1 INTERNET-DRAFT   Roger deBry
2 <draft-ietf-ipp-collection-04.txt>   Utah Valley State College
3   T. Hastings
4   Xerox Corporation
5   R. Herriot
6   Xerox Corporation
7   K. Ocke
8   Xerox Corporation
9   P. Zehler
10   Xerox Corporation
11   January 24, 2001

12

Internet Printing Protocol (IPP):
The 'collection' attribute syntax

16

26 **Abstract**
27   This document specifies an OPTIONAL attribute syntax called 'collection' for use with the Internet
28   Printing Protocol/1.0 (IPP) [RFC2565, RFC2566], IPP/1.1 [RFC2911, RFC2910], and subsequent
29   versions. A 'collection' is a container holding one or more named values, which are called "member"
30   attributes. A collection allows data to be grouped like a PostScript dictionary or a Java Map. This
31   document also specifies the conformance requirements for a definition document that defines a collection
32   attribute. Finally, this document gives some illustrative example collection attribute definitions that are not
33   intended as actual attribute specifications.
34

34   The full set of IPP documents includes:

35   Design Goals for an Internet Printing Protocol [RFC2567]
36   Rationale for the Structure and Model and Protocol for the Internet Printing Protocol [RFC2568]
37   Internet Printing Protocol/1.1: Model and Semantics [RFC2911]
38   Internet Printing Protocol/1.1: Encoding and Transport [RFC2910]
39   Internet Printing Protocol/1.1: Implementer's Guide [IPP-IIG]
40   Mapping between LPD and IPP Protocols [RFC2569]
41
42   The "Design Goals for an Internet Printing Protocol" document takes a broad look at distributed printing
43   functionality, and it enumerates real-life scenarios that help to clarify the features that need to be included in a
44   printing protocol for the Internet. It identifies requirements for three types of users: end users, operators, and
45   administrators. It calls out a subset of end user requirements that are satisfied in IPP/1.0. A few OPTIONAL
46   operator operations have been added to IPP/1.1.

47   The "Rationale for the Structure and Model and Protocol for the Internet Printing Protocol" document
48   describes IPP from a high level view, defines a roadmap for the various documents that form the suite of IPP
49   specification documents, and gives background and rationale for the IETF working group's major decisions.

50   The "Internet Printing Protocol/1.1: Encoding and Transport" document is a formal mapping of the abstract
51   operations and attributes defined in the model document onto HTTP/1.1 [RFC2616]. It defines the encoding
52   rules for a new Internet MIME media type called "application/ipp". This document also defines the rules for
53   transporting over HTTP a message body whose Content-Type is "application/ipp". This document defines a
54   new scheme named 'ipp' for identifying IPP printers and jobs.

55   The "Internet Printing Protocol/1.1: Implementer's Guide" document gives insight and advice to implementers
56   of IPP clients and IPP objects. It is intended to help them understand IPP/1.1 and some of the considerations
57   that may assist them in the design of their client and/or IPP object implementations. For example, a typical
58   order of processing requests is given, including error checking. Motivation for some of the specification
59   decisions is also included.

60   The "Mapping between LPD and IPP Protocols" document gives some advice to implementers of gateways
61   between IPP and LPD (Line Printer Daemon) implementations.

62

62   **Table of Contents**

63

98

99   **Table of Tables**

00

12

.12

1 Introduction

1.1 Problem Statement

.15 The IPP Model and Semantics [RFC2911] supports most of the common data structures that are available in
.16 programming languages. It lacks a mechanism for grouping several attributes of different types. The Java
.17 language uses the Map to solve this problem and PostScript has a dictionary. The new mechanism for
.18 grouping attributes together (called 'collection' mechanism) must allow for optional members and subsequent
.19 addition of new members.

.20 The 'collection' mechanism must be encoded in a manner consistent with existing 1.0 and 1.1 parsing rules (see
.21 [RFC2910]). Current 1.0 and 1.1 parsers that don't support the 'collection' mechanism must not confuse
.22 collections or parts of collection they receive with other attributes.

1.2 Solution

.24 The new mechanism is a new IPP attribute syntax called a 'collection'. As such, each collection value is a
.25 value of an attribute whose attribute syntax type is defined to be a 'collection'. Such an attribute is called a
.26 collection attribute. The name of the collection attribute serves to identify the collection value in an operation
.27 request or response, as with any attribute value.

.28 The 'collection' attribute syntax is a container holding one or more named values (i.e., attributes), which are
.29 called member attributes. Each collection attribute definition document lists the mandatory and optional
.30 member attributes of each collection value. A collection value is similar to an IPP attribute group in a request
.31 or a response, such as the operation attributes group. They both consist of a set of attributes.

