

1 INTERNET-DRAFT **There are 3 issues highlighted like this.**
2 <draft-ietf-ipp-collection-025.txt>

Roger deBry
Utah Valley State College
T. Hastings
Xerox Corporation
R. Herriot
Xerox Corporation
K. Ocke
Xerox Corporation
P. Zehler
Xerox Corporation
March 9, 2000

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

Copyright (C) The Internet Society (2000). All Rights Reserved.

16
17 Status of this Memo:

18 This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of
19 [RFC2026]. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its
20 areas, and its working groups. Note that other groups may also distribute working documents as Internet-
21 Drafts.

22 Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or
23 obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or
24 to cite them other than as "work in progress".

25 The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>

26 The list of Internet-Draft Shadow Directories can be accessed as <http://www.ietf.org/shadow.html>.

27 Abstract

28 This document specifies an OPTIONAL attribute syntax called 'collection' for use with the
29 Internet Printing Protocol/1.0 (IPP) [RFC2565, RFC2566], IPP/1.1 [ipp-mod, ipp-pro], and
30 subsequent versions. A 'collection' is a container holding one or more named values, which are
31 called "member" attributes. A collection allows data to be grouped like a PostScript dictionary or
32 a Java Map.

33 The full set of IPP documents includes:

- 34 Design Goals for an Internet Printing Protocol [RFC2567]
- 35 Rationale for the Structure and Model and Protocol for the Internet Printing Protocol [RFC2568]
- 36 Internet Printing Protocol/1.1: Model and Semantics (this document)
- 37 Internet Printing Protocol/1.1: Encoding and Transport [IPP-PRO]
- 38 Internet Printing Protocol/1.1: Implementer's Guide [IPP-IIG]
- 39 Mapping between LPD and IPP Protocols [RFC2569]

40

41 The "Design Goals for an Internet Printing Protocol" document takes a broad look at distributed printing
42 functionality, and it enumerates real-life scenarios that help to clarify the features that need to be included
43 in a printing protocol for the Internet. It identifies requirements for three types of users: end users,
44 operators, and administrators. It calls out a subset of end user requirements that are satisfied in IPP/1.0. A
45 few OPTIONAL operator operations have been added to IPP/1.1.

46 The "Rationale for the Structure and Model and Protocol for the Internet Printing Protocol" document
47 describes IPP from a high level view, defines a roadmap for the various documents that form the suite of
48 IPP specification documents, and gives background and rationale for the IETF working group's major
49 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
52 encoding rules for a new Internet MIME media type called "application/ipp". This document also defines
53 the rules for transporting over HTTP a message body whose Content-Type is "application/ipp". This
54 document defines a 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
56 implementers of IPP clients and IPP objects. It is intended to help them understand IPP/1.1 and some of the
57 considerations that may assist them in the design of their client and/or IPP object implementations. For
58 example, a typical order of processing requests is given, including error checking. Motivation for some of
59 the specification 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 **Table of Contents**

63	1	Problem Statement.....	4
64	2	Solution.....	4
65	3	Definition of a Collection Attribute	5
66	3.1	Member Attribute Naming Rules	5
67	3.2	Remaining rules for a collection attribute definition.....	6
68	3.3	Nested Collections.....	7
69	3.4	Collection Attributes as Operation Attributes	8
70	3.5	Collections as Job Template Attributes.....	8
71	3.6	Collections and Get-Printer-Attributes and Get-Job-Attributes operations	10
72	4	New Out-of-band value	11
73	4.1	'none'.....	11
74	5	Unsupported Values	11
75	6	Sample specification.....	11
76	7	Encoding.....	13
77	7.1	encoding of a collection (using solution 1a).....	17
78	7.2	Sample Encoding (using solution 1a).....	18
79	7.3	1setOf Collection encoding (using solution 1a).....	20
80	7.4	Sample 1setOf Collection encoding (using solution 1a).....	20
81	8	Legacy issues	22
82	9	IANA Considerations	22
83	10	Internationalization Considerations.....	22
84	11	Security Considerations.....	23
85	12	References	23
86	13	Author's Addresses	24
87	14	Appendix A: Full Copyright Statement.....	24
88			

89

90 1 Problem Statement

91 The IPP Model and Semantics [ipp-mod] supports most of the common data structures that are available in
92 programming languages. It lacks a mechanism for grouping several attributes of different types. The Java
93 language uses the Map to solve this problem and PostScript has a dictionary. [The new mechanism for](#)
94 [grouping attributes together must allow for optional members and subsequent extension of the collection.](#)

95 The mechanism must be encoded in a manner consistent with existing 1.0 and 1.1 parsing rules (see [ipp-
96 pro]). Current 1.0 and 1.1 parsers that don't support collections should not confuse collections they receive
97 with attributes that they do support.

98 2 Solution

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

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

108 As with any attribute syntax, the collection attribute definition document specifies whether the attribute is
109 single-value (collection) or multi-valued (1setOf collection).

