists the Printer attributes that might vary depending on the document format. While this attribute was designed for use by a management Client engaged in manipulating the state of the Printer via Set-Printer-Attributes operations, a Client can use it in the process of submitting a Job to identify those attributes that could have different values depending on the document format.

This attribute is useful to a Client because it provides another mechanism to detect attributes or attribute value ranges that could vary depending on the document format.

5.8.4 ipp-features-supported

The "ipp-features-supported" Printer attribute [PWG5100.13] provides a mechanism for listing support for a particular IPP extension feature. A feature can be comprised of

multiple attributes, can also include new operations, and can have associated semantics defined as well. Examples of these features from [PWG5100.13] include "document-object", "job-save", "page-overrides", "proof-print", and "subscription-object". Other IPP specifications can define their own keywords and associated meanings, which are context-specific. It can be difficult to switch a feature on based solely on the existence of a particular set of attributes or attribute values.

This attribute is useful because it provides a way for a Client to easily enable potentially complex feature support without having to rely on more complicated heuristic analysis and interpolation of the attributes that are used to support the feature.

5.8.5 printer-config-change-date-time and printer-config-change-time

The "printer-config-change-date-time" and "printer-config-change-time" Printer attributes [PWG5100.13] provide 2 different representations of the most recent time at which the printer-config-changed Printer Event occurred.

This attribute is useful because a Client that records and subsequently monitors this attribute will be able to decide whether it needs to re-evaluate the printer's attributes or whether a cached set of values can be used.

5.8.6 xxx-supported and xxx-default

[RFC2911] §4.2 provides a set of rules for specifying how a Job Template attribute is defined. Generally, a Job Template attribute is defined in a cluster of 3, including the attribute itself ("xxx"), an attribute that conveys the range of possible values that "xxx" can contain ("xxx-supported"), and an attribute that conveys the current default value ("xxx-default") that the Printer will use if that attribute isn't provided by the Client at Job submission time.

The "xxx-supported" and "xxx-default" attributes are reported to the Client using a Get-Printer-Attributes operation. These values can change if the Client includes attributes with the Get-Printer-Attributes operation such as "document-format" as discussed in §5.8.1 or one of the attributes enumerated by the "printer-get-attributes-supported" attribute as discussed in §5.8.2.

A Client SHOULD always provide all intended and applicable Job Template attributes explicitly in its operation requests; a Client SHOULD NOT depend on default values, because the default can change before the Job is processed.

5.8.7 xxx-supported vs. xxx-ready

Most Job Template attributes are defined using a cluster of 3 attributes ("xxx", "xxx-supported" and "xxx-default", as discussed in §5.8.6. Some attributes also have a fourth "xxx-ready" attribute in their cluster. This attribute is used to indicate which of the values in the "xxx-supported" attribute are ready for use with no user interaction needed. The attributes that have an associated "xxx-ready" attribute are generally those that involve

consumables such as paper or marking agents. A common example would be the "media-ready" attribute [RFC2911], which indicates the set of media that is physically in the paper trays. This would likely differ from "media-supported" because the set of supported media types is most likely larger than the set of media types currently installed in the paper trays.

This attribute is useful to a Client because it allows the Client to be able to determine whether certain consumables choices are already available for use, and allows the Client to make certain choices as to the workflow it provides to the user.

5.8.8 job-creation-attributes-supported

The "job-creation-attributes-supported" attribute [PWG5100.11] lists the set of Job attributes that can be set by a Client with a Job Creation operation (Create-Job, Print-Job, Print-URI) or a Validate-Job operation. The list of attributes is not limited to Job Template attributes; the list can also include additional Job submission attributes, such as "job-name".

The Client can use this attribute to know which attributes the Printer supports.

5.8.9 document-creation-attributes-supported

The "document-creation-attributes-supported" attribute [PWG5100.5] is similar to the "job-creation-attributes-supported" attribute except it applies to those attributes that can be supplied in Document Creation (Print- Job, Print-URI, Send-Document, and Send-URI) and manipulation (Set-Document-Attributes, Set-Job-Attributes) operations. All recommendations made for the use of "job-creation-attributes-supported" also apply to "document-creation-attributes-supported".

5.8.10 job-settable-attributes-supported

The "job-settable-attributes-supported" attribute [RFC3380] lists the Job object attributes that can have their values changed via a Set-Job-Attributes operation. This is useful to the Client because it provides a mechanism to control some aspects of Job state, which can affect processing time and execution. For instance, a Client can use this attribute to update Job attributes and release a Job in the pending-held state after all documents have been submitted.