.32 As with any attribute syntax, the document that defines a collection attribute specifies whether the attribute is
.33 single-value (collection) or multi-valued (1setOf collection). If the attribute is multi-valued (1setOf collection)
.34 each collection value MUST be a separate instance of a single definition of a collection, i.e. it MUST have the
.35 same member attributes except for OPTIONAL member attributes. If we view each collection definition as a
.36 separate syntax type, this rule continues the IPP/1.1 notion that each attribute has a single type or pattern (e.g.
.37 "keyword | name" is a pattern). Without this rule, the supported values would be more difficult to describe and
.38 the mechanism defined in item 4 of section 3.1 would not be sufficient.

.39 The name of each member attribute MUST be unique for a collection attribute, but MAY be the same as the
.40 name of a member attribute in another collection attribute and/or MAY be the same as the name of an
.41 attribute that is not a member of a collection. The rules for naming member attributes are given in section 3.1.

42   Each member attribute can have any attribute syntax type, including 'collection', and can be either single-
43   valued or multi-valued. The length of a collection value is not limited. However, the length of each member
44   attribute MUST NOT exceed the limit of its attribute syntax.

45   The member attributes in a collection MAY be in any order in a request or response. When a client sends a
46   collection attribute to the Printer, the order that the Printer stores the member attributes of the collection value
47   and the order returned in a response MAY be different from the order sent by the client.

48   A collection value MUST NOT contains two or more member attributes with the same attribute name. Such
49   a collection is mal-formed. Clients MUST NOT submit such malformed requests and Printers MUST NOT
50   return such malformed responses. If such a malformed request is submitted to a Printer, the Printer MUST
51   (depending on implementation) either (1) reject the request with the 'client-error-bad-request' status code (see
52   section 13.1.4.1), or (2) accept the request and use only one of each duplicate member attribute.

53   2   Terminology

54   This section defines terminology used throughout this document.

55   2.1   Conformance Terminology

56   Capitalized terms, such as MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, MAY, NEED
57   NOT, and OPTIONAL, have special meaning relating to conformance. These terms are defined in
58   [RFC2911] section 12.1 on conformance terminology, most of which is taken from RFC 2119 [RFC2119].

59   The following specialization of these terms apply to this document:

60   REQUIRED: if an implementation supports the extensions described in this document, it MUST support a
61   REQUIRED feature.

62   OPTIONAL: if an implementation supports the extensions described in this document, it MAY support an
63   OPTIONAL feature.

64   2.2   Other terminology

65   This document uses terms such as Job object (or Job), IPP Printer object (or Printer), "operation", "request",
66   response", "attributes", "keywords", and "support". These terms have special meaning and are defined in the
67   model terminology [RFC2911] section 12.2. The following additional terms are introduced in this document:

68   collection: an attribute syntax in which each attribute value is a set of attributes, called *member attributes*.

69   member attribute: an attribute that is defined to be used as one of the attributes in a *collection*.

170   collection attribute: an attribute whose definition specifies the 'collection' attribute syntax and each of the
171   member attributes that MAY occur in a collection attribute value.

3   Definition of a Collection Attribute

173   This section describes the requirements for any collection attribute definition.

3.1   Information to Include

175   When a specification document defines an "xxx" collection attribute, i.e., an attribute whose attribute syntax
176   type is 'collection' or '1setOf collection'; the definition document MUST include the following aspects of the
177   attribute semantics. Suppose the "xxx" collection attribute contains N member attributes named "aaa1",
178   "aaa2", …, "aaaN" ("aaaI" represents any one of these N member attributes).

179   1. The name of the collection attribute MUST be specified (e.g. "xxx"). The selection of the name "xxx"
180   MUST follow the same rules for uniqueness as for attributes of any other syntax type (as defined by
181   IPP/1.1) unless "xxx" is a member attribute of another collection. Then the selection of the name "xxx"
182   MUST follow the rules for uniqueness defined in item 5a) of this list.

183   2. The collection attribute syntax MUST be of type 'collection' or '1setOf collection'.

184   3. The context of the collection attribute MUST be specified, i.e., whether the attribute is an operation
185   attribute, a Job Template attribute, a Job Description attribute, a Printer Description attribute, a member
186   attribute of a particular collection attribute, etc.

187   4. An "xxx-supported" attribute MUST be specified and it has one of the following two forms:

188   a) "xxx-supported" is a "1setOf collection" which enumerates all of the supported collection values of
189   "xxx". If a collection of this form contains a nested collection, it MUST be of the same form.
190
191   For example, "media-size-supported" might have the values {{x-dimension:210, y-dimension:297},{x-
192   dimension:297, y-dimension:420}} to show that it supports two values of "media size": A4 (210x297)
193   and A3 (297x420). It does not support other combinations of "x-dimension" and "y-dimension"
194   member attributes, such as 210x420 or 297x297 and it does not supported non-enumerated values,
195   such as 420x595.