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

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

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

119 A collection value MUST NOT contains two or more member attributes with the same attribute name.
120 Such a collection is mal-formed. Clients MUST NOT submit such malformed requests and Printers MUST

121 NOT return such malformed responses. If such a malformed request is submitted to a Printer, the Printer
122 MUST reject the request with the 'client-error-bad-request' status code (see section 13.1.4.1)

123 ISSUE 01: In attribute groups [ipp-mod] allows a Printer either (1) to reject a request with duplicate named
124 attributes OR (2) to choose exactly one of the attributes as the one to be used. Should we REQUIRE the
125 Printer to reject duplicate named attributes in a collection value as stated above or allow the Printer to
126 choose one member attribute as a second alternative as we do with attribute groups?

127 3 Definition of a Collection Attribute Type

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

129 3.1 Member Attribute Naming Rules

130 Each collection attribute MUST have a globally-unique name within the scope in which the collection
131 attribute occurs. If the collection attribute occurs as a member of a request or response attribute group, it
132 MUST be unique within that group, same as for any other attribute. If a collection attribute occurs as a
133 member attribute of another collection, the collection attribute MUST have a unique name within that
134 collection value, same as for any other attribute.

135 Each member attribute in a collection value MUST have unique name within that collection value.
136 Member attribute names MAY be reused between different collection attributes. An example is the
137 "media" attribute which MAY be used as a job template attribute (see [ipp-mod]) and in a collection. All
138 attribute names that are reused MUST have an identical syntax. All attribute names that are reused MUST
139 have a similar semantics. The semantic difference MUST be limited to boundary conditions and constraints
140 placed on the reused attributes. All attributes that are not reused from elsewhere in the IPP model MUST
141 have a globally unique name.

142 Assume that it is desirable to extend IPP by adding a Job Template attribute that allows the client to select
143 the media by its properties, e.g., weight, color, size, etc., instead of by name as the "media (type3 keyword |
144 name) Job Template attribute in IPP/1.1 (see [ipp-mod]). The first rule is that the existing attribute MUST
145 NOT be extended by adding the 'collection' attribute syntax to the existing "media" attribute. That would
146 cause too many interoperability problems and complicates the validation and defaulting rules as well.
147 Instead, a new attribute will be defined with a suffix of "-col" (for collection), e.g., "media-col" (collection).

148 For a second example, suppose it is desirable to extend IPP by allowing the client to select the media for the
149 job start sheet. Again, this would not be done by adding the 'collection' attribute syntax to the existing "job-
150 sheets" (type2 keyword | name) Job Template attribute. Instead, a new "job-sheet-col" (collection) Job
151 Template attribute MUST be introduced. The member of the "job-sheet-col" collection might be:

152 "job-sheet-format" (type3 keyword | name)

153 "media" (type3 keyword | name)

154 if any of the "media-supported" (1setOf (type3 keyword | name)) Printer attribute values could be specified
155 for job sheets. The reason that the "job-sheet-format" member attribute isn't named simply, "job-sheet", is

156 because its values only indicate the format, and don't imply any media, while the "job-sheets" (type2
 157 keyword | name) Job Template attribute do imply a media. This example illustrates when a member
 158 attribute can be the same as another attribute (in this case a Job Template attribute) and when the member
 159 attribute MUST have a different name.

160 If the definers of the "job-sheet-col" (collection) attribute intended that the System Administrator be
 161 allowed to have a different set of media values for job sheets than documents, then the definition document
 162 for the "job-sheet-col" collection attribute would have the following member attributes instead:

163 "job-sheet-format" (type3 keyword | name)

164 "job-sheet-media" (type3 keyword | name)

165 Then the supported values would be include in a separate "job-sheet-media-supported" (1setOf (type3
 166 keyword | name)) Printer attribute.

167 **3.2 Remaining rules for a Specification of a collection attribute definition**

168 When a specification document defines an "xxx" collection attribute "~~xxx~~", i.e., an attribute whose
 169 attribute syntax type is 'collection' or '1setOf collection'; ~~it must define~~ the definition document MUST
 170 include the following aspects of the attribute semantics. Suppose T the "xxx" collection attribute contains
 171 an "aaa" member attribute "~~aaa~~". A simplified example of a collection specification is given in section 6

- 172 1. The name of the collection attribute MUST be specified. (e.g. "xxx")
- 173 2. The collection attribute syntax MUST be of type 'collection' or '1setOf collection'.
- 174 3. The context of the collection attribute MUST be specified, i.e., whether the attribute is an operation
 175 attribute, a Job Template attribute, a Job Description attribute, a Printer Description attribute, a
 176 member attribute of a particular collection attribute, etc.
- 177 4. The member attributes MUST be defined. For each member attribute the definition document
 178 MUST provide the following ~~MUST be provided~~:-
 - 179 a) The member attribute's name, "aaa", MUST either (1) reuse the attribute name of another
 180 attribute if the member attribute shares the syntax and semantics with the other attribute or (2)
 181 be unique across the entire IPP attribute name space
 - 182 b) Whether the member attribute is REQUIRED or OPTIONAL for the Printer to support
 - 183 c) Whether the member attribute is REQUIRED or OPTIONAL for the client to supply in a request
 - 184 d) The member attribute's syntax type, which can be any attribute syntax, including '1setOf X',
 185 'collection', and '1setOf collection'. If this attribute name is the same as another attribute (case of
 186 option a-1 above), it MUST have the same attribute syntax, including cardinality (1setOf or
 187 not)~~This MAY be expressed with a reference to the associated attribute in the case of option a-1~~
 188 above.

