IPP Implementor's Guide

Editor: Carl-Uno Manros
Version: 1.0
Date: August 17, 1998

This document is prepared by the Printer Working Group (PWG), in accordance with the editing rules that apply to PWG documents. The information in this document will be continuously updated and replaced as decided in the meetings of the PWG. The document is made freely available also to non-members of the PWG, but no guarantee is given that the content of this document is fully correct and consistent with the official documents on IPP from the IETF.

This version includes questions raised on the IPP DL between July 1 and August 17, 1998

All references are to the June 30, 1998 drafts.

The purpose of this document is to collect information about implementation questions against the current IPP draft documents. Allowable questions are about things like suspected errors, inconsistencies, or needs for further clarifications. Questions about extensions or functional changes to the drafts are dealt with in the overall IPP development activities and are outside the scope of this document. Please note that even if a question does get listed, the PWG might decide that it is outside the scope of the Implementor's Guide and remove it in a later version.

The document may contain advice to implementors that goes beyond the exact IPP conformance requirements, e.g. how to ensure interoperability with earlier versions of Internet components, or even early implementations of IPP itself.

Each new **Question** on the IPP DL has been listed in a separate table. Added in the table is also one section called **Discussion**, which reflects comments back from other IPP DL participants. When the PWG has come up with an agreed Answer to the Question, it is reflected in the **Answer** section of the table. At this stage, the **Discussion** section is usually removed.

1. Model & Semantics

Question	For each job template attribute there is the associated default and supported values. I have a question about the xxx-supported values. Imagine a printer that say supports binding which may be controlled by various PDL commands, but does not support controlling binding via the IPP finishings job template attribute. Should the printer response to finishings-supported include binding or not? I assume that it should not include binding as this would give the idea to the client that binding can be controlled with the finishings attribute. Thus, xxx-supported is not intended to indicate printer capabilities, but rather support for the IPP attributes. Is this correct?

Question	It looks like the problem discussed in "document-format-supported" [MOD needs clarification], http://www.findmail.com/list/ipp/showthread.html?num=3864 was addressed in the new MOD, ftp://ftp.ietf.org/internet-drafts/draft-ietf-ipp-model-10.txt June 30, 1998. The new words say: "If the Printer object does distinguish between different sets of supported values for each different document format specified by the client, this specialization applies only to the following Printer object attributes: - Printer attributes that are Job Template attributes ("xxx-default" "xxx-supported", and "xxx-ready" in the Table in Section 4.2), - "pdl-override-supported", - "compression-supported", - "job-k-octets-supported", - "job-impressions-supported, - "job-media-sheets-supported" - "printer-driver-installer", - "color-supported", and - "reference-uri-schemes-supported" "The values of all other Printer object attributes (including "document-format-supported") remain invariant with respect to the client supplied document format (except for new Printer description attribute as registered according to section 6.2). While this new wording gets around the problem, I think it presents a poor model. It blatantly violates Second Normal Form, in that some Printer attributes depend on the (Printer identifier, document-format) tuple, while others depend only on the Printer identifier. The model says that all these attributes, including those that vary with document-format (e.g., number-up), are attributes of the Printer class of objects. But the implication is that each real-wold printer maps to a whole set of Printer object instances, selected by document-format. Attributes (e.g. printer-name) which don't vary with document-format are redundantly stored in each instance. Updates to attributes that don't vary with document-format (e.g. printer-state) require visiting all the instances. A better model would split the existing Printer into two classes of objects: 1) a new, reduced Printer, and 2) something else that could be called "Interpreter". Then the attributes can be normalized between these two new classes. Attributes that don't vary with document-format are assigned to the Printer. Each real-world printer maps to one instance of Printer. Attributes that do vary with document-format are assigned to Interpreter. Each Printer instance contains one or more Interpreter instances, selected by document-format. I know that IPP doesn't claim to be truly object-oriented. But I think considerations like this are important for a few reasons:

	model trace cleanly back to the real world. **Carl Kugler**
Discussion	
Answer	

Question	In the Job Template Attributes there are attributes that can be a type3 keyword or a name (job-hold-until, job-sheets, and media). As I read the spec, these attributes are usually type3 keywords but can optionally be changed at the printer to a name type. Is this correct or did I miss something in the spec? My question is how does an IPP client know which type to send? If the wrong type is sent, what should the expected reply be? **Rajesh Chawla**
Discussion	My understanding, based on my reading of the spec and questions I've asked here in the past: Those attributes can be typed, and tagged as any of the following: 0x36 nameWithLanguage 0x42 nameWithoutLanguage 0x44 keyword In general, an IPP Object may send any one of the three types, and must accept any one of the three. However, for any 'name' attribute in the request that is in a different natural language than the value supplied in the "attributes-natural-language", the sender must use the nameWithLanguage form. Type 3 keywords have standard, registered values. If the wrong type is sent in a request, according to MOD section 16.4.3, the response should be 'client-error-request-value-too-long'. Quote: "IF NOT any single 'keyword' or 'name' value less than or equal to 255 octets, REJECT/RETURN 'client-error-request-value-too-long'.") **Carl Kugler**
Answer	

Question	The table in Section 4 says that "document-format-default" and "document-format-supported" are REQUIRED, but the descriptions of those attributes in sections 4.4.18 and 4.4.19 do not say REQUIRED. I believe that 4.4.18 and 4.4.19 should be fixed by adding REQUIRED to agree with the table, like the other attributes that are REQUIRED.

Question	How should the server handle the situation where the "attributes-charset" of the response itself is "us-ascii", but one or more attributes in that response is in the "utf-8" format? Consider a case where a client sends a Print-Job request with "utf-8" as the value of "attributes-charset" and with the "job-name" attribute supplied. Later another client sends a Get-Job-Attribute or Get-Jobs request. This second request contains the "attributes-charset" with value "us-ascii" and "requested-attributes" attribute with exactly one value "job-name". According to the IPP-Mod document (section 3.1.4.2), the value of the "attributes-charset" for the response of the second request must be "us-ascii" since that is the charset specified in the request. The "job-name" value, however, is in "utf-8" format. Should the request be rejected even though both "utf-8" and "us-ascii" charsets are supported by the server? or should the "job-name" value be converted to "us-ascii" and return "successful-ok-conflicting-attributes" (0x0002) as the status code? **Van Dang**
Discussion	
Answer	

Question	I recently noticed there is no pages-per-minute attribute in IPP. I noticed this first when reviewing the draft printer scheme for SLP (draft-ietf-srvloc-printer-scheme-02.txt). The printer scheme seems to inherit it's attribute definitions from IPP. I think ppm is one of the most fundamental attributes in terms of printer selection. I'm sure this must have been discussed at some point during IPP development, probably at a time when I wasn't paying much attention to the mail list. I do remember a discussion about a cost attribute that was eliminated because it was deemed too qualitative. But ppm is quantitative and universal in advertising printers. So, can someone explain why it is not an IPP printer attribute? And, for those familiar with the SLP printer scheme effort, why is it not part of the SLP printer scheme? **Angelo Caruso**
Discussion	You could make this a directory attribute, but I don't think its absolutely necessary to support it in IPP. Besides, its in the printer MIB ;) **Randy Turner** I think that we discussed this at some stage and found that it was not clear that we could come up with a single value. For example, depending on the type of printer, the speed is often dependent on whether you run in "draft" mode vs. "quality" mode, and whether you run B/W or color. So we would have ended up with some kind of conditions and several values to cover all cases. **Carl-Uno Manros**

	configuration? **Harry Lewis**
Answer	

Question	Can you implement the operations "Create-Job", "Send-Document" and "Send-URI", without the need to support multiple documents? This could be useful for environments where you have long jobs, but do not need support for multiple documents. **Carl-Uno Manros**
Discussion	The model document supports the notion of a Create-Job operation followed by only one Send-Document operation as semantically equivalent to a Print-Job operation. It cautions regarding performance, however. If you are asking is it ok to support Creat-Job, Send-Doc with only one document - Yes. If you are asking is it ok to support Create-Job but LIMIT Send-Doc to only one document... I'd say that would be a non-no! **Harry Lewis**
Answer	