196   b) "xxx-supported" is a "1setOf type2 keyword" which enumerates the names of all of the member
197   attributes of "xxx": "aaa1", "aaa2", …, "aaaN". If a collection of this form contains a nested collection, it
198   MAY be of either form. See item 5f) below for details on supported values of member attributes.
199
200   For example, "media-col-supported" might have the keyword values: "media-size" and "media-color".

201   5. The member attributes MUST be defined. For each member attribute the definition document MUST
202   provide the following information:

203   a) The member attribute's name (e.g., "aaa") MUST be unique within the collection being defined and
204   MUST either

205   i)   reuse the attribute name of another attribute (that is unique across the entire IPP attribute name
206   space) and have the same syntax and semantics as the reused attribute (if the condition of item 4b)
207   above is met). For example, a member attribute definition could reuse the IPP/1.1 "media"
208   attribute.

209   ii) potentially occur elsewhere in the entire IPP attribute name space. (if the condition of item 4a)
210   above is met). For example, a member attribute could be "x-dimension" which could potentially
211   occur in another collection or as an attribute outside of a collection.

212   iii) be unique across the entire IPP attribute name space (if the condition of item 4b) above is met).
213   For example, a member attribute could be "media-color" which must unique be across the entire
214   IPP attribute name space.

215   b) Whether the member attribute is REQUIRED or OPTIONAL for the Printer to support

216   c) Whether the member attribute is REQUIRED or OPTIONAL for the client to supply in a request

217   d) The member attribute's syntax type, which can be any attribute syntax, including '1setOf X', 'collection',
218   and '1setOf collection'. If this attribute name reuses the name of another attribute (case of item a1
219   above), it MUST have the same attribute syntax, including cardinality (whether or not 1setOf).

220   e) The semantics of the "aaa" member attribute. The semantic definition MUST include a description of
221   any constraint or boundary conditions the member attribute places on the associated attribute,
222   especially if the attribute reuses the name of another attribute (case of item a1 above)

223   f) The supported values for the each "aaaI" member attribute (of the member attributes "aaa1", "aaa2",
224   …, "aaaN") is specified by one of two mechanisms.

225   i)   If "xxx-supported" is a "1setOf collection" (see item 4a) above), the value for each "aaaI" is
226   specified in each collection value of "xxx-supported" in the context of other member attributes. That
227   is, "xxx-supported" enumerates all supported values of "xxx".
228

229   ii) If the value of "xxx-supported" is a "1setOf type2 keyword" (see item 4b) above), the supported
230   values of "aaaI" are the values specified by either i) the "aaaI-supported" attribute or ii) the
231   definition of the member attribute "aaaI" within the document defining the "xxx" attribute. The values
232   of each member attribute "aaaI" are specified independently of other member attributes though a
233   Printer is not required to support all combinations of supported values.
234

235   For example, "media-col-supported" might have the keyword values: "media-size" and "media-
236   color". Using the first method for defining supported values (an "aaaI-supported" attribute), the
237   collection values of "media-col" are combinations of values of "media-size-supported" and "media-
238   color-supported".If "media-size-supported" has the values of '210x297' and '297x420' and
239   "media-color-supported" has the values of 'white' and 'pink', the Printer might support only the
240   combinations 'white-210x297', 'pink-210x297'and 'white-297x420', and not 'pink-297x420'.
241
242   If a collection contains a member "aaaI" whose syntax type is "text", the supported values would
243   probably be defined by the definition of "xxx" rather than by the attribute "aaaI-supported".

244   g) the default value of each "aaaI" member attribute if it is OPTIONAL for a client to supply the "aaa"
245   member attribute in a request. The default value is specified by in the attribute's definition within a
246   document and MUST be one of the following:

247   i)   a fixed default

248   ii) a mechanism by which the Printer determines default

249   iii) an indefinite default that is left to the implementation.

250   iv) an attribute that the Printer uses to determine the default

251 6. The default value of "xxx" if a client does not supply it. The default value is specified by in the attribute's
252   definition within a document and MUST be one of the following:

253   a) a fixed default

254   b) a mechanism by which the Printer determines default

255   c) an indefinite default that is left to the implementation

256   d) a Printer attribute "xxx-default" which is a collection with the same member attributes as "xxx". Though
257   optional member attributes may be absent in which case the Printer uses the defaulting rules of item 5g)
258   above.

259 7. The "xxx-ready (1setOf collection)" attribute if human intervention is required to make many of the
260   supported values available. For example, "media-col" is an attribute which has a "ready" attribute. Most
261   attributes do not have a "ready" attribute.

3.2   Nested Collections

263   A member attribute may have a syntax type of 'collection' or '1setOf collection', in which case it is called a
264   nested collection attribute. The rules for a nested collection attribute are the same as for a collection attribute
265   as specified in section 3.1.

4   Collection Attributes as Attributes in Operations

4.1   General Rules

A collection value is like any other IPP/1.1 value, except that it is structured. The rules for attributes with collection values are the same as for attributes of any other syntax type (see IPP/1.1), be they in any group of a request of a response.