- 189 e) The semantics of the "aaa" member attribute. The semantic definition ~~should~~MUST include a
 190 description of any constraint or boundary conditions the member attribute places on the
 191 associated attribute, especially if the attribute is the same as another attribute used in a different
 192 context (case of option a-1 above)
- 193 f) ~~its~~the supported values for the "aaa" member attribute, either enumerated explicitly or specified
 194 by the values of a referenced attribute which may be specified by either:
- 195 – the attribute's definition
 - 196 – a Printer attribute, such as "yyyaaa-supported", which contains the explicit values
 197 supported. The "yyyaaa-supported" attribute is a Printer attribute and not in a collection.
 198 For example, if a collection contains the "media" attribute and its supported values are
 199 specified by the "media-supported" attribute, the "media-supported" attribute is the same
 200 Printer attribute that the "media" attribute uses.
- 201 g) the default value of "yyyaaa" member attribute if it is OPTIONAL for a client to supply the
 202 "yyyaaa" member attribute in a request. The default value is specified by either:
- 203 – the attribute's definition
 - 204 – a Printer attribute, such as "yyyaaa-default", which may have a collection value
 205 – or an implementation defined algorithm that takes into account the values of the other
 206 member attributes of the collection value
- 207 h) ~~For any member attribute of a job template collection the syntax of "aaa-supported" MUST be~~
 208 ~~specified. See section a) below.~~Depending on the collection attributes context, it MUST follow
 209 the additional rules specified below for the various contexts.

210 3.3 Nested Collections

211 A member attribute may have a syntax type of 'collection' or '1setOf collection'. The following example
 212 assumes a "yyy" collection "yyy"attribute is a member attribute of the preceding ~~collection~~"xxx"collection
 213 attribute. The "yyy" collection attribute contains ~~an attribute~~"bbb"member attribute. The definition
 214 document for the nested collection ~~proceeds as follows~~MUST include:-

- 215 1. The name of the collection attribute, e.g., "yyy"
- 216 2. The collection attribute syntax MUST be of type 'collection' or '1setOf collection'
- 217 3. The member attributes MUST be defined. For each member attribute the definition document MUST
 218 provide the following:~~For each member attribute the following MUST be provided~~

- 219 a) The member attribute's name, "bbb", MUST either (1) reuse the attribute name of another attribute if
220 the member attribute shares the syntax and semantics with the other attribute or (2) be unique across
221 the entire IPP attribute name space
- 222 b) Whether the member attribute is REQUIRED or OPTIONAL for the Printer to support
- 223 c) Whether the member attribute is REQUIRED or OPTIONAL for the client to supply in a request
- 224 d) The member attribute's syntax type, which can be any attribute syntax, including '1setOf X',
225 'collection', and '1setOf collection'. If this attribute name is the same as another attribute (case of
226 option a-1 above), it MUST have the same attribute syntax, including cardinality (1setOf or not).~~This~~
227 ~~MAY be expressed with a reference to the associated attribute in the case of option a-1 above.~~
- 228 e) The semantics of the member attribute. The semantic definition ~~should~~ MUST include a description
229 of any constraint or boundary conditions the member attribute places on the associated attribute,
230 especially if the attribute is the same as another attribute used in a different context (case of option
231 a-1 above)
- 232 f)
- 233 g) Depending on the collection attributes context, it MUST follow the additional rules specified below
234 for the various contexts.~~For any member attribute of a job template collection the syntax of "bbb-~~
235 ~~supported" MUST be specified. See section a) below.~~

236 **3.4 Collection Attributes as Operation Attributes**

237 The definition documents that define a collection attribute for use as an operation attribute MUST follow
238 these additional rules:

- 239 a) Define in which operation requests the collection attribute is intended to be used.
- 240 b) Define in which operation responses the collection attribute is intended to be used.

241 **3.5 Collections as Job Template Attributes**

242 The definition documents for collection attributes that are specified to be ~~Collections that are j~~Job
243 †Template attributes (see [ipp-mod] section 4.2) MUST have associated printer attributes with suffixes of "-
244 supported" and "-default" (or indicate that there is no "-default"), just as for any Job Template attribute.
245 Certain Job Template collection attributes also have an associated Printer attribute with "-ready" (for
246 example, see the "media-ready" attribute in [ipp-mod]). ~~The attributes with "-ready" are explicitly called~~
247 ~~out in the IPP Model and Semantics specification.~~ Furthermore member attributes of job template
248 attributes are addressed using the same suffix convention.