5.8.11 media-col-ready vs. media-col-database

The “media-col-database” attribute describes all the pre-defined "media-col" values implemented by the Printer, regardless of whether they are currently available. By definition, this attribute is not sent by the Printer unless explicitly requested since it is large, expensive to generate, and likely contains media that are not readily available, which can be misleading to the user. The “media-col-ready” attribute [PWG5100.3] describes the attributes of the media that is currently ready to use.

The Printer SHOULD implement the “media-col-ready” attribute.

The Client SHOULD use “media-col-ready” if the Printer implements it. The Client SHOULD NOT retrieve the “media-col-database” attribute unless the User does something overt that depends on presenting or using the entire "media-col-database".

5.9 Resolving Job Attribute Conflicts and Constraints

The "job-constraints-supported" and "job-resolvers-supported" attributes [PWG5100.13] provide respectively mechanisms for a Printer to describe to a Client the constraints between sets of attributes and/or attribute values, and associated formulae for resolving constraint conflicts. A Client that supports these attributes properly can resolve conflicts between constrained attributes or attribute values locally, without having to contact the Printer with requests to validate the attribute set correctness. The "preferred-attributes" attribute provides an additional mechanism for a Printer to report attribute conflicts detected during a "Validate-Job" operation. This is covered in more detail in §4.5.

The “job-constraints-supported” attribute definition in [PWG5100.13] §5.6.8 provides this example:

For example, a constraint for duplex printing on transparency media would be encoded as a collection containing “resolver-name”, “sides”, and “media-col” member attributes. The “sides” member attribute would have two values - “two-sided-long-edge” and “two-sided-short-edge” - while the “media-col” member attribute would have a single "media-type" member attribute with the value “transparency”.

The “job-resolvers-supported” attribute definition in [PWG5100.13] §5.6.11 provides this example:

For example, a resolver for duplex printing on transparency media would be encoded as a collection containing “resolver-name”, “sides”, and “media-col” member attributes. The “sides” member attribute would have the value “one-sided” while the “media-col” member attribute would contain a "media-type" member attribute with the value “stationery”.

To illustrate how this works via 2 examples, the Client is assumed to have received the following attributes from the Printer in a Get-Printer-Attributes operation response. The attributes relevant to these examples are listed below, using the Open Standard Print API [PAPI] attribute list text representation syntax:

```
sides-default="one-sided"
media-col-ready=
{
   {
      media-size =
      {
         x-dimension=21590
         y-dimension=27940
      }
```



```
      media-top-margin=423
      media-bottom-margin=423
      media-left-margin=423
      media-right-margin=423
      media-source=tray-3
      media-type="stationery"
      media-source-properties=
      {
         media-source-feed-direction="long-edge-first"
         media-source-feed-orientation=4
      }
   }
   {
      media-size=
      {
         x-dimension=27940
         y-dimension=43180
      }
      media-top-margin=423
      media-bottom-margin=423
      media-left-margin=423
      media-right-margin=423
      media-source="tray-2"
      media-type="transparency"
      media-source-properties=
      {
         media-source-feed-direction="short-edge-first"
         media-source-feed-orientation=3
      }
   }
}

job-constraints-supported=
{
   resolver-name=“A”
   sides=
   {
      "two-sided-long-edge"
      "two-sided-short-edge"
   }
   media-col=
   {
      media-type="transparency"
   }
}
```



```
job-resolvers-supported=
{
   resolver-name=“A”
   sides="one-sided"
   media-col=
   {
      media-type="stationery"
   }
}
```

To illustrate how constraints and resolvers work using the above attribute information, two examples will be described. In each of them, the Client loads the attributes from the Printer and presents a print dialog with print options to the User. The state of "sides" is initialized with the value from "sides-default", and “media-col" is initialized with the value from "media-col-default".

As a first example, the User chooses "sides" = 'two-sided-long-edge'. The Client detects a control change, and evaluates the changed value against the constraints. The Client detects no conflicts, so the Client accepts the attribute value change. Then the User selects "transparency" for a media type. The Client detects a control change and evaluates the changed value against the constraints. The Client detects a conflict, so the Client takes some action to resolve the conflict. The Client implementation determines how the conflict is resolved. It could be that the new value is rejected ("media-col" change in this example), or it could be that the new value is kept but the conflicting attribute is changed as per the resolver. Or the choices could be presented to the User in some way.