4.2   Unsupported Values

The rules for returning an unsupported collection attribute are an extension to the current rules:

1.   If the entire collection attribute is unsupported, then the Printer returns just the collection attribute name with the 'unsupported' out-of-band value (see the beginning of [RFC2911] section 4.1) in the Unsupported Attributes Group.

2.   If a collection contains unrecognized, unsupported member attributes and/or conflicting values, the attribute returned in the Unsupported Group is a collection containing the unrecognized, unsupported member attributes, and/or conflicting values. The unrecognized member attributes have an out-of-band value of 'unsupported' (see the beginning of [RFC2911] section 4.1). The unsupported member attributes and conflicting values have their unsupported or conflicting values.

5   Example definition of a collection attribute

In some printing environments, it is desirable to allow the client to select the media by its properties, e.g., weight, color, size, etc., instead of by name. In IPP/1.1 (see [RFC2911]), the "media (type3 keyword | name) Job Template attribute allows selection by name.   It is tempting to extend the "media" attribute syntax to include "collection", but then existing clients could not understand default or supported media values that use the collection value. To preserve interoperability, a new attribute MUST BE added, e.g., "media-col (collection)". The following subsections contain a sample definition of a simplified "media-col" attribute. The definition follows the rules in section 3.

All of the example attribute definitions in this document are illustrative examples, rather than actual definitions. These examples are intended to illustrate how to define collection attributes. Other documents MUST define collection attributes for use in actual interchange. Such definitions may be very similar to the examples in this document, since we attempted to pick useful examples.

Note: we picked the name "media-col" because the name "media" is already in use. Ordinarily the collection attribute would have a name like any other attribute and would not end in "col".

295   The member attributes of "media-col" attribute ("media-color (type 3 keyword)" and "media-size (collection)")
296   both follow the naming rules of item 4a3 of section 3, i.e. the names are unique across the entire IPP attribute
297   name space. The member attributes of the "media-size (collection)" member attribute ("x-dimension
298   (integer(0,MAX))" and "y-dimension (integer(0,MAX))") both follow the naming rules of item 4a2 of section
299   3, i.e. they potentially occur elsewhere in the IPP attribute name space.

300   **5.1   media-col (collection)**

301   The "media-col" (collection) attribute augments the IPP/1.1 [RFC2911] "media" attribute. This collection
302   attribute enables a client end user to submit a list of media characteristics to the Printer. When the client
303   specifies media using the "media-col" collection attribute, the Printer object MUST match the requested media
304   exactly. The 'collection' consists of the following member attributes:

305   **Table 1 - "media-col" member attributes**

Attribute name	attribute syntax	request	Printer Support
media-color	type3 keyword | name (MAX)	MAY	MUST
media-size	collection	MUST	MUST

306

307   The definitions for the member attributes is given in the following sub-sections:

308   **5.1.1   media-color (type3 keyword | name(MAX)**

309   This member attribute identifies the color of the media. Valid values are 'red', 'white' and 'blue'

310   The "media-color-supported" (1setOf (type3 keyword | name(MAX))) Printer attribute identifies the values
311   of this "media-color" member attribute that the Printer supports, i.e., the colors supported.

312   If the client omits this member attribute, the Printer determines the value in an implementation dependent
313   manner.

314   **5.1.2   media-size (collection)**

315   This member attribute identifies the size of the media. The 'collection' consists of the member attributes
316   shown in Table 2:

317   **Table 2 - "media-size" collection member attributes**

Attribute name	attribute syntax	request	Printer Support
x-dimension	integer (0:MAX)	MUST	MUST
y-dimension	integer (0:MAX)	MUST	MUST

318

319   The definitions for the member attributes is given in the following sub-sections:

5.1.2.1   x-dimension (integer(0:MAX))

321   This attribute identifies the width of the media in inch units along the X axis.

5.1.2.2   y-dimension (integer(0:MAX))

323   This attribute identifies the height of the media in inch units along the Y axis.

324   The "media-size-supported" (1setOf collection) Printer attribute identifies the values of this "media-
325   size" member attribute that the Printer supports, i.e., the size combinations supported. The names
326   of the member attributes are the same as the member attributes of the "media-size" collection
327   attribute, namely "x-dimension", and "y-dimension", since they have the same attribute syntax and
328   the same semantics.

5.2   media-col-default (collection)

330   The "media-col-default" Printer attribute specifies the media that the Printer uses, if any, if the client omits the
331   "media-col" and "media". Job Template attribute in the Job Creation operation (and the PDL doesn't include a
332   media specification). The member attributes are defined in Table 1. A Printer MUST support the same
333   member attributes for this default collection attribute as it supports for the corresponding "media-col" Job
334   Template attribute.