249 See also section 3.6 on the interaction of collections and the Get-Printer-Attributes and Get-Jobs-Attributes.

- 250 For the following rules assume the "xxx" (collection) example from section 3.2 is a job template attribute.
- 251 1) There ~~are two~~MUST be two associated printer attributes. The attributes are "xxx-supported" and "xxx-
252 default"
- 253 2) The "xxx-default" is a collection with a syntax identical to the "xxx" specification in section 3.2 .
- 254 – Each member attribute has the same name as in the "xxx" definition.
- 255 – A Get-Printer-Attributes operation MUST return the "xxx-default" (collection) Printer attribute
256 and all the member attributes. Any default values that have been set MUST be returned. Any
257 default values that have not been set MUST return an out of band attribute of 'no-value'.
- 258 3. If the definition of the collection does not mention an "xxx-ready" attribute than it is assumed that one
259 is not defined, though implementer's are free to support an "xxx-ready" as an extension.
- 260 4. The collection attribute definition document MUST define an "xxx-supported" an-attribute with either a
261 syntax of '1setOf type2 keyword' or '1setOf collection':
- 262 – If the definition uses the '1setOf type2 keyword' attribute syntax, it MUST be the attribute
263 keyword names of all of the member attributes that the Printer implementation supports in a Job
264 Creation operation. Furthermore, the definition MUST include corresponding definitions of
265 each of the "aaa-supported" attributes that correspond to each "aaa" member attribute. Then a
266 client can determine the supported values of each member attribute in the Job Template
267 collection attribute
- 268 – If the definition uses the '1setOf collection' attribute syntax, then the values are the supported
269 instances of the "xxx" (collection) attribute that a client can supply in a Job Creation operation.
270 It is expected that this second approach will be used for small collections whether the number of
271 possible collection values is small. For example, a "media-size" (collection) member attribute in
272 which the member attributes are "x-dimension" (integer) and "y-dimension" (integer). The pairs
273 of integers are just like keywords as far as the client localization is concerned, except that if the
274 client doesn't recognize a size pair of numbers, it can display the numbers.
- 275 ~~ISSUE-03—For certain small collections where all member attributes MUST be supplied and supported,~~
276 ~~such as "media.size" (collection) where the collection is "media.size.x" and "media.size.y", it would be~~
277 ~~useful to allow the "xxx-supported" (1setOf collection) to show the possible combinations of x and y~~
278 ~~dimensions. Thus this rule should be amended to allow either form in a definition. The pairs of integers are~~
279 ~~just like keywords as far as the client localization is concerned, except that if the client doesn't recognize a~~
280 ~~size pair of numbers, it can display the numbers.~~
- 281 a) The keywords returned lists all the contained member attribute names. This example would return
282 the "aaa" keyword.

- 283 b) The list is recursive and lists all the member attributes of the contained collections. In section 3.3
284 the printer would return "aaa" and "bbb" for collection "xxx"
- 285 c) The encoding convention allows the reconstruction of the collection structure. The will allow the
286 client to reconstruct the collections. The client would know that "aaa" is a member of collection
287 "xxx". It can also be derived that collection "bbb" is a member of collection "yyy". See section 7
288 for more information on encoding.
- 289 d) To obtain the supported values for any member attribute a client performs a Get-Printer-Attributes
290 operation explicitly requesting the member attribute name with the suffix "supported". If a member
291 attribute is itself a collection rule 4 above applies to member attribute.

292 **3.6 Collections and Get-Printer-Attributes and Get-Job-Attributes operations**

293 The behavior of collections for "job-description" and "printer-description" is similar to any other attribute.
294 Simple attributes return the attribute and its value. For a collection, the collection and its entire member
295 attributes and their values are returned. This includes any containing collections, its member attributes and
296 their values. The same logic applies for the "-default" and "-ready" printer attribute associated with a job-
297 template attributes.

298 Whether the Printer applies individual member attributes independently or takes into account the member
299 attributes supplied by the client in the collection, depends on implementation. Therefore, a client SHOULD
300 query the Printer's "xxx-default" (collection) attribute, allow the user to make any changes, and then submit
301 the entire collection to the Printer. Then the variability in defaulting between different implementations
302 will not cause the user to get unexpected results.

303 The semantics for "-supported" is different for a collection. Here the focus is on the member attributes that
304 the collection supports. This solution allows for extension of collections and allowing the member
305 attributes of a collection to vary (i.e. mandatory and optional member attributes). Once a client determines
306 what member attributes are supported in a collection a subsequent request can be constructed to determine
307 the supported values for the member attributes.

308 Another advantage of that the behavior of the "-supported" printer collection attribute is limiting the amount
309 of data that is returned on general queries. A 'get-printer-attributes' that returns all the attributes of a printer
310 will not have to return what may turn out to be extensive lists of "-supported" attribute values. An example
311 might be "media-col" that could be a representation for media using a collection that goes beyond the
312 information currently provided by the job-template attribute "media". The "media-col" could now be used
313 to represent a job's media, insert sheets and inserted tab sheets. An IPP Printer implementation would
314 return the member attributes for each of the "-supported" collections.

315 4 New Out-of-band value

316 4.1 'none'

'none'	The specified Job Template attribute in the request MUST NOT be applied to the job. Specifically, this value overrides the Printer's "xxx-default" attribute value for the Job Template attribute, if one exists.
--------	---

317 This "out-of-band" value allows a client to specify "turn-off" a feature that is specified by an attribute
 318 whose value is a collection. Because a client specifies a value, the Printer uses the client-specified value and
 319 not the Printer's default value.

320 If a Printer supports the use of the 'collection' attribute syntax for an attribute, a Printer MUST support the
 321 use of the "out-of-band" value 'none'.