For a second example, the User selects "transparency" for a media type. The Client detects a control change and evaluates the changed value against the constraints. The Client detects no conflicts, so the Client accepts the attribute value change. Then the User selects "sides"='two-sided-short-edge'. The Client detects a control change and evaluates the changed value against the constraints. The Client detects a conflict, so the Client takes some action to resolve the conflict. Again, how the Client manages the resolution is at the discretion of the Client implementors.

5.10 IPP Object Status Attributes

A Client will evaluate different sets of attributes when following the procedures in §4.7 or 4.9 to monitor the status of a Printer or Job. These sets SHOULD be interpreted in the context of one another to ensure correct status evaluation.

5.10.1 Printer Status

A Printer conveys its status using the "printer-state", "printer-state-reasons" and "printer-state-message" attributes [RFC2911], as well as the "printer-alert" and "printer-alert-description" [PWG5100.9]. A Client can acquire these attributes using a Get-Printer-

Attributes operation. A Printer SHOULD append to its "printer-state-reasons" attribute values the appropriate suffixes defined in [RFC2911] §4.4.12.

If the Printer and Client both support IPP Notifications, the Client can create a Printer subscription that includes the "printer-state", "printer-state-reasons" and "printer-state-message" attributes in the "notify-attributes" list. Refer to §5.11 for more information on IPP Notifications.

5.10.2 Job Status

A Job conveys its status using the "job-state", "job-state-reasons" and "job-state-message" attributes. A Client can acquire these attributes using a Get-Job-Attributes operation.

If the Printer and Client both support IPP Notifications, the Client can create a Job subscription that includes the "job-state", "job-state-reasons" and "job-state-message" attributes in the "notify-attributes" list. Refer to §5.11 for more information on IPP Notifications.

5.10.3 Document Status

If the Client wants detailed information about the status of Document objects within a Job, it can get the status of the various Documents within the Job using the Get-Document-Attributes operation, and would examine the "document-state", "document-state-reasons" and "document-state-message" attributes.

5.11 IPP Notifications Attributes

A well-implemented IPP Printer SHOULD support IPP Notifications [RFC3995]. The Printer SHOULD include additional Printer attributes in Printer event notifications (e.g. "printer-supply" for supply events, "printer-input-tray" for media events, etc.).

When subscribing for IPP notifications, a Client SHOULD include the "notify-attributes" Subscription Template attribute in its Job Creation operations rather than subscribing to events using the separate Create-Job-Subscriptions or Create-Printer-Subscriptions operations. Having the Job Creation operation create the subscription makes the subscription creation and Job creation operations atomic and reduces the need for additional operations.

A Client uses the "notify-attributes" attribute to list the attributes it wants request that each notification include particular attributes. If each notification includes the attributes the Client needs, the Client will not need to perform a subsequent Get-Printer-Attributes operation for a Printer subscription, or Get-Job-Attributes operation for a Job subscription.

6. IPP Document Page Order

Document formats intended for streaming (printed directly), such as PWG Raster, SHOULD have the pages sent in the order they are to emerge from the printer. This order will be affected in part by which side the marking appears on when the page emerges from the printer. On a Printer that implements the "printer-output-tray" attribute and prints its documents “face down” (the paper emerges with the marking on the bottom side of the page), the pages SHOULD be sent in the order 1-N. On a Printer that prints its pages “face up” (the paper emerges with the marking on the top side of the page), the pages SHOULD be sent in the reverse order (N-1). Other aspects, including duplex and page reordering, will further affect this page ordering. If the Printer does not support the "printer-output-tray" attribute, the "output-bin" and "output-bin-default" attributes can be evaluated. The 'face-up' value indicates last-to-first order; all other values are first-to-last.

Other document formats that are not intended for streaming but are rather intended for more general use, such as PDF or word processing document formats, need to be sent as-is, with no page reordering. It will be the Printer’s responsibility to paginate and print the file in the order that is appropriate given the paper orientation and other built-in information in the document format (with overriding behavior as per §5.5). This could require the entire Document to be received before the first page can be printed. For instance, if a Client submits a Job with its document content in PDF document format, the Client SHOULD send the document format in 1-N page ordering, and depend on the Printer processing the pages in whatever order is appropriate to comply with the preferences in the Job IPP attributes and the settings in the PDF document.

7. IPP Printer Best Practices

This section enumerates practices that SHOULD be followed by implementers of IPP Printer objects, whether embedded in a Printer or in a separate Print Server.

7.1 IPP Objects and URI Resource Paths

IPP operations require a target consisting of a "printer-uri" attribute and zero or more operation identifiers, such as "job-id", "document-number", or others [RFC2911]§3.1.5.