Question	What was the rationale for making the "printer-up-time" attribute a REQUIIRED attribute, considering that the other 3 attributes "time-at-creation", "time-at-processing", and "time-at-completed", with which it is assicociated, are all OPTIONAL? **Carl-Uno Manros**
Discussion	Don't know for sure but I suspect this attempts to make a running "time marker" available for monitoring, tracking accounting etc... without mandating all the possible time recording points on each IPP device. This si somewhat analogous to the sysUpTime concept in MIB-II. **Harry Lewis**
Answer	

2. Encoding and Transport

Question	The document says "A client MUST NOT expect a response from an IPP server until after the client has sent the entire response." What about "100 Continue" responses (and related HTTP Expect headers)? **Carl Kugler**
Discussion	
Answer	

Question	The URI in the HTTP layer is either relative or absolute and is used by the HTTP server to route the HTTP request to the correct resource relative to that HTTP server. This "resource" is either an IPP Printer or a Job, right? The HTTP server need not be aware of the URI within the operation request. Once the HTTP server resource begins to process the HTTP request, it might get the reference to the appropriate IPP Printer object from either the HTTP URI (using to the context of the HTTP server for relative URLs) or from the URI within the operation request; the choice is up to the implementation. Once the Printer or Job ("the HTTP server resource") has begun to process the job, why would it need a reference to an appropriate IPP Printer object? Implementation question: the server is REQUIRED to check for the presence of target URI operation attributes in every request (and respond with client-error-bad-request if not found) but is otherwise free to ignore target op. atts.? The Request-URI and target URLs might not be literally identical although they MUST both reference the same IPP object; however the server isn't required to verify this? **Carl Kugler**
Discussion	The document says "It is REQUIRED that a printer implementation support HTTP over the IANA assigned Well Known Port 631...". Therefore, a conforming implementation MUST be on 631, although it might also be on some other port simultaneously. **Larry Masinter** I believe that if a product offers compliant operation--but also offers configuration options to disable compliant operation—that the product is still considered compliant. **Jay Martin** I think we need some clarification in the wording here, because I think it's ambiguous as it stands. Here's another quote: "IPP server implementations MUST offer IPP services

The intention is that IPP software must be able to listen on port 631 and may be able to listen on other ports. If such software allows an administrator to configure the ports, then such an adminstrator may be able to have IPP software listen other ports, such as port 80 and might make it be the only port that the IPP server listens on. The customer has the right to do what he wants, even if we don't think it is the wisest choice.

Bob Herriot

What I find ambiguous, then, are the words "in addition to" and "as well" in these sentences:

"IPP server implementations may support other ports, in addition to this port."

"It is REQUIRED that a printer implementation support HTTP over the IANA assigned Well Known Port 631 (the IPP default port), though a printer implementation may support HTTP over port some other port as well."

Maybe it should say:

"It is REQUIRED that a printer implementation support HTTP over the IANA assigned Well Known Port 631 (the IPP default port), though a printer implementation may be configured to listen on some other port instead."

Carl Kugler

Or perhaps:

"...may be configured to listen on one or more ports instead."

Jay Martin

No, I don't think so. Implementations should default to port 631, but people who buy these things should be able to change that default to anything they want.

Here's how I would say this:

1. Implementations MUST support the ability to accept IPP requests on port 631
2. Implementations MAY support the ability to accept IPP requests on other ports instead of, or in addition to, port 631, but only if explicitly configured to do so by the printer administrator.