322 A Printer MUST support the "out-of-band" value 'none' as the value for an attribute "xxx" if:

- 323 – the definition of the attribute specifies 'none' MUST be supported AND
- 324 – the definition of the attribute specifies 'none' MAY be supported and it is a value of the attribute
 325 "xxx-supported".

326 5 Unsupported Values

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

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

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

336 6 Sample specification

337 This example is for a collection called "media-col". The "media-col" attribute is a job template attribute.
 338 This collection is simplified and fictitious and is used for illustrative purposes only.

339 Name: media~~col~~

340 Syntax: collection

341 Member Attributes:

342 Name: "media-color"

343 Syntax: type3 keyword | name

344 Mandatory

345 Semantics: This attribute identifies the color of the media. Valid values are "red" "white" and
346 "blue"

347 "media-color-supported" syntax: 1setOf (type2 keyword | name)

348 Name: "media-size"

349 Syntax: collection

350 Member Attributes:

351 Name: "x-dimension"

352 Syntax: integer

353 Mandatory

354 Semantics: This attribute identifies length of the media in inches. Valid values are any
355 integer though in practice implementation will constrain the range.

356 x-supported syntax: rangeOfInteger

357 Name: "y-dimension"

358 Syntax: integer

359 Mandatory

360 Semantics: This attribute identifies the width of the media in inches. Valid values are any
361 integer though in practice implementation will constrain the range.

362 y-supported syntax: rangeOfInteger

363 Name: name

364 Syntax: See job template attribute "media"

365 Optional

366 Semantics: See job template attribute "media". Additional restrictions on "media" in this collection
367 are that the "media" value must be valid based on the size and color. When invalid names are given
368 based on the size or color, the size or color value takes precedence.

369 Supported values identical to job template attribute "media-supported".

370

371 7 Encoding

372 This section is still under construction.

373 We are now down to considering two encodings for collections. The goals of the encoding are:

374 a) must be simple

375 b) a legacy receiver must correctly ignore a collection value and not incorrectly decode part of a
376 collection as a legitimate attribute.

377 c) it parses an attributes with collection values as a single unknown attribute rather than as
378 many unknown attributes.

379 The two encodings are:

380 1) encode attributes within collections in the same way as attributes outside of collections,
381 but encode each attribute name in a collection so that its name cannot be the same as an
382 attribute name outside of a collection. We have considered two solutions for encoding
383 attribute names.

384 a) add a prefix to each collection member attribute name where the prefix is the
385 (outer) attribute's name following by a dot ("."). Nested collections have extra levels
386 of dotted names. For example, the "media-size" attribute in "media-col" is encoded
387 as "media-col.media-size" and the "x" attribute in "media-size" which is inside
388 "media" is encoded as "media-col.media-size.x". The outer attribute name is the
389 "name" of the begin-collection and end-collection value.

390 b) add a hyphen suffix to each attribute name in a collection. For example, the
391 "media-size" attribute in "media-col" is encoded as "media-size-" and the "x"
392 attribute in "media-size" which is inside "media" is encoded as "x-". Note the
393 hyphen must be a suffix so that the attribute name follows the rules for a legal
394 keyword, and the hyphen is chosen because no attributes currently end with a
395 hyphen. The empty name is used for the end-collection value and all but the first
396 begin-collection value.

397 2) encode attributes within a collection as a 1setOf values where each attribute whose
 398 name is M and whose values are V1 ... Vn are encoded as a sequence of n+1 values M,
 399 V1, ... Vn. Subsequent member attributes continue the value in the 1setOf values.

400 **ISSUE 02: Which encoding do we want to use for collections, 1a, 1b, or 2?**

401 The following are examples of encodings. In the real encoding, each "attribute" consists of

402 a) a one byte tag

403 b) a two byte name length whose value is "n"

404 c) "n" bytes of a name

405 d) a two bytes value length whose value is "v"

406 e) "v" bytes of a value

407 To make it easy to read, we show only items c (the name), a (the tag) and e (the value), in that
 408 order.

409 There are 3 encoding examples for each solution:

410 i) media-col with media-color and media-size as member attributes, and where media-size
 411 contains "x" and "y" as collection members.

412 ii) media-size-supported with two collection values.

413 iii) job-notify with notify-recipients and notify-events which is a 1setOf keyword with 3 values in
 414 this example

415 Solution 1a)

416

Name	syntax-type	value
"media-col "	begin-collection	" "
"media-col.media-color "	keyword	white
"media-col.media-size "	begin-collection	" "
"media-col.media-size.x "	integer	850
"media-col.media-size.y "	integer	1100
"media-col.media-size "	end-collection	" "
"media-col "	end-collection	" "

425

Name	syntax-type	value
"media-size-supported "	begin-collection	" "
"media-size-supported.x "	integer	850
"media-size-supported.y "	integer	1100

429

```

430 "media-size-supported"          end-collection      ""
431 "media-size-supported"          begin-collection    ""
432 "media-size-supported.x"        integer             850
433 "media-size-supported.y"        integer             1400
434 "media-size-supported"          end-collection      ""
435
436 Name                            syntax-type         value
437 "job-notify"                     begin-collection    ""
438 "job-notify.notify-recipients"   url                 "mailto://bill@foo.com"
439 "job-notify.notify-events"       keyword             job-completed
440 " "                               keyword             job-created
441 " "                               keyword             job-state-changed
442 "job-notify"                     end-collection      ""
443
444