5.3   media-col-ready (1setOf collection)

336   The "media-col-ready" Printer attribute identifies the media that are available for use without human
337   intervention, i.e., the media that are ready to be used without human intervention. The collection value MUST
338   have all of the member attributes that are supported in Table 1.

5.4   media-col-supported (1setOf type2 keyword)

340   The "media-col-supported" Printer attribute identifies the keyword names of the member attributes supported
341   in the "media-col" collection Job Template attribute, i.e., the keyword names of the member attributes in Table
342   1 that the Printer supports.

6   A Second Example Definition Of A Collection Attribute

344   All of the example attribute definitions in this document are illustrative examples, rather than actual definitions.
345   These examples are intended to illustrate how to define collection attributes. Other documents MUST define

846   collection attributes for use in actual interchange. Such definitions may be very similar to the examples in this
847   document, since we attempted to pick useful examples.

848   In some printing environments, it is desirable to allow the client to select the media for the job start sheet. The
849   reason for not adding the 'collection' attribute syntax to the existing "job-sheets" Job Template attribute is the
850   same as for "media". Instead, a new Job Template attribute is introduced, e.g. "job-sheet-col (collection)".

851   The member attributes of "job-sheet-col" attribute ("job-sheets (type 3 keyword)" and "media (type3
852   keyword | name)") both follow the naming rules of item 4a1 of section 3, i.e they reuse existing IPP attributes.
853   According to the rules, their supported values come from the existing IPP attributes: "job-sheets-supported"
854   and "media-supported". However, their default values do not come from "job-sheets-default" and "media-
855   default", respectively. Rather the definition of "job-sheet-col" says that "job-sheets (type 3 keyword)" is
856   required and if "media (type3 keyword | name)" is absent, the Printer uses the same media as the rest of the
857   job uses.

858   If "job-sheet-col" attribute were defined to contain the member attribute "job-sheet-media (type3 keyword |
859   name)" instead of "media (type3 keyword | name)", then the definition would also have to specify a "job-sheet-
860   media-supported (1setOf (type3 keyword | name))" whose values would be independent of "media-
861   supported (1setOf (type3 keyword | name))" and would be set separately by a System Administrator.

862   The actual text for the definition of the attribute is left as an exercise for the reader.

863   **7   Encoding**

864   This section defines the additional encoding tags used according to [RFC2910] and gives an example of their
865   use. The encoding tags define in this document MUST be used by all collection attributes defined in other
866   documents. However, the example of their use is illustrative only.

867   **7.1   Additional tags defined for representing a collection attribute value**

868   The 'collection' attribute syntax uses the tags defined in Table 3.

869   **Table 3 - Tags defined for encoding the 'collection' attribute syntax**

Tag name	Tag value	Meaning
begCollection	0x34	Begin the collection attribute value.
endCollection	0x37	End the collection attribute value.
memberAttrName	0x4A	The value is the name of the collection member attribute

870

871   When encoding a collection attribute "xxx" that contains an attribute "aaa" and is not inside another collection,
872   the encoding follows these rules:

373   1.   The beginning of the collection is indicated with a value tag that MUST be syntax type 'begCollection'
374   (0x34) with a name length and Name field that represent the name of the collection attribute ("xxx") as
375   with any attribute, followed by a value. The Printer MAY ignore the value and its length of MAY be 0.
376   In the future, however, this field MAY contain useful information, such as the collection name (cf. the
377   name of a C struct).

378   2.   Each member attribute is encoded as a sequence of two or more values that appear to be part of a
379   single multi-valued attribute, i.e. 1setOf. The first value after the 'begCollection' value has the attribute
380   syntax 'memberAttrName' (0x4A) and its value holds the name of the first member attribute (e.g. "aaa").
381   The second value holds the first member's attribute value, which can be of any attribute syntax, except
382   'memberAttrName' or 'endCollection'. If the first member's attribute value is multi-valued, the third
383   value holds the second value of the first member's value. Otherwise, the third value holds the name of
384   second member attribute (e.g. "bbb") and its attribute syntax is 'memberAttrName'. In this case, the
385   fourth member's value is the value of "bbb".
386
387   Note that the technique of encoding a 'collection' as a '1setOf' makes it easy for a Printer that doesn't
388   support a particular collection attribute (or the collection attribute syntax at all) to simply skip over the
389   entire collection value.

390   3.   The end of the collection is indicated with a value tag that MUST be syntax type 'endCollection' (e.g.
391   0x37) and MAY have a zero name length and a zero value length. In the future, this field MAY contain
392   useful information,such as the collection name that matches the one in the 'begCollection' .

393   4.   It is valid to have a member attribute that is, itself, a collection attribute, i.e., collections can be nested
394   within collections. This is represented by the occurrence of a member attribute that is of attribute syntax
395   type 'begCollection'. Such a collection is terminated by a matching 'endCollection'. The name of such a
396   member attribute is in the immediately preceding value whose syntax type is 'memberAttrName'.

