February 5, 2013
Working Draft

The Printer Working Group

IPP Client Use Best Practices

Status: Initial

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 6
2. Terminology 6
2.1 Conformance Terminology 6
2.2 Other Terminology 6
2.3 Acronyms and Organizations 6
3. Requirements 7
3.1 Rationale 7
3.2 Use Cases 7
3.2.1 Developer Implementing New IPP Client Support 7
3.2.2 Developer Implementing New IPP Printer Support 7
3.3 Out of Scope 7
3.4 Design Requirements 7
4. Tasks and Implementation Alternatives 8
4.1 Create A Relationship With A Printer 8
4.1.1 Discover And Select A Printer Via A Discovery Protocol 8
4.1.2 Select A Printer Via User Provided DNS Hostname Or Raw Ipv4 / Ipv6 Address 9
4.2 Validate User Access to Printer 10
4.3 Get Printer Options 10
4.4 Check constraints between presented options 11
4.5 Submitting a Print Job 12
4.5.1 Submitting a print job "by value" 12
4.5.2 Submitting a print job "by reference" 13
4.6 Monitoring print job status 14
4.7 Canceling a Print Job 14
4.8 Getting printer supplies status 15
5. Conformance Requirements 16
6. Internationalization Considerations 16
7. Security Considerations 16
8. IANA Considerations 16
9. References 16
9.1 Normative References 16
9.2 Informative References 16
10. Authors' Addresses 16
11. Change History 17
11.1 February 5, 2013 17

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:

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

4. Tasks and Implementation Alternatives

4.1 Create A Relationship With A Printer

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

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

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

4.1.1 Discover And Select A Printer Via A Discovery Protocol

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

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.

Options

- POOR:
  - Perform network discovery with a synchronous API
    - Show progress bar

208 ▪ Discovery.Start()
209 ▪ sleep(X) where X
210 ▪ Discovery.Stop()
211 o Present the results of the discovery
212 o "Refresh" button restarts process
213 ▪ Why this is bad:
214 • list can be stale
215 • The results are not "live"
216 • "Reset" button is unnecessary and is a crutch
217 o User selects a printer and presses "Continue" or equivalent
218 • BETTER:
219 o Perform network discovery with an asynchronous API
220 ▪ Show List UI widget
221 ▪ Discovery.Start() with a callback
222 ▪ callback is called when discovery responses (add or remove) are
223 received
224 o User selects a printer and presses "Continue" or equivalent
225 ▪ Discovery.Stop()

226 **4.1.2 Select A Printer Via User Provided DNS Hostname Or Raw Ipv4 / Ipv6 Address**

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

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

234 **Options**

235 • BAD:
236 o Let each printer model make up its own path, and depend on some other
237 protocol to get the resource path
238 ▪ IPP has no defined standard mechanism to enumerate the Printer
239 objects' resource paths
240 • POOR:
241 o IPP Get-Printer-Attributes with printer-uri set to a URI that was manually
242 entered by the user
243 ▪ "ipp" URI scheme could be used to encode the hostname and the
244 resource path
245 ▪ Having the user enter the URI exposes too many details to the user,
246 including the detail about the fact that IPP is actually beiing used.
247 Users need not be aware of which print protocol is being used.

- GOOD:
  - IPP Get-Printer-Attributes with printer-uri set to a well-known Printer resource path
    - "/ipp/print"
- BETTER:
  - IPP Get-Printer-Attributes with printer-uri set to "/"
  - examine the "printer-uri-supported" attribute; use the first URI in the list
  - IPP Get-Printer-Attributes with printer-uri set to first URI
- BEST:
  - IPP Get-Services operation
    - Coming with System Control Service
    - Is this really going to be better?
      - Yes, expected to have metadata associated with each URI specifying the class of service

4.2 Validate User Access to Printer

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

Options

- BAD:
  - Do Nothing
    - The user may choose a printer but not be able to use it due to not having access credentials (username or password or whatever) to use that printer
- GOOD:
  - IPP Validate-Job operation
    - Use the defaults, but provide the credentials to allow the user access to be determined

4.3 Get Printer Options

Once a printer has been identified, it is necessary for the print system to understand the capabilities that the printer device's print service provides. This includes what print job payload formats can be consumed by the print service, the available options and default 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:
  - IPP Validate-Job
    - when "Print" button is pressed, confirms the job creation / submission will succeed (authentication, etc.)

322 - client depends on this operation to perform constraints validation
323 printer-side
324 • BETTER:
325 o IPP Get-Printer-Attributes
326 - device implements job-constraits-supported
327 - device implements job-resolvers-supported
328 o <Local processing of constraints>
329 o IPP Validate-Job
330 - when "Print" button is pressed, confirms the job creation / submission
331 will succeed (authentication, etc.)
332 - constraints validation already handled client-side

333 ## 4.5 Submitting a Print Job

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

337 ### 4.5.1 Submitting a print job "by value"

338 This is the classical way that a print job is sent from the client to the print service: a job is
339 created; and the job information and payload content are sent by the client to the print
340 service.