Each instance of a Printer provided by an Imaging Device is identified by a URI. The URI scheme MUST be "ipps" [RFC7472] or "ipp" [RFC3510]. It MUST NOT be "http" or "https" [RFC2910].

The path component of an IPP Printer URI SHOULD be “/ipp/print” for the only (or default) instance of the service on an Imaging System and “/ipp/print/instance-name” for each additional, non-default instance on the Imaging System.

If the Printer supports IPP operations with the "/" resource path, then the Printer SHOULD support Get-Printer-Attributes and SHOULD NOT support Job Creation operations.

[RFC2911] requires particular Client behavior concerning the use of URIs in an operation's "printer-uri" attribute. [RFC2911] § 2.4 requires the Client to use URIs listed in the Printer's "printer-uri-supported" attribute. [RFC2911] § 3.1.5 requires the Client to only use absolute URIs for the value sent for the "printer-uri" operation attribute. Despite the recommendations in [RFC3196], a Printer SHOULD NOT accept a request where the "printer-uri" value is a relative URI and SHOULD NOT accept a request where the "printer-uri" doesn't match one of those URIs in the Printer's "printer-uri-supported" attribute.

A Printer SHOULD reject IPP operations where the value of the "printer-uri" operation attribute is a relative URI. A Client SHOULD NOT use a relative URI in the "printer-uri" operation attribute.

The "job-uri" path has never been fully specified. Printers SHOULD choose a path appropriate for the implementation. Clients SHOULD use the "printer-uri" and "job-id" attributes to target a Job object for a Job operation. Clients SHOULD NOT use the "job-uri" attribute to target a Job object.

7.2 Printer Resource URIs

URIs to various Printer resident resources SHOULD be Network Accessible even if all other network services have been disabled in the Output Device.

As an example, the "printer-icons" URI SHOULD be Network Accessible even if the Output Device has its embedded web server turned off. Since the "printer-icons" attribute is defined to contain a set of "http:" or "https:" URIs, each URI SHOULD include the port number of the IPP Server, e.g. "http://myprinter.mydomain.com:631/icon.png".

7.3 Error Reporting

A Printer SHOULD log errors using the PWG Common Log Format [PWG5110.3].

8. HTTP Protocol Use

IPP currently uses HTTP/1.1 for its transport. IPP/2.0 and other IPP specifications have specified some of the facilities of HTTP that IPP clients and servers SHOULD support in order to provide the semantics that IPP needs to provide a great user experience. Even so, there are best practices that SHOULD be followed.

What follows includes recommendations for both Client and Server implementations. Each subsection will provide Client recommendations in an 8.x.1 subsection; Server recommendations will be listed in an 8.x.2 subsection.

8.1 New HTTP/1.1 Specifications

In June 2014 the IETF published the following specifications, which obsolete RFC 2616:

- Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing [RFC7230]
- Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content [RFC7231]
- Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests [RFC7232]
- Hypertext Transfer Protocol (HTTP/1.1): Range Requests [RFC7233]
- Hypertext Transfer Protocol (HTTP/1.1): Caching [RFC7234]
- Hypertext Transfer Protocol (HTTP/1.1): Authentication [RFC7235]

8.1.1 HTTP Client

Implementors SHOULD review their HTTP stack implementation to ensure conformance with these current HTTP/1.1 specifications.

8.1.2 HTTP Server

Implementors SHOULD review their HTTP stack implementation to ensure conformance with these current HTTP/1.1 specifications.

8.2 HTTP/1.1 Expect Header

The HTTP/1.1 "Expect" header field [RFC7230] allows the HTTP Client to send the HTTP request header to the Server and pause to receive an HTTP response code before sending the request body, if present. This allows a Server to indicate via an appropriate response code if there are any authorization or encryption requirements to be satisfied before sending the HTTP request body.

8.2.1 HTTP Client

In its first request to the HTTP Server, and in all first uses of a particular IPP operation while involved in a session consisting of a sequence of IPP operations, the HTTP Client SHOULD include the "Expect: 100-continue" header in the request headers [RFC7231] §5.1.1. The HTTP Client SHOULD then wait at least 1 second for a response from the HTTP Server. If no response is received, the Client SHOULD record this for subsequent requests so that no further response delays are incurred, and then continue sending the message body; otherwise the HTTP Client SHOULD process the response in one of the following ways (which is a more detailed set of behaviors than described in HTTP/1.1 [RFC7230]):

1. **100 Continue** : Continue sending the message body
2. 301 Moved Permanently or 302 Moved Temporarily:

a. If the request type is POST or PUT, fail and report an error, since redirection is generally unexpected for this request type

b. If the request type is HEAD or GET, redirect the request to the new URI or address the redirection in an implementation-defined way, e.g. request confirmation from the User, validate the redirected URI scheme and origin server, etc.