397   5.   It is valid for a collection attribute to be multi-valued, i.e., have more than one collection value. If the
398   next attribute immediately following the 'endCollection' has a zero name length and a tag of
399   'begCollection', then the collection attribute is a multi-valued collection, as with any attribute. This
400   statement applies to collections within collections and collections that are not in collections.

401   **7.2   Example encoding: "media-col" (collection)**

402   The collection specified in section 5 is used for the encoding example shown in Table 5. The example also
403   shows nested collections, since the "media-size" member attribute is a 'collection. The encoding example
404   represents a blue 4x6-index cards and takes 216 octets. The Appendices contains more complex examples.

405   Additional examples have been included in the appendices.

406   The overall structure of the two collection values can be pictorially represented as:

407   "media-col" =

```
408        {    "media-color" =  'blue';
409              "media-size" =
410              {        "x-dimension" = 6;
411                        "y-dimension" = 4
412                   }
413        },
414
415    The full encoding is in table 4.  A simplified view of the encoding looks like this:
```

416   **Table 4 - Overview Encoding of "media-col" collection**

Tag Value	Name	Value
begCollection	media-col	""
memberAttrName	""	media-color
keyword	""	blue
memberAttrName	""	media-size
begCollection	""	""
memberAttrName	""	x-dimension
integer	""	6
memberAttrName	""	y-dimension
integer	""	4
endCollection	""	""
endCollection	""	""

417

418

419   **Table 5 - Example Encoding of "media-col" collection**

Octets	Symbolic Value	Protocol field	comments
0x34	begCollection	value-tag	beginning of the "media-col" collection attribute
0x0009		name-length	length of (collection) attribute name
media-col	media-col	name	name of (collection) attribute
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)
0x4A	memberAttrName	value-tag	starts a new member attribute: "media-color"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf

Octets	Symbolic Value	Protocol field	comments
			no name (since name-length was 0)
0x000B		value-length	length of "media-color" keyword
media-color	media-color	value	value is name of 1st member attribute
0x44	keyword type	value-tag	keyword type
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0004		value-length	
blue	blue	value	value of 1st member attribute
0x4A	memberAttrName	value-tag	starts a new member attribute: "media-size"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x000A		value-length	length of "media-size" keyword
media-size	media-size	value	Name of 2nd member attribute
0x34	begCollection	value-tag	Beginning of the "media-size" collection attribute which is a sub-collection
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0000		value-length	collection attribute names have no value
			no value (since value-length was 0)
0x4A	memberAttrName	value-tag	starts a new member attribute: "x-dimension"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x000B		value-length	length of "x-dimension" keyword
x-dimension	x-dimension	value	name of 1st sub-collection member attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4

Octets	Symbolic Value	Protocol field	comments
0x0006		value	value of 1st sub-collection member attribute
0x4A	memberAttrName	value-tag	starts a new member attribute: "y-dimension"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x000B		value-length	length of the "y-dimension" keyword
y-dimension	y-dimension	value	name of 2nd sub-collection member attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4
0x0004		value	value of 2nd sub-collection member attribute
0x37	endCollection	value-tag	end of the sub-collection
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)
0x37	endCollection	value-tag	end of the 1st collection value in 1setOf
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)

8   Legacy issues

IPP 1.x Printers and Clients will gracefully ignore collections and its member attributes if it does not
understand the collection. The begCollection and endCollection elements each look like an attribute with an
attribute syntax that the recipient doesn't support and so should ignore the entire attribute. The individual
member attributes and their values will look like a 1setOf values of the collection attribute, so that the Printer

425   simply ignores the entire attribute and all of its values. Returning unsupported attributes is also simple, since
426   only the name of the collection attribute is returned with the 'unsupported' out-of-band value (see section 4.2).

9   IANA Considerations

428   This section contains the exact information for IANA to add to the IPP Registries according to the procedures
429   defined in "IPP/1.1 Model and Semantics" document [RFC2911] section 6.

430   *Note to RFC Editors: Replace RFC NNNN below with the RFC number for this document, so that it*
431   *accurately reflects the content of the information for the IANA Registry.*

9.1   Attribute Syntax Registration

433   The attribute syntax defined in this document will be published by IANA according to the procedures in RFC
434   2911 [RFC2911] section 6.3 with the following path:

435   ftp.isi.edu/iana/assignments/ipp/attribute-syntaxes/

436   The registry entry will contain the following information:

```
437   Reference:
438   RFC NNNN [this document]
439
440   Attribute Syntaxes:                                    Ref.        Section:
441   collection                                             RFC NNNN          3
442
```

10 Internationalization Considerations

444   This attribute syntax by itself has no impact on internationalization. However, the member attributes that are
445   subsequently defined for use in a collection may have internationalization considerations, as may any attribute,
446   according to [RFC2911].