341 Options

342 • POOR:
343 o IPP Print-Job
344 - no pre-flight checks
345 - the printer may reject it but only after it has been transmitted in whole
346 or in part.
347 - better to check ticket and content types first.
348 • GOOD:
349 o IPP Validate-Job
350 - pre-flight checks the job by validating the job attributes and document
351 type
352 o IPP Print-Job
353 - creates the job and sends the payload in one operation
354 - however, the Job object's URI isn't usually known until the job
355 transmission is complete
356 - doesn't work well with flow-controlled (low-end) printers
357 • BETTER:
358 o IPP Validate-Job
359 - pre-flight checks the job by validating the job attributes and document
360 type
361 o IPP Create-Job

362 - returns immediately with the job URI for monitoring and ticket
363 processing status
364 - if there is a problem then Create-Job may fail the same as Validate-
365 Job would, but may not, which is why we do a Validate-Job first (so
366 that there isn't a zombie job there)
367 - IPP Send-Document
368 - payload transmission is de-coupled from the creation of the job
369 - multiple documents can be sent to build up a compound job
370 - means that the client doesn't have to be prepared for an early HTTP
371 response
372 - allows the job URI to be learned before job payload is sent
373 - client MUST check to see if value of "multiple-document-jobs-
374 supported" is "true", to see if it is OK to do multiple Send-Document
375 operations to the same Job object.

376 **4.5.2 Submitting a print job "by reference"**

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

380 **Options**

381 • POOR:
382 - IPP Print-URI
383 - no pre-flight checks
384 - the printer may reject it but only after it has been transmitted in whole
385 or in part.
386 - better to check ticket and content types first.
387 • GOOD:
388 - IPP Validate-Job
389 - pre-flight checks the job by validating the job attributes and document
390 type
391 - IPP Print-URI
392 - creates the job and sends a URL to where the payload can be
393 retrieved in one operation
394 - print service retrieves the payload file itself
395 - however, the Job object's URI isn't usually known until the job
396 transmission is complete
397 - doesn't work well with flow-controlled (low-end) printers
398 • BETTER:
399 - IPP Validate-Job
400 - pre-flight checks the job by validating the job attributes and document
401 type
402 - IPP Create-Job

403 - returns immediately with the job URI for monitoring and ticket
404 processing status
405 - if there is a problem then Create-Job will fail the same as Validate-Job
406 would
407 - IPP Send-URI
408 - payload URI transmission is de-coupled from the creation of the job
409 - means that the client doesn't have to be prepared for an early HTTP
410 response
411 - allows the job URI to be learned before job payload is sent

412 ## 4.6 Monitoring print job status

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

417 **Options**

418 • POOR:
419 - IPP Get-Printer-Attributes
420 - monitor the value of the printer-state attribute
421 - polling (lame)
422 • GOOD:
423 - IPP Get-Job-Attributes
424 - targets the specific job status
425 - may be more precise
426 - (need to see if job attributes will show printer issues (top cover open,
427 etc.) that block a job as well)
428 - polling (lame)
429 • BETTER:
430 - IPP Create-Job-Subscriptions
431 - asynchronous / long running queries for notifications that don't require
432 polling

433 ## 4.7 Canceling a Print Job

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

438 There is also a dependency between the options below and how the job was submitted. If
439 Print-Job is used, but the document payload is not completely transmitted, then is a Job
440 object even created? (Is this true in all cases? It is - provide a cross-reference.) Also, if

Create-Job / Send-Document is used and the Cancel-Job is sent during the Send-Document operation submission, then the job object would still need to be cleaned up by the client that created the Job Object. (Is this true? Provide a cross reference.)

Options

- BAD:
  - Client stop sending data and close the connection
    - *Problem*: The IPP Job Object may have been created and still exist, and be in an indeterminate state. It should be explicitly cleaned up by the client for best performance and correctness.
    - *Question*: Are there any realistic or theoretical conditions under which a Print-Job operation does NOT create a Job object?
- POOR:
  - Client stops sending chunks but doesn't close the connection because it is expecting an IPP operation reply.
  - IPP Cancel-Job operation request for the job via a second connection, which for some printers could result in a PDL interpreter hang because the last chunk sent didn't stop on a "statement" boundary
    - But if the client stops with a zero length chunk then the IPP stack will know that transport is complete
    - Need to define a "magic chunk" that operates kind of like an in-band inline Cancel-Job operation, to tell the PDL interpreter that no more job payload chunks?
- GOOD:
  - IPP Cancel-Job
  - Client stops sending chunks and closes the connection
- BETTER:
  - ???

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

5. Conformance Requirements

TBD.

6. Internationalization Considerations

For interoperability and basic support for multiple languages, conforming implementations MUST support the Universal Character Set (UCS) Transformation Format -- 8 bit (UTF-8) [RFC3629] encoding of Unicode [UNICODE] [ISO10646] and the Unicode Format for Network Interchange [RFC5198].

7. Security Considerations

TBD.

8. IANA Considerations

No IANA registrations were necessary for this document to be authored.

9. References

9.1 Normative References

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119/BCP 14, March 1997, http://www.ietf.org/rfc/rfc2119.txt

9.2 Informative References

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

10. Authors' Addresses

Primary authors:

Smith Kennedy

Hewlett-Packard Co.
11311 Chinden Blvd. MS 506
Boise, ID 83702
smith.kennedy@hp.com

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

Evan Williams - Kentucky State Board of Recreation

11. Change History

11.1 February 5, 2013

Initial revision.