```

Solution 1b)

```

447 Name                            syntax-type         value
448 "media-col"                      begin-collection    ""
449 "media-color-"                   keyword             white
450 "media-size-"                    begin-collection    ""
451 "x-"                              integer             850
452 "y-"                              integer             1100
453 "media-size-"                    end-collection      ""
454 " "                               end-collection      ""
455

```

```

456 Name                            syntax-type         value
457 "media-size-supported"           begin-collection    ""
458 "x-"                              integer             850
459 "y-"                              integer             1100
460 " "                               end-collection      ""
461 " "                               begin-collection    ""
462 "x-"                              integer             850
463 "y-"                              integer             1400
464 " "                               end-collection      ""
465

```

```

466 Name                            syntax-type         value
467 "job-notify"                     begin-collection    ""
468 "notify-recipients-"             url                 "mailto://bill@foo.com"
469 "notify-events-"                 keyword             "job-completed"
470 " "                               keyword             "job-created"
471 " "                               keyword             "job-state-changed"
472 "job-notify"                     end-collection      ""
473
474

```

Solution 2)

```

477 Name                            syntax-type         value

```



```

478     "media-col"          begin-collection      ""
479     " "                  attribute-name    "media-color"
480     " "                  keyword          white
481     " "                  attribute-name    "media-size"
482     " "                  begin-collection  ""
483     " "                  attribute-name    "x"
484     " "                  integer          850
485     " "                  attribute-name    "y"
486     " "                  integer          1100
487     " "                  end-collection   ""
488     " "                  end-collection   ""

```

```

489
490     Name                  syntax-type      value
491     "media-size-supported" begin-collection  ""
492     " "                  attribute-name    "x"
493     " "                  integer          850
494     " "                  attribute-name    "y"
495     " "                  integer          1100
496     " "                  end-collection   ""
497     " "                  begin-collection  ""
498     " "                  attribute-name    "x"
499     " "                  integer          850
500     " "                  attribute-name    "y"
501     " "                  integer          1400
502     " "                  end-collection   ""

```

```

503
504     Name                  syntax-type      value
505     "job-notify"         begin-collection  ""
506     " "                  attribute-name    "notify-recipients"
507     " "                  url              mailto://bill@foo.com"
508     " "                  attribute-name    "notify-events"
509     " "                  keyword          "job-completed"
510     " "                  keyword          "job-created"
511     " "                  keyword          "job-state-changed"
512     " "                  end-collection   ""
513

```

514

515

Observations:

516 Solution 1a have identical properties to solution 1b except that the rules for encoding the name
517 are more complicated for 1a, and the name of the attribute appears before each end-collection
518 and end-collection in 1a but only before the first begin-collection in 1b.

519 If a collection aware client sends a collection to a collection unaware Printer:

520 For solutions 1a and 1b) the Printer sees many attributes in place of the collection and it returns
521 in the Unsupported attribute group, all of the attributes: the attribute outside the collection and
522 each attribute in the collection with it altered name. Thus the unsupported attributes have names
523 that the client didn't send and they may be in an order that makes it hard to reconstruct the
524 collection. In addition, because the "end-collection" has the same name as the attribute for 1a,
525 some printers will reject the job because the attribute appears twice. Also, 1a does not work for a
526 1setOf collection because the name of the attributes appear in front of each begin-collection and
527 thus cannot be distinguished from two occurrences of the same attribute.

528 For solution 2) the Printer sees the collection as a 1setOf values where some values have
529 unknown syntax types and other values have known syntax types. When a collection-unaware
530 printer discovers it doesn't understand an attribute that is a collection, it sees the unknown
531 attribute as a 1setOf rather than a collection. It still returns the attribute-name with the out-of-
532 band value "unsupported" making it easier for the client.

533

534

535 ***7.1 encoding of a collection (using solution 1a)***

536 NOTE: If we pick another solution to the encoding, this section will change.

537 Each collection MUST have a globally unique name. Each attribute in an attribute group or a collection
538 MUST have globally unique name. Uniqueness is generated by prepending the collection name to the
539 attribute using a period, '.' as a separator.

540 For encoding attributes that have a 'collection' attribute syntax, the attribute's name is REQUIRED to be the
541 first part of each of the member attribute name separated by a PERIOD (.) character. For example, if a
542 "media-col" (collection) Job Template attribute is added to IPP and contains a member attribute "color, it
543 MUST be encoded as a "media-col.color". In another example, if the "job-sheets" (collection) Job
544 Template attribute is added to IPP and reuses the "color" member attribute, the "color" attribute MUST be
545 encoded as "job-sheets.color". The "xxx.color" attribute has an identical attribute syntax and similar
546 semantics.

547 When encoding a collection attribute "xxx" that contains an attribute "aaa". A simplified example of a
548 collection specification is given in section 6

549 1. The beginning of the collection is indicated with a value tag that MUST be syntax type 'begincollection'
550 (e.g. 0x34).