3. **400 Bad Request**: Record the lack of an Expect header, and re-send the HTTP request without the Expect header. (An HTTP/1.1 Server is supposed to support the Expect header as per [RFC7230] §5.1.1 but a robust Client might include logic for recovering from interacting with poorly implemented servers.)

4. **401 Unauthorized**: Close the connection, acquire new credentials from the User, and retry the HTTP request with the newly acquired credentials

5. **403 Forbidden**: Abandon the connection and report an error as per §4.10.

6. **417 Expectation Failed**: Re-send the HTTP request without the Expect header

7. **426 Upgrade Required**: If the upgrade response header contains a TLS identifier, upgrade the connection to TLS [RFC2817] and then re-send the HTTP request

8. **Any other HTTP status code** : Treat as a fatal error and report it as per §4.10.

8.2.2 HTTP Server

The HTTP Server SHOULD accept and process the "Expect" header as defined in § 5.1.1 of "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content" [RFC7231].

8.3 "Host" Header Field

An HTTP Server validates the Host request header in order to protect against DNS rebinding attacks. Section 5.1.1 of "IPP Everywhere" [PWG5100.14] places additional requirements on the Host header.

8.3.1 HTTP Client

An HTTP Client SHOULD include the Host header field, and its value SHOULD match the name in the URI it is using.

8.3.2 HTTP Server

An HTTP Server SHOULD use the Host value in generated URIs, such as the embedded web server web pages or IPP operation responses.

8.4 If-Modified-Since, Last-Modified, and 304 Not Modified

The If-Modified-Since request header allows an HTTP Client to efficiently determine whether a particular resource that can be retrieved using a GET request (icon, ICC profile, localization file, etc.) has been updated since the last time the HTTP Client requested it.

8.4.1 HTTP Client

An HTTP Client SHOULD utilize the "If-Modified-Since" header in GET requests for resources that are already locally available. An HTTP Client SHOULD support the "Last-Modified" response header and "304 Not Modified" response code, to avoid re-fetching these resources if they are locally available and haven't changed.

8.4.2 HTTP Server

HTTP Servers SHOULD support the "If-Modified-Since" request header [RFC7232], the Last-Modified response header [RFC7232], and the 304 “Not Modified” response code [RFC7232].

8.5 Cache-Control

Most resource files provided by a Printer in a GET response will typically be cacheable. However, IPP responses in a POST response are not [PWG5100.14]. Most resources retrieved by an IPP Client from an IPP Printer via an HTTP GET are usually quite static and ought to be treated as a persistent resource. The HTTP "Cache-Control" header [RFC7234] formalizes this.

8.5.1 HTTP Client

An HTTP Client SHOULD conform to the HTTP/1.1 caching semantics [RFC7234]. An HTTP Client SHOULD support the "Cache-Control" response header.

8.5.2 HTTP Server

An HTTP Server SHOULD conform to the HTTP/1.1 caching semantics [RFC7234]. A Printer's HTTP Server MAY provide a "Cache-Control" header in each HTTP GET response, with an appropriate "max-age" value for that resource. A Printer's HTTP Server SHOULD provide a Cache-Control header in IPP POST responses with the value "no-cache".

9. Important Implementation Options

9.1 Job Receipts: The "xxx-actual" Attributes

The "xxx-actual" [PWG5100.8] attributes are not guaranteed to be set until the Job reaches a terminating state ('aborted', 'canceled' or 'completed'). Processors of the Job

receipt SHOULD NOT examine the "xxx-actual" attributes before the Job has reached a terminating state.

9.2 Printer Constraints: The "preferred-attributes" Attribute

The Printer indicates it supports the "preferred-attributes" attribute [PWG5100.13] with the "preferred-attributes-supported" Printer Description attribute. When supported, the Printer will return the "preferred-attributes" attribute in the Validate-Job response if the Printer will perform substitutions to resolve detected constraints problems. This will provide the Client the opportunity to resolve these issues locally before creating the Job. However, although this is useful when performing Job ticket validation, a Client SHOULD NOT be implemented to use Validate-Job operations and the "preferred-attributes" attribute as a mechanism for resolving constraints conflicts in an interactive UI. As outlined in §4.5, a Client SHOULD start by using the procedures described in §5.9, which can be performed locally on the Client and doesn't require any network utilization.

