March 20, 2013
Technical Whitepaper

The Printer Working Group

IPP Client Use Best Practices

Status: Interim

Abstract: This document enumerates a number of tasks that are commonly performed by a client in the process of interacting with a print service, and explores options for how the Internet Printing Protocol (IPP) may be used to perform those tasks, some of which are preferred and others that are less than optimal.

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

This document is available electronically at:

ftp://ftp.pwg.org/pub/pwg/general/templates/tb-ipp-best-practices-20130205.pdf

Title: *IPP Client Use Best Practices*

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

The use case descriptions below represent stages or sub-tasks that users perform in the process of using a printer. Each of these below include a textual description as well as a series of workflow options for how it might be implemented using IPP. Each workflow option will be informally labeled according to its perceived quality, using the set of labels {"BAD", "POOR", "GOOD", "BETTER", "BEST"}, that are ordered from least desirable to most desirable.

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 term CONDITIONALLY REQUIRED is additionally defined for a conformance requirement that applies to a particular capability or feature.

2.2 Other Terminology

Capitalized Term In Italics: definition of the term with any references as appropriate.

2.3 Acronyms and Organizations

IANA: Internet Assigned Numbers Authority, http://www.iana.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 would benefit from an updated best practices document that covers the use of these extensions, as well as the core of IPP that has remained unchanged, to assist implementers in their efforts to deliver a quality client experience.

3.2 Use Cases

3.2.1 Developer Implementing New IPP Client Support

Garrett is a developer working on a new client platform that is adding system-level printing support. Many printers support IPP Everywhere [PWG5100.14], so he plans to implement printing support in his client platform using this standard as well. But IPP Everywhere and its related standards don't describe how best to use IPP for the various tasks his software must perform, in order to deliver a quality client user experience. He finds RFC 3196 but its recommendations are insufficient. Using the IPP Use Best Practices document, he is able to avoid some common design pitfalls and quickly deliver a quality IPP client experience.

3.2.2 Developer Implementing New IPP Printer Support

Duncan is a firmware developer at a printer vendor creating a new printer that implements IPP Everywhere. In reading the IPP Client Use Best Practices, he can more easily 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 considered out of scope for this specification:

1. Specifications to extend or replace portions of the Internet Printing Protocol itself
2. Normative requirements regarding user experience

3.4 Design Requirements

The design requirements for this specification are:

178 1. Explore tasks performed by client implementations
179 2. Enumerate a series of alternatives
180 3. Rank those options according to a non-numeric qualitative grading scheme

181 4. Tasks and Implementation Alternatives

182 For a number of tasks, the set of IPP operations provides a rich enough set of semantics
183 that it is possible to perform those tasks in a few different ways. In this section a number
184 of common tasks will be enumerated, and some alternatives for how those tasks might be
185 performed will be evaluated.

186 4.1 Create A Relationship With A Printer

187 You can't print to a printer if you cannot establish a connection to it. Historically,
188 connecting to a printer to establish a "relationship" with it meant identifying a printer and
189 then creating a persistent local records and resources for that printer relationship with your
190 system's print spooler. This was called a "print queue", and it involved binding drivers to
191 create the relationships needed to communicate at the different levels, and then keeping
192 record of that set of relationships so that it could be re-used at a later time. The set of
193 printers or other devices the user's system might encounter was relatively small and fairly
194 static.

195 More recent re-thinking of this relationship between client and printer has resulted in more
196 "dynamic" relationship creation, where universal drivers can interrogate a device hosting a
197 print service using a standardized protocol solution stack, and using that dynamically
198 ascertain and update print service attributes. In this paradigm, a "persistent" print service
199 record is more like a Web browser bookmark.

200 Both paradigms still require a method of identifying the target devices. That can be done
201 using dynamic service discovery protocols where the services respond to discovery
202 requests, or explicitly by name (host name or raw IPv4/IPv6 address).

203 4.1.1 Discover And Select A Printer Via A Discovery Protocol

204 Discovery protocols are used to identify instances of print services or printers by searching
205 the network for service types or device types. This helps the user by making it so that they
206 don't need to do a physical survey of devices' addresses.