551 2. The length of the collection name (e.g. 0x03)

552 3. The collection name (e.g. "xxx")

- 553 4. A null collection value length (e.g. 0x00)
- 554 5. The attributes are encoded as with any other attribute. It is valid to have a collection a member of a
555 collection. The modifications necessary for encoding member attributes of a collection are as follows.
- 556 a) The name of the member attribute MUST be prepended with the collection name and a period.
- 557 b) The length of the member attribute name MUST be adjusted appropriately.
- 558 6. The end of the collection is indicated with a value tag that MUST be syntax type 'endCollection' (e.g.
559 0x37).
- 560 7. The length of the collection name (e.g. 0x03)
- 561 8. The collection name (e.g. "xxx")
- 562 9. A null collection value length (e.g. 0x00)
- 563

564 **7.47.2 Sample Encoding (using solution 1a)**

565 **NOTE: If we pick another solution to the encoding, this section will change.**

566 This section defines the encoding of a collection syntax type using solution 1a. The collection specified in
567 section 6 is used. The encoding is of an implementation that does not support any optional attributes. A
568 collection is encoded by using two new tags:

Tag name	Tag value	Meaning
beginCollection	0x34	Begin the named collection.
endCollection	0x37	End the named collection.

569 A collection value is encoded as a sequence of attribute values preceded by a beginCollection attribute and
570 followed by an endCollection attribute. The name field of a beginCollection and an endCollection both
571 contain the name of the collection type, i.e., the keyword name of the collection attribute, which is a string
572 of ASCII characters. The value field contains the prefix used for all subordinate member attributes. The
573 following example is written in the style of the IPP/1.1 "Encoding and Transport" document [ipp-pro]. The
574 following example is for a media collection attribute. The media collection contains 2 member attributes.
575 One member is "color" that contains a keyword for the media's color. The second attribute is a collection
576 that gives the media's size. The size collection has two integer attributes "x" and "y" that gives the media's
577 size in inches

Octets	Symbolic Value	Protocol field	comments
--------	----------------	----------------	----------

Octets	Symbolic Value	Protocol field	comments
0x34	beginCollection	value-tag	Beginning of the collection
0x0009		name-length	Length of collection's name
media-col		Name	Collection's name
0x0000		Value-length	
0x44	keyword type	value-tag	Member attribute type
0x000F		name-length	Length of member attribute name
media-col.color		Name	Name of member attribute
0x0004		value-length	
blue	blue	Value	
0x34	beginCollection	value-tag	Beginning of the sub-collection
0x000E		name-length	Length of sub-collection's name
media-col.size		Name	Sub-collection's name
0x0000		Value-length	
0x21	integer type	value-tag	Member attribute type
0x0010		name-length	Length of member attribute name
media-col.size.x		Name	Name of member attribute
0x0004		value-length	
0x0006		Value	
0x21	integer type	value-tag	Member attribute type
0x0007		name-length	Length of member attribute name
media-col.size.y		Name	Name of member attribute
0x0004		value-length	
0x0004		Value	
0x37	endCollection	value-tag	end of the sub-collection
0x0007		name-length	Length of sub-collection's name
media-col.size		Name	Sub-collection's name
0x0000		Value-length	
0x37	endCollection	value-tag	end of the collection
0x0007		name-length	Length of collection's name
media-col		Name	Sub-collection's name
0x0000		Value-length	

578 **7.27.3 1setOf Collection encoding (using solution 1a)**

579 The encoding of a set of collections follows the standard method of encoding multi-valued IPP attributes.
 580 The "beginCollection" attribute is coded normally. The first instance of the collection follows. The
 581 "endCollection" MUST appear only once in a collection and MUST follow the last member of the set of
 582 collection. The member collections of a set of collections are delineated by a specially encoded
 583 "beginCollection" attribute. The type MUST be "beginCollection" (i.e. 0x34). The length of the name field
 584 MUST be 0x0000. The name field MUST be omitted. The length of the value MUST be the length of the
 585 collection's prefix. The value MUST be the prefix.

586 **7.37.4 Sample 1setOf Collection encoding (using solution 1a)**

587 NOTE: If we pick another solution to the encoding, this section will change.

588 This section defines the encoding of a collection syntax type using solution 1a. The collection specified in
 589 section 7 is used. The difference is that the type of "media-col" is 1setOf collection instead of collection.
 590 The encoding is of an implementation that does not support any optional attributes.

591

Octets	Symbolic Value	Protocol field	comments
0x34	beginCollection	value-tag	Beginning of the collection
0x0009		name-length	Length of collection's name
media-col	media-col	Name	Collection's name
0x0000		Value-length	
0x44	keyword type	value-tag	Member attribute type
0x000F		name-length	Length of member attribute name
media-col.color	media-col.color	Name	Name of member attribute
0x0004		value-length	
blue	blue	Value	
0x34	beginCollection	value-tag	Beginning of the sub-collection
0x000E		name-length	Length of sub-collection's name
media-col.size	media-col.size	Name	Sub-collection's name
0x0000		Value-length	
0x21	integer type	value-tag	Member attribute type
0x00010		name-length	Length of member attribute name
media-col.size.y	media-col.size.y	Name	Name of member attribute
0x0004		value-length	
0x0006		Value	
0x21	integer type	value-tag	Member attribute type