10. Conformance Recommendations

10.1 Client Conformance Recommendations

Clients SHOULD implement the following:

1. all of the "best" options in section 4;
2. all of the Client recommendations in section 5;
3. all of the recommendations in section 6;
4. all of the Client recommendations in section 8;
5. all of the Client recommendations in section 11;
6. all of the Client recommendations in section 12

10.2 Printer Conformance Recommendations

Printers SHOULD implement the following:

1. all of the "best" options provided in section 4;
2. all of the recommendations in section 5;
3. all of the recommendations in section 7;
4. all of the Printer recommendations in section 8;

5. all of the Printer recommendations in section 11;

6. all of the Printer recommendations in section 12

11. Internationalization Considerations

For interoperability and basic support for multiple languages, conforming implementations MUST support:

1. The Universal Character Set (UCS) Transformation Format -- 8 bit (UTF-8) [STD63] encoding of Unicode [UNICODE] [ISO10646]; and
2. The Unicode Format for Network Interchange [RFC5198] that requires transmission of well-formed UTF-8 strings and recommends transmission of normalized UTF-8 strings in Normalization Form C (NFC) [UAX15].

Unicode NFC is defined as the result of performing Canonical Decomposition (into base characters and combining marks) followed by Canonical Composition (into canonical composed characters wherever Unicode has assigned them).

WARNING – Performing normalization on UTF-8 strings received from IPP Clients and subsequently storing the results (e.g., in IPP Job objects) could cause false negatives in IPP Client searches and failed access (e.g., to IPP Printers with percent-encoded UTF-8 URIs now 'hidden').

11.1 Client Considerations

A Client SHOULD query and use localization catalogs provided by the IPP Printer [PWG5100.13], if available.

11.2 Server Considerations

A Printer SHOULD support localization catalogs [PWG5100.13].

12. Security Considerations

12.1 Client Security Considerations

In addition to the sections below, Client implementors SHOULD also pay attention to §7.2, §8.2.1 and §8.3.

12.1.1 HTTP/1.1 Expect Header

As discussed in §8.2, a Client SHOULD send the HTTP/1.1 Expect header where it is important to determine if the Printer will request to upgrade the connection to use TLS if it isn't being used already. Examples of this scenario include:

a. First HTTP request for a connection
b. First IPP request for a connection
c. Any request where the Client will be transmitting potentially sensitive data

12.1.2 HTTP Upgrade

A Client or Server MAY be implemented to refuse to use certain operations if a connection is not employing TLS encryption. Encryption can be supplied via HTTP Upgrade [RFC2817] or HTTP over TLS (HTTPS) [RFC2818]. Clients and Servers SHOULD support TLS encryption via HTTP Upgrade [RFC2817] and/or HTTPS [RFC2818].

12.1.3 Using The Validate-Job Operation

A Client SHOULD use the Validate-Job IPP operation and the behavior recommendations enumerated in §8.2.1 to test whether the Printer requires authentication or encryption for Job submission.

12.1.4 Preferring The "ipps" URI Scheme

A Client SHOULD support the “ipps” URI scheme. A Client SHOULD prefer the “ipps” URI variant when available. A Client probing a Printer for the first time (§4.1.2 and §4.2) SHOULD first try using IPP OVER HTTPS; if the connection fails the Client MAY then try IPP over HTTP.

12.2 Server Security Considerations

In addition to the sections below, Client implementors SHOULD also pay attention to §7.2, §8.2.1 and §8.3.

12.2.1 HTTP/1.1 Expect Header

As discussed in §8.2, a Server MUST support the HTTP/1.1 Expect header and respond appropriately [RFC7230].

12.2.2 HTTP Upgrade

A Server SHOULD support HTTP Upgrade [RFC2817] to indicate to a Client the need to upgrade the connection to TLS.

12.2.3 Support for The "ipps" URI Scheme

A Printer SHOULD support the "ipps" URI scheme [RFC7472].

If a Printer is configured to only support IPPS (or IPP over HTTPS), and a Client sends an IPP over HTTP request, the Printer MAY do one of the following:

a. Close the connection;

b. Return 301 Moved Permanently;

c. Return 400 Bad Request;

d. Return 426 Upgrade Required to upgrade the connection to TLS

There is no IPP response, just a HTTP response. HTTP requests for embedded web server or resources can be handled in the same way.

12.2.4 DNS Rebinding

Printers SHOULD validate the HTTP Host request header in order to protect against DNS rebinding attacks [PWG5100.14]. The Host header is discussed in §8.3.