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

Options

- POOR:
  - Perform network discovery with a synchronous API
    - Show progress bar
    - Discovery.Start()
    - sleep(X) where X is some reasonably short number of seconds
    - Discovery.Stop()
  - Present the results of the discovery process
  - "Refresh" button restarts the process
    - Why this is bad:
      - List contents can be stale
      - Results are not "live"
      - "Reset" button is unnecessary and is a crutch
  - User selects a printer and presses "Continue" or equivalent
- BETTER:
  - Perform network discovery with an asynchronous API
    - Show List UI widget
    - Discovery.Start() with a callback
    - Callback is called when discovery responses (add or remove) are received
  - User selects a printer and presses "Continue" or equivalent
    - Discovery.Stop()

4.1.2 Select A Printer Via User Provided DNS Hostname Or Raw Ipv4 / Ipv6 Address

In some cases a discovery protocol is either not adequate or unnecessary. Examples of when this use case is encountered include pre-published names or addresses, and also situations where the target device is not on the local link. (DNS-SD and WS-Discovery are generally used for link-local discovery, though wide-area variants as well as LDAP systems may also be used, but are frequently not for various reasons.)

For each of these options below, the assumption is that the client has been given an address string, and should attempt to connect to the host at that address.

Options

- BAD:
  - Let each printer model make up its own path, and depend on some other protocol to get the resource path
    - IPP has no defined standard mechanism to enumerate the Printer objects' resource paths
- POOR:
  - IPP Get-Printer-Attributes with printer-uri set to a URI that was manually entered by the user

252 - The "ipp" URI scheme could be used to encode the hostname and the
253 resource path
254 - Having the user enter the URI exposes too many details to the user,
255 including the detail about the fact that IPP is actually being used.
256 Users need not be aware of which print protocol is being used.
257 • GOOD:
258 ○ IPP Get-Printer-Attributes with printer-uri set to a well-known Printer resource
259 path
260 - "/ipp/print"
261 • BETTER:
262 ○ IPP Get-Printer-Attributes with printer-uri set to "/"
263 ○ Examine the "printer-uri-supported" attribute; use the first URI in the list
264 ○ IPP Get-Printer-Attributes with printer-uri set to first URI
265 • BEST:
266 ○ IPP Get-Services operation
267 - Coming with System Control Service
268 - Is this really going to be better?
269 • Yes, expected to have metadata associated with each URI
270 specifying the class of service

271 ## 4.2 Validate User Access to Printer

272 Selecting a printer is misleading to the user if the user isn't allowed to use the selected
273 printer. Therefore, access restrictions should be validated before selection confirmation
274 (queue creation, etc.) is done on the client system.

275 **Options**

276 • BAD:
277 ○ Do Nothing
278 - The user may choose a printer but not be able to use it due to not
279 having access credentials (username or password or whatever) to use
280 that printer
281 • GOOD:
282 ○ IPP Validate-Job operation
283 - Use the defaults, but provide the credentials to allow the user access
284 to be determined

285 ## 4.3 Get Printer Options

286 Once the user has selected a printer, it is necessary for the print system to understand the
287 capabilities that the printer device's print service provides. This includes what print job
288 payload formats can be consumed by the print service, the available options and default
289 choices, and so forth. It also includes other information about the device itself, such as its

location. Some of this is done at relationship creation time (queue creation time), perhaps by consulting information stored statically in the printer. It may be that this information can all be retrieved from the printer itself. This is basically the print dialog's activity between the time that the user performs an action to request that the print dialog be presented, and the time that the dialog is presented to the user, populated with the available option choices.

Options

- SAD:
  - Depend on a-priori knowledge about a particular model as a way of listing options for the model of device identified as the target
    - Model specific print drivers fall in this bucket
- GOOD:
  - IPP Get-Printer-Attributes Operation
    - Request includes no printer attributes; only operation attributes
    - Reply will contain the job template attributes for all PDLs
  - Client guesses at what attributes may work or not work for a given PDL, or uses a-priori knowledge
- BETTER:
  - IPP Get-Printer-Attributes Operation
    - Any specific attributes?
  - Process results; decide on a PDL
  - IPP Get-Printer-Attributes Operation
    - Request includes the document-format attribute with value specifying the chosen PDL
    - Reply will contain the job template attributes appropriately filtered ("colored") for that particular document-format

4.4 Check constraints between presented 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, and they must be calculated any time the state of a control changes state. There are various ways that this can be done.