11 Security Considerations

448   This attribute syntax causes no more security concerns than any other attribute syntax. It is only the attributes
449   that are subsequently defined to use this or any other attribute syntax that may have security concerns,
450   depending on the semantics of the attribute, according to [RFC2911].

12 References

451 [ipp-ntfy]
453 Isaacson, S., Martin, J., deBry, R., Hastings, T., Shepherd, M., Bergman, R. " Internet Printing
454 Protocol/1.0 & 1.1: IPP Event Notification Specification" draft-ietf-ipp-not-spec-02.txt, work in progress,
455 February 2, 2000.

456 [RFC2565]
457 Herriot, R., Butler, S., Moore, P., Tuner, R., "Internet Printing Protocol/1.0: Encoding and Transport",
458 RFC 2565, April 1999.

459 [RFC2566]
460 R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell, "Internet Printing Protocol/1.0: Model and
461 Semantics", RFC 2566, April 1999.

462 [RFC2567]
463 Wright, D., "Design Goals for an Internet Printing Protocol", RFC 2567, April 1999.

464 [RFC2568]
465 Zilles, S., "Rationale for the Structure and Model and Protocol for the Internet Printing Protocol", RFC
466 2568, April 1999.

467 [RFC2569]
468 Herriot, R., Hastings, T., Jacobs, N., Martin, J., "Mapping between LPD and IPP Protocols", RFC 2569,
469 April 1999.

470 [RFC2616]
471 R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer
472 Protocol - HTTP/1.1", RFC 2616, June 1999.

473 [RFC2910]
474 Herriot, R., Butler, S., Moore, P., Turner, R., "Internet Printing Protocol/1.1: Encoding and Transport",
475 draft-ietf-ipp-protocol-v11-05.txt, March 1, 2000.

476 [RFC2911]
477 Isaacson, S., deBry, R., Hastings, T., Herriot, R., Powell, P., "Internet Printing Protocol/1.1: Model and
478 Semantics", RFC 2911, September 2000.

13 Author's Addresses

480 Roger deBry
481 Utah Valley State College
482 Orem, UT 84058

483   Phone: (801) 222-8000
484   EMail: debryro@uvsc.edu
485
486   Tom Hastings
487   Xerox Corporation
488   737 Hawaii St. ESAE 231
489   El Segundo, CA 90245
490   Phone: 310-333-6413
491   Fax: 310-333-5514
492   e-mail: hastings@cp10.es.xerox.com
493
494   Robert Herriot
495   Xerox Corp.
496   3400 Hill View Ave, Building 1
497   Palo Alto, CA 94304
498   Phone: 650-813-7696
499   Fax: 650-813-6860
500   e-mail: robert.herriot@pahv.xerox.com
501
502   Kirk Ocke
503   Xerox Corp.
504   800 Phillips Rd
505   M/S 139-05A
506   Webster, NY 14580
507   Phone: (716) 442-4832
508   EMail: kirk.ocke@usa.xerox.com
509
510   Peter Zehler
511   Xerox Corp.
512   800 Phillips Rd
513   M/S 139-05A
514   Webster, NY 14580
515   Phone: (716) 265-8755
516   EMail: peter.zehler@usa.xerox.com

14 Appendix A: Encoding Example of a Simple Collection

518   The overall structure of the collection value can be pictorially represented as:

519   " media-size " =

520   {   "x-dimension" = 6;

521   "y-dimension" = 4

522   }

523

524   A simplified view of the encoding would look like this:

525   **Table 6 - Overview Encoding of simple collection**

Tag Value	Name	Value
begCollection	media-size	""
memberAttrName	""	x-dimension
integer	""	6
memberAttrName	""	y-dimension
integer	""	4
endCollection	""	""

526

527   Note: "" represents a name or value whose length is 0.

528

529   **Table 7 - Example Encoding of simple collection**

Octets	Symbolic Value	Protocol field	comments
0x34	begCollection	value-tag	beginning of the "media-size" collection attribute
0x000A		name-length	length of (collection) attribute name
media-size	media-size	name	name of (collection) attribute
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)
0x4A	memberAttrName	value-tag	starts member attribute: "x-dimension"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x000B		value-length	length of "x-dimension" keyword
x-dimension	x-dimension	value	name of 1st collection member attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4

Octets	Symbolic Value	Protocol field	comments
0x0006		value	value of 1st collection member attribute
0x4A	memberAttrName	value-tag	starts a new member attribute: "y-dimension"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x000B		value-length	length of the "y-dimension" keyword
y-dimension	y-dimension	value	name of 2nd collection member attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf for media-size
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4
0x0004		value	value of 2nd collection member attribute
0x37	endCollection	value-tag	end of the collection
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)

630

15 Appendix B: Encoding Example of 1setOf Collection

632   The overall structure of the collection value can be pictorially represented as:

633   "media-size-supported" =
634   {   "x-dimension" = 6;
635   "y-dimension" = 4
636   },
637   {   "x-dimension" = 3;
638   "y-dimension" = 5
639   };
640
641   A simplified view of the encoding would look like this:

542   **Table 8 - Overview Encoding of 1setOf collection**

Tag Value	Name	Value
begCollection	media-size-supported	""
memberAttrName	""	x-dimension
integer	""	6
memberAttrName	""	y-dimension
integer	""	4
endCollection	""	""
begCollection	""	""
memberAttrName	""	x-dimension
integer	""	3
memberAttrName	""	y-dimension
integer	""	5
endCollection	""	""

543

544   **Table 9 - Example Encoding of 1setOf collection**

Octets	Symbolic Value	Protocol field	comments
0x34	begCollection	value-tag	beginning of the "media-size-supported (1setOf collection" attribute
0x00014		name-length	length of (collection) attribute name
media-size-supported	media-size-supported	name	name of (collection) attribute
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)
0x4A	memberAttrName	value-tag	starts member attribute: "x-dimension"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x000B		value-length	length of "x-dimension" keyword
x-dimension	x-dimension	value	name of 1st collection member attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4
0x0006		value	value of 1st collection member attribute
0x4A	memberAttrName	value-tag	starts member attribute: "y-dimension"

Octets	Symbolic Value	Protocol field	comments
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x000B		value-length	length of the "y-dimension" keyword
y-dimension	y-dimension	value	name of 2nd collection member attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4
0x0004		value	value of 2nd collection member attribute
0x37	endCollection	value-tag	end of the collection
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)
0x34	begCollection	value-tag	beginning of the 2nd member of the 1SetOf "sizes-avail " collection attribute
0x0000		name-length	Zero length name indicates this is member of previous attribute
		name	no name (since name-length was 0)
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)
0x4A	memberAttrName	value-tag	starts member attribute: "x-dimension"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x000B		value-length	length of "x-dimension" keyword
x-dimension	x-dimension	value	name of 1st collection member attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4
0x0003		value	value of 1st collection member attribute
0x4A	memberAttrName	value-tag	starts member attribute: "y-dimension"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x000B		value-length	length of the "y-dimension" keyword

Octets	Symbolic Value	Protocol field	comments
y-dimension	y-dimension	value	name of 2nd collection member attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4
0x0005		value	value of 2nd collection member attribute
0x37	endCollection	value-tag	end of the 1setOf collection value
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)

645

16 Appendix C: Encoding Example of Collection containing 1setOf XXX attribute

648   The overall structure of the collection value can be pictorially represented as:

649   "wagons" =
650   {   "colors" = red, blue;
651   "sizes" = 4, 6, 8
652   }
653
654   A simplified view of the encoding would look like this:

655   **Table 10 - Overview Encoding of collection with 1setOf value**

Tag Value	Name	Value
begCollection	wagons	""
memberAttrName	""	colors
keyword	""	red
keyword	""	blue
memberAttrName	""	sizes
integer	""	4
integer	""	6
integer	""	8
endCollection	""	""

656

657   **Table 11 - Example Encoding of collection with 1setOf value**

Octets	Symbolic Value	Protocol field	comments
0x34	begCollection	value-tag	beginning of the "wagons" collection attribute
0x0005		name-length	length of (collection) attribute name
wagons	wagons	name	name of (collection) attribute
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)
0x4A	memberAttrName	value-tag	starts a new member attribute: "colors"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x0006		value-length	length of "colors" keyword
colors	colosr	value	value is name of 1st member attribute
0x44	keyword type	value-tag	keyword type
0x0000		name-length	0 indicates 1setOf wagons
			no name (since name-length was 0)
0x0004		value-length	
blue	blue	value	value of 1st member attribute
0x44	keyword type	value-tag	keyword type
0x0000		name-length	0 indicates 1setOf wagons
			no name (since name-length was 0)
0x0003		value-length	
red	red	value	value of 1st member attribute
0x4A	memberAttrName	value-tag	starts a new member attribute: "sizes"
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x0005		value-length	length of "length-avail" keyword
sizes	sizes	value	Name of 2nd member attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf wagons
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4
0x0004		value	1st value for 1SetOf integer attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf

Octets	Symbolic Value	Protocol field	comments
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4
0x0006		value	2nd value for 1SetOf integer attribute
0x21	integer type	value-tag	attribute type
0x0000		name-length	0 indicates 1setOf
			no name (since name-length was 0)
0x0004		value-length	length of an integer = 4
0x0008		value	3rd value for 1SetOf integer attribute
0x37	endCollection	value-tag	end of the collection
0x0000		name-length	defined to be 0 for this type, so part of 1setOf
			no name (since name-length was 0)
0x0000		value-length	defined to be 0 for this type
			no value (since value-length was 0)

658

17 Appendix D: Full Copyright Statement

659

660   Copyright (C) The Internet Society (1998,1999,2000,2001). All Rights Reserved

676