13. References

13.1 Informative References

[IANAIPP] Internet Assigned Numbers Authority (IANA) Registries, "Internet Printing Protocol", http://www.iana.org/assignments/ipp-registrations/

[ISO10646] "Information technology -- Universal Coded Character Set (UCS)", ISO/IEC 10646:2011

[PAPI] A. Hlava, N. Jacobs, M. Sweet, "Open Standard Print API (PAPI)", July 2005, http://prdownloads.sourceforge.net/openprinting/PAPI-specification.pdf?download

[PWG5100.1] M. Sweet, "IPP Finishings 2.0", PWG 5100.1-2014, December 2014, http://ftp.pwg.org/pub/pwg/candidates/cs-ippfinishings20-20141219-5100.1.pdf

[PWG5100.3] K. Ocke, T. Hastings, "Internet Printing Protocol (IPP): Production Printing Attributes – Set1", PWG 5100.3-2001, February 2001, http://ftp.pwg.org/pub/pwg/candidates/cs-ippprodprint10-20010212-5100.3.pdf

[PWG5100.5] D. Carney, T. Hastings, P. Zehler. “Internet Printing Protocol (IPP): Document Object”, PWG 5100.5-2003, October 2003, http://ftp.pwg.org/pub/pwg/candidates/cs-ippdocobject10-20031031-5100.5.pdf

[PWG5100.7] T. Hastings, P. Zehler, "Standard for The Internet Printing Protocol (IPP): Job Extensions", PWG 5100.7-2003, October 2003, http://ftp.pwg.org/pub/pwg/candidates/cs-ippjobext10-20031031-5100.7.pdf

[PWG5100.8] D. Carney, H. Lewis, "Internet Printing Protocol: '-actual' Attributes", PWG 5100.8-2003, March 2003, http://ftp.pwg.org/pub/pwg/candidates/cs-ippactuals10-20030313-5100.8.pdf

[PWG5100.9] I. McDonald, C. Whittle, "Internet Printing Protocol (IPP) Printer State Extensions v1.0", PWG 5100.9-2009, July 2009, http://ftp.pwg.org/pub/pwg/candidates/cs-ippstate10-20090731-5100.9.pdf

[PWG5100.11] T. Hastings, D. Fullman, “IPP: Job and Printer Operations – Set 2”, PWG 5100.11-2010, October 2010, http://ftp.pwg.org/pub/pwg/candidates/cs-ippjobprinterext10-20101030-5100.11.pdf

[PWG5100.12] R. Bergman, H. Lewis, I. McDonald, M. Sweet, "IPP/2.0 Second Edition", PWG 5100.12-2011, February 2011, ftp://ftp.pwg.org/pub/pwg/candidates/cs-ipp20-20110214-5100.12.pdf

[PWG5100.13] M. Sweet, I. McDonald, P. Zehler, " IPP: Job and Printer Extensions – Set 3 (JPS3)", 5100.13-2012, July 2012, ftp://ftp.pwg.org/pub/pwg/candidates/cs-ippjobprinterext3v10-20120727-5100.13.pdf

[PWG5100.14] M. Sweet, I. McDonald, A. Mitchell, J. Hutchings, "IPP Everywhere", 5100.14-2013, January 2013, ftp://ftp.pwg.org/pub/pwg/candidates/cs-ippeve10-20130128-5100.14.pdf

[PWG5100.16] M. Sweet, “IPP Transaction-Based Printing Extensions”, PWG 5100.16-2013, November 2013, http://ftp.pwg.org/pub/pwg/candidates/cs-ipptrans10-20131108-5100.16.pdf

[PWG5110.3] M. Sweet, "PWG Common Log Format", PWG 5110.3-2013, April, 2013, http://ftp.pwg.org/pub/pwg/candidates/cs-ids-log10-20130401.5110.3.pdf

[RFC2119] S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels”, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2608] E. Guttman, C. Perkins, J. Veizades, M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999, http://www.ietf.org/rfc/rfc2608.txt

[RFC2817] R. Khare, S Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, September 2000, http://www.ietf.org/rfc/rfc2817.txt

[RFC2818] E. Rescorla, “HTTP Over TLS”, RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

[RFC2910] R. Herriot, S. Butler, P. Moore, R. Tuner, J. Wenn, "Internet Printing Protocol/1.1: Encoding and Transport", RFC 2910, September 2000, http://www.ietf.org/rfc/rfc2910.txt

[RFC2911] R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell, "Internet Printing Protocol/1.1: Model and Semantics", RFC 2911, September 2000, http://www.ietf.org/rfc/rfc2911.txt