Options

- POOR:
  - IPP Validate-Job
    - Every time a control is changed, the client sends IPP Validate-Job with attribute values corresponding to current state of controls
- GOOD:

328 - IPP Validate-Job
329 - When "Print" button is pressed, confirms the job creation / submission
330 will succeed (authentication, etc.)
331 - Client depends on this operation to perform constraints validation
332 printer-side
333 • BETTER:
334 - IPP Get-Printer-Attributes
335 - Printer Object implements job-constraints-supported
336 - Printer Object implements job-resolvers-supported
337 - <Local processing of constraints>
338 - IPP Validate-Job
339 - When "Print" button is pressed, confirms the job creation / submission
340 will succeed (authentication, etc.)
341 - Constraints validation already handled client-side

342 ## 4.5 Submitting a Print Job

343 Once the user has decided on options, the print job is generated and ultimately made
344 available to the printer in some fashion. There are several different ways that this may
345 occur.

346 ### 4.5.1 Submitting a print job with document data

347 This is the classical way that a print job is sent from the client to the print service: first a job
348 is created, and then the job information and payload content are sent from the client to the
349 print service.

350 Options

351 • POOR:
352 - IPP Print-Job
353 - No pre-flight checks
354 - The printer may reject it but only after it has been transmitted.
355 - Better to check ticket and content types first.
356 • GOOD:
357 - IPP Validate-Job
358 - Pre-flight checks the job by validating the job attributes, document
359 type, authentication and transport encryption upgrades (if needed)
360 - IPP Print-Job
361 - Creates the job and sends the payload in one operation
362 - However, the Job object's URI isn't usually known until the job
363 transmission is complete
364 - Doesn't work well with flow-controlled (low-end) printers
365 • BETTER:
366 - IPP Validate-Job

367 ▪ Pre-flight checks the job by validating the job attributes and document
368 type, authentication and transport encryption upgrades (if needed)
369 o IPP Create-Job
370 ▪ Returns immediately with the job URI for monitoring and ticket
371 processing status
372 ▪ If there is a problem then Create-Job may fail the same as Validate-
373 Job would, but may not, which is why we do a Validate-Job first (so
374 that there isn't a zombie job there)
375 ▪ Once the job is created, the client will receive a list of the actual job
376 processing attributes from the IPP Printer.
377 ▪ Response to this operation will include xxx-actual job attributes that
378 could be used to detect substitutions that would be used by the Printer
379 Object. Observing this, the client may decide whether to cancel the
380 job rather than submit the document with this job. If the original job
381 was cancelled, the client could create another job with a new set of
382 attributes submitted, or error out and not submit a job at all
383 ▪ Allows an opportunity to perform a Cancel-Job operation during
384 document submission
385 o IPP Send-Document
386 ▪ Payload transmission is de-coupled from the creation of the job
387 ▪ Multiple documents can be sent to build up a compound job
388 ▪ Client MUST check to see if value of "multiple-document-jobs-
389 supported" is "true", to see if it is OK to do multiple Send-Document
390 operations to the same Job object.

391 **4.5.2 Submitting a print job with document references**

392 This is a slightly different way that a print job is sent from the client to the print service: a
393 job is created and made available for retrieval by the print service, and when the print job
394 the job information and job payload content are sent by the client to the print service.