Keith Moore, AD

It is REQUIRED that a printer implementation support HTTP over the IANA assigned

	Carl Kugler Here's another try: Replace: It is REQUIRED that a printer implementation support HTTP over the IANA assigned Well Known Port 631 (the IPP default port), though a printer implementation may support HTTP over some other port as well. In addition, a printer may have to support another port for privacy (See Section 5 “Security Considerations”). Note: even though port 631 is the IPP default, port 80 remains the default for an HTTP URI. Thus a URI for a printer using port 631 MUST contain an explicit port, e.g. "http://forest:631/pinetree". with: IANA has assigned Well Known Port 631 to the default IPP port number. An IPP server listens on port 631 unless explicitly configured to listen on one or more ports instead of, or in addition to, port 631. If the port is empty or not given in an "ipp" schemed URL, port 631 is assumed. **Carl Kugler** I think we should take the text suggested by Keith Moore. **Carl-Uno Manros** I think Keith's words: 1. Implementations MUST support the ability to accept IPP requests on port 631. 2. Implementations MAY support the ability to accept IPP requests on other ports instead of, or in addition to, port 631, but only if explicitly configured to do so by the printer administrator. are still a little ambiguous. What does it mean to "support the ability to accept"? **Carl Kugler**
Answer	

3. Interoperability Questions for the September 1998 Bake-off

Question	The document says the client and server MUST support the "chunked" transfer encoding when receiving. My question is: Can we count on this? I.e., if our client always transmits requests using the "chunked" transfer encoding, will we be able to interoperate

I think this is a good example of the things that we need to put in an Implementor's Guide document. Formally, as we are referencing IETF's HTTP 1.1, all IPP implementations MUST support "chunking". However, I think it would be smart for an actual implementation to also support using HTTP 1.0 to allow for maximum interoperability. E.g. in Xerox we have had to base our current client code (written in Java) on HTTP 1.0, as the JavaSoft libraries do not yet support some of the HTTP 1.1 features. These problems will disappear over time, but we also need to get things working short term.

Carl-Uno Manros

You might find that some implementations don’t support chunking.

Paul Moore

Well, I'm assuming since we "last-called" these documents in the WG, that everybody is in agreement that an implementation that doesn't support chunking isn't compliant.

Randy Turner

I dont see where it says that a server must support chunking. It says I must support 1.1. Maybe I am reading it wrong (I guess thats why we have bake-offs).

Paul Moore

Okay, as a practical matter it looks like we need to be able to support HTTP/1.0 in addition to HTTP/1.1, and "Content-Length" in addition to "Transfer-Encoding: chunked" whether sending or receiving.

This leads me to my next question: What is the transition sequence for the cases in which an HTTP/1.1 client that normally uses chunking wants to make a request of a server that is HTTP/1.0 and/or doesn't understand chunking?

Does this scenario look correct?:

1. Client requests HTTP/1.1, Transfer-Encoding: chunked
2. Server responds HTTP/1.0 501 (Unimplemented), and closes the connection
3. Client opens new connection and requests HTTP/1.0, Content-Length

Also, don't some HTTP/1.0 implementations understand Transfer-Encoding as an extension? We have some client situations where we can't avoid chunking, so it would be nice if even the HTTP/1.0 servers understood chunked transfer coding. (Presumably all HTTP/1.1 applications are able to receive and decode the “chunked” transfer coding.)

Carl Kugler

	Larry Masinter Here is our situation. For reasons beyond the scope of this discussion, we need to build a client-side API that supports an open,write,write...close|cancel paradigm for sending the document data. This is easy to do with chunking. It may be impossible to do with Content-Length (due to buffering limitations). So our client will normally use chunking. We'd really want to avoid having to design it to fall-back to Content-Length when it encounters an HTTP/1.0 IPP server, because that would be complicated and expensive, and won't always work. On the other hand, we don't want to paint ourselves into a corner and build a client that relies heavily on chunking, only to find out that many server implementations can't receive and decode the chunked transfer coding. Is it safe to assume that all IPP 1.0 products will support HTTP/1.1 (and therefore chunking), even if prototype implementations don't? **Carl Kugler** It doesn't make sense to require HTTP1.1 which requires eating chunks (not emitting) but write IPP as if eating chunks is optional. **Harry Lewis** I agree that we should all be eating chunks when we roll out products. The problem is that some do not have it in the prototypes right now. Let us check this out in our bake-off and see how much of a problem it really is. Come September, there may be more chunk eaters... **Carl-Uno Manros** Is it worth traveling to Redmond just to find out we can't interoperate at all because some implementations send chunks while others can't receive them? Can't we just take a poll and see now how many plan to be able to receive and decode the chunked transfer coding by September? **Carl Kugler**
Answer	