[RFC3196] T. Hastings, C. Manros, P. Zehler, C. Kugler, H. Holst, "Internet Printing Protocol/1.1: Implementer's Guide", RFC 3196, November 2001, http://www.ietf.org/rfc/rfc3196.txt

[RFC3380] T. Hastings, R. Herriot, C. Kugler, H. Lewis, "Internet Printing Protocol (IPP): Job and Printer Set Operations", RFC 3380, September 2002, http://www.ietf.org/rfc/rfc3381.txt

[RFC3381] T. Hastings, H. Lewis, R. Bergman, "Internet Printing Protocol (IPP): Job Progress Attributes", RFC 3381, September 2002, http://www.ietf.org/rfc/rfc3381.txt

[RFC3510] R. Herriot, I. McDonald, "Internet Printing Protocol/1.1 IPP URL Scheme", RFC 3510, April 2003, http://www.ietf.org/rfc/rfc3510.txt

[RFC3805] R. Bergman, H. Lewis, I. McDonald, "Printer MIB v2", RFC 3805, June 2004, http://www.ietf.org/rfc/rfc3805.txt

[RFC3806] R. Bergman, H. Lewis, I. McDonald, "Printer Finishing MIB", RFC 3806, June 2004, http://www.ietf.org/rfc/rfc3806.txt

[RFC3966] H. Schulzrinne, "The tel URI for Telephone Numbers", RFC 3966, December 2004, http://www.ietf.org/rfc/rfc3966.txt

[RFC3995] R. Herriot, T. Hastings, "Internet Printing Protocol (IPP): Event Notifications and Subscriptions", RFC 3995, March 2005, http://www.ietf.org/rfc/rfc3995.txt

[RFC3996] R. Herriot, T. Hastings, H. Lewis, "Internet Printing Protocol (IPP): The ’ippget’ Delivery Method for Event Notifications", RFC 3996, March 2005, http://www.ietf.org/rfc/rfc3996.txt

[RFC3997] R. Herriot, T. Hastings, "Internet Printing Protocol (IPP): Requirements for IPP Notifications", RFC 3997, March 2005, http://www.ietf.org/rfc/rfc3997.txt

[RFC4510] K. Zeilenga, Ed., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, June 2006, http://www.ietf.org/rfc/rfc4510.txt

[RFC5198] J. Klensin, M. Padlipsky, " Unicode Format for Network Interchange" RFC 5198, March 2008, http://www.ietf.org/rfc/rfc5198.txt

[RFC6762] S. Cheshire, M. Krochmal, "Multicast DNS", RFC 6762, February 2013, http://www.ietf.org/rfc/rfc6762.txt

[RFC6763] S. Cheshire, M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013, http://www.ietf.org/rfc/rfc6763.txt

[RFC7230] R. Fielding, J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014, http://www.ietf.org/rfc/rfc7230.txt

[RFC7231] R. Fielding, J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014, http://www.ietf.org/rfc/rfc7231.txt

[RFC7232] R. Fielding, J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, June 2014, http://www.ietf.org/rfc/rfc7232.txt

[RFC7233] R. Fielding, Y. Lafon, J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, June 2014, http://www.ietf.org/rfc/rfc7233.txt

[RFC7234] R. Fielding, M. Nottingham, J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014, http://www.ietf.org/rfc/rfc7234.txt

[RFC7235] R. Fielding, J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, June 2014, http://www.ietf.org/rfc/rfc7235.txt

[RFC7472] I. McDonald, M. Sweet, "IPP over HTTPS Transport Binding and 'ipps' URI Scheme", RFC 7472, March 2015, http://www.ietf.org/rfc/rfc7472.txt

[RFC7540] Belshe, M., Peon, R., and M. Thompson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, May 2015, http://www.ietf.org/rfc/rfc7540.txt

[WSDiscovery-1.1] OASIS, "OASIS Web Services Dynamic Discovery (WS-Discovery) Version 1.1", July 2009, http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html

14. Annex A: HTTP/2

HTTP/2 was ratified by the IETF as RFC 7540 in 2015. HTTP/1.1 [RFC7230] is the REQUIRED transport layer for IPP. HTTP/2 [RFC7540] is an OPTIONAL transport layer for IPP.

15. Authors' Addresses

Primary author:

Smith Kennedy
HP Inc.
11311 Chinden Blvd. MS 506
Boise ID 83714
smith.kennedy@hp.com

The author would like to thank the entire PWG IPP Working Group for their help, and would like to particularly thank the following individuals for their patience and many contributions:

Mike Sweet – Apple Inc.

Ira McDonald – High North, Inc.

William Wagner – TIC