395 **Options**

396 • POOR:
397 o IPP Print-URI
398 ▪ No pre-flight checks
399 ▪ Printer may reject it but only after it has been transmitted
400 ▪ Better to check ticket and content types first
401 • GOOD:
402 o IPP Validate-Job
403 ▪ Pre-flight checks the job by validating the job attributes and document
404 type
405 o IPP Print-URI
406 ▪ Creates the job and sends a URL to where the payload can be
407 retrieved in one operation

408 - Printer Object "pulls" the document file rather than being given it by
409 the client
410 - However, the Job object's URI isn't usually known until the job
411 transmission is complete
412 - Printer may respond with client-error-document-access-error status
413 code, or might add document-access-error to job-state-reasons
414 - URI may not be accessible at time of processing
415 • BETTER:
416 ○ IPP Validate-Job
417 - Pre-flight checks the job by validating the job attributes and document
418 type
419 ○ IPP Create-Job
420 - Returns immediately with the job URI for monitoring and ticket
421 processing status
422 - If there is a problem then Create-Job will fail the same as Validate-Job
423 would
424 ○ IPP Send-URI
425 - Payload URI transmission is de-coupled from the creation of the job
426 - Printer may respond with client-error-document-access-error status
427 code, or might add document-access-error to job-state-reasons
428 - URI may not be accessible at time of processing
429 • (How to handle this appropriately? What recommendations
430 should be provided?)

431 ## 4.6 Monitoring print job status

432 While the print job is being processed, users may wish to know whether it is proceeding
433 successfully, or whether there are conditions that they need to handle that are preventing
434 processing from proceeding, such as a media jam, open covers, marking agents depleted,
435 and so forth.

436 For those options below that involve polling the Printer Object, the degree to which the
437 option is better or worse is due in no small part to the polling frequency. The interval
438 should be tuned so that the frequency of queries is not so great that it burdens the Printer
439 Object or Job Object or the network, but not so small that there is an undesirable lag
440 between when an event occurs and when the user is notified. It is certainly NOT a best
441 practice in any case if a client is polling as fast as the network can handle traffic.

442 **Options**

443 • POOR:
444 ○ IPP Get-Jobs / IPP Get-Printer-Attributes
445 - Monitor the value of the printer-state attribute and the state of all jobs
446 - Not precise; polling for status without knowing the actual job ID
447 - Polling is generally not desirable

448 - See above regarding polling intervals
449 - GOOD:
450 - IPP Get-Job-Attributes / IPP Get-Printer-Attributes
451 - Monitor the value of printer-state attribute as well as targeted
452 monitoring of a specific job's status
453 - Polling is generally not desirable
454 - See above regarding polling intervals
455 - BETTER:
456 - IPP Create-Printer-Subscriptions / IPP Get-Notifications / IPP Get-Job-
457 Attributes
458 - Asynchronous / long running queries for notifications that don't require
459 polling
460 - When you see that a job has completed, query the state of that job at
461 that time
462 - Printer state changes will be provided by subscribing to the printer;
463 subscribing to the job will provide less information and not be as
464 useful

465 4.7 Canceling a Print Job

466 It may be that the user wants to terminate a job before it has been fully processed, for
467 whatever reason. There are things that must be done to ensure that the client has
468 decisively cleaned up the state of the Job Object if the client is responsible for canceling
469 the job. Clients' leaving broken Job objects on the Print service is bad behavior.

470 There is also a dependency between the options below and how the job was submitted.

471 **Options**

472 - BAD:
473 - IPP Print-Job operation
474 - Client stops sending chunks
475 - POOR:
476 - IPP Print-Job operation
477 - Client stops sending chunks
478 - IPP Cancel-Job operation request for the job via a second connection, which
479 for some printers could result in a PDL interpreter hang because the last
480 chunk sent didn't stop on a "statement" boundary
481 - GOOD:
482 - IPP Create-Job operation
483 - IPP Send-Document operation
484 - Potentially truncating job during Send-Document payload transmission
485 - IPP Cancel-Job operation

4.8 Getting printer supplies status

Some administrative tasks, like checking consumables levels, are presented to end users in some cases, such as during print job status or in print dialogs. This is useful to end-users and should be supported.

Options

- POOR:
  - Don't use IPP but use some proprietary protocol or platform-specific extension to IPP
    - The point is to use only IPP extensions based on open standards (i.e. PWG standard) and this violates that core principle
- GOOD:
  - IPP Get-Printer-Attributes
    - Printer must implement JPS3 "printer-supply" attribute
- BETTER
  - IPP Create-Printer-Subscription operation + IPP Get-Notifications operation
  - IPP Get-Printer-Attributes operations

5. Attributes and Their Use in Operations

Some attributes that IPP has labeled as optional should always be used as a best practice. Below are some of these attributes and how they should be used in various contexts.

5.1 Explicit "document-format" Selection

While IPP Printer Objects provide a default document format (which is known via the document-format-default attribute), as a general principle, it is much better for a client to explicitly provide the document-format attribute with all operations relating to validating or submitting a document payload to the printer (Validate-Job, Print-Job, Send-Document).

5.2 Prefer "media-col" Attribute To "media" Attribute

Given a Printer Object that supports both "media" and "media-col" attributes, a client should prefer to include the "media-col" attribute with operations that accept one of these attributes. This is true for when "media" and "media-col" are top-level attributes as well as when "media" or "media-col" may be included within other collection attributes, such as "job-sheets", "job-error-sheet", "job-accounting-sheets", and others.

5.3 Prefer "finishings-col" Attribute To "finishings" Attribute

Given a Printer Object that supports both "finishings" and "finishings-col" attributes, a client should prefer to include the "finishings-col" attribute with operations that accept one of these attributes.

5.4 Using "ipp-attribute-fidelity"

TBD

5.5 Using "pdl-override"

TBD

6. HTTP Protocol Usage

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.

6.1 HTTP/1.1 Expect Header

As defined in [RFC 2616 "HTTP/1.1"], the "Expect" header allows the client to check with the server on the HTTP connection negotiation before sending the HTTP request payload.

The IPP client should implement the following:

- On first request to a printer, include the "Expect: 100-continue" header.
- Wait up to 1 second for a response.
- If no response is received, remember this for the next request so that you don't have the 1-second delay; continue sending the request.
- If a 100 (continue) status code is returned, continue sending the request
- If a 301 (moved permanently) or 302 (moved temporarily) status code is returned, redirect the request to the new URI *or* fail/report an error depending on the security requirements of the Client (redirection is generally unexpected)
- If a 400 (Bad Request) status code is returned, remember this (don't use Expect header) and re-send the POST request. This Printer is technically non-conforming since it fails RFC 2616 requirements for a HTTP/1.1 server.

- If a 401 status code is returned, re-send the POST request with the requested credentials.
- If a 403 status code is returned, fail/report an error.
- If a 426 status code is returned, send an OPTIONS * request to upgrade to TLS, then re-send the POST request.

The IPP server should implement the following:

- Return status code 403 for unauthorized client addresses when the HTTP level authentication or authorization is not adequate
- Return status code 200 with an IPP response containing the client-error-not-authorized status code when the IPP level authentication or authorization is not adequate
- Status codes 301 and 302 are not recommended
- Return status code 400 only if problems are detected with the HTTP request itself
- Return status code 200 with an IPP response containing the client-error-bad-request status code if problems are detected with the IPP operation

7. Security Considerations

TBD

- What you might do to ensure that the documents submitted remain private
- Using the [IPPS URI]

8. References

8.1 Informative References

[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

[PWG5100.14] F. Last author list or standards body, "IPP Everywhere", 5100.14-2013, January 2013, ftp://ftp.pwg.org/pub/pwg/candidates/cs-ippeve10-20130128-5100.14.pdf

[RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, " Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.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

[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

[REFERENCE] F. Last author list or standards body, "Title of referenced document", Document Number, Month YYYY, URL (if any)

9. Authors' Addresses

Primary authors:

Smith Kennedy
Hewlett-Packard Co.
11311 Chinden Blvd. MS 506
Boise, ID 83714
smith.kennedy@hp.com

The authors would also like to thank the following individuals for their contributions to this white paper:

Evan Williams - Kentucky State Board of Recreation

10. Change History

10.1 February 5, 2013

Initial revision.

10.2 March 20, 2013

Resolved issues from feedback provided during the IPP conference call on February 25, 2013, as documented in teleconference meeting minutes and author's own notes.

1. Added Validate-Job operation as operation to be used during printer selection process to validate access by client / user

2. Replaced previous Section 5 "Conformance Requirements" with new Section 5 "Attributes and Their Use in Operations"

3. Replaced previous Section 6 "Internationalization Considerations" with new Section 6 "HTTP Protocol Usage"

4. Added updated list of references