Octets	Symbolic Value	Protocol field	comments
0x00010 media-col.size.x 0x0004 0x0004	media-col.size.x	name-length Name value-length Value	Length of member attribute name Name of member attribute
0x37 0x000E media-col.size 0x0000	endCollection media-col.size	value-tag name-length Name Value-length	end of the sub-collection Length of sub-collection's name Sub-collection's name
			Second collection in set
0x34 0x0000 0x0000	beginCollection	value-tag name-length Value-length	Beginning of the collection Indicates continuation of set
0x44 0x000F media-col.color 0x0003 red	keyword type media-col.color red	value-tag name-length Name value-length Value	Member attribute type Length of member attribute name Name of member attribute
0x34 0x000E media-col.size 0x0000	beginCollection media-col.size	value-tag name-length Name Value-length	Beginning of the sub-collection Length of sub-collection's name Sub-collection's name
0x21 0x0010 media-col.size.y 0x0004 0x0006	integer type media-col.size.y	value-tag name-length Name value-length Value	Member attribute type Length of member attribute name Name of member attribute
0x21 0x0010 media-col.size.x 0x0004 0x0004	integer type media-col.size.x	value-tag name-length Name value-length Value	Member attribute type Length of member attribute name Name of member attribute
0x37	endCollection	value-tag	end of the sub-collection

Octets	Symbolic Value	Protocol field	comments
0x000E media-col.size 0x0000	media-col.size	name-length Name Value-length	Length of sub-collection's name Sub-collection's name
0x37 0x0009 media-col 0x0000	endCollection media-col	value-tag name-length Name Value-length	end of the set of collections Length of collection's name collection's name Length of collection's prefix

592

593 8 Legacy issues

594 IPP 1.x Printers and Clients will gracefully ignore collections and its member attributes if it does not
 595 understand the collection. The begCollection and endCollection elements each look like an attribute with
 596 an attribute syntax that the recipient doesn't support and so should ignore the entire attribute. The
 597 individual member attributes will look like ordinary attributes, but since they each are encoded with a
 598 unique name that can't be the same as a top level attribute, each of the member attributes will also look like
 599 attributes that the recipient doesn't support and so should ignore.

600 9 IANA Considerations

601 This attribute syntax will be registered with IANA after the WG approves its specification according to the
 602 procedures for extension of the IPP/1.1 Model and Semantics [ipp-mod].

603 **ISSUE 03 - Since this is intended to be a standards track document, do we also register the attribute syntax**
 604 **with IANA?**

605 10 Internationalization Considerations

606 This attribute syntax by itself has no impact on internationalization. However, the member attributes that
 607 are subsequently defined for use in a collection may have internationalization considerations, as may any
 608 attribute, according to [ipp-mod].

609 11 Security Considerations

610 This attribute syntax causes no more security concerns than any other attribute syntax. It is only the
611 attributes that are subsequently defined to use this or any other attribute syntax that may have security
612 concerns, depending on the semantics of the attribute, according to [ipp-mod].

613 12 References

614 [ipp-mod]

615 Isaacson, S., deBry, R., Hastings, T., Herriot, R., Powell, P., "Internet Printing Protocol/1.1: Model
616 and Semantics" draft-ietf-ipp-model-v11-06.txt, March 1, 2000.

617 [ipp-ntfy]

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

621 [ipp-pro]

622 Herriot, R., Butler, S., Moore, P., Turner, R., "Internet Printing Protocol/1.1: Encoding and
623 Transport", draft-ietf-ipp-protocol-v11-05.txt, March 1, 2000.

624 [RFC2565]

625 Herriot, R., Butler, S., Moore, P., Tuner, R., "Internet Printing Protocol/1.0: Encoding and
626 Transport", RFC 2565, April 1999.

627 [RFC2566]

628 R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell, "Internet Printing Protocol/1.0: Model and
629 Semantics", RFC 2566, April 1999.

630 [RFC2567]

631 Wright, D., "Design Goals for an Internet Printing Protocol", RFC 2567, April 1999.

632 [RFC2568]

633 Zilles, S., "Rationale for the Structure and Model and Protocol for the Internet Printing Protocol",
634 RFC 2568, April 1999.

635 [RFC2569]

636 Herriot, R., Hastings, T., Jacobs, N., Martin, J., "Mapping between LPD and IPP Protocols", RFC
637 2569, April 1999.

638 [RFC2616]

639 R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext
640 Transfer Protocol - HTTP/1.1", RFC 2616, June 1999.

641 13 Author's Addresses

642 Roger deBry
643 Utah Valley State College
644 Orem, UT 84058
645 Phone: (801) 222-8000
646 EMail: debryro@uvsc.edu

647
648 Tom Hastings
649 Xerox Corporation
650 737 Hawaii St. ESAE 231
651 El Segundo, CA 90245
652 Phone: 310-333-6413
653 Fax: 310-333-5514
654 e-mail: hastings@cp10.es.xerox.com

655
656 Robert Herriot
657 Xerox Corp.
658 3400 Hill View Ave, Building 1
659 Palo Alto, CA 94304
660 Phone: 650-813-7696
661 Fax: 650-813-6860
662 e-mail: robert.herriot@pahv.xerox.com

663
664 Kirk Ocke
665 Xerox Corp.
666 800 Phillips Rd
667 M/S 139-05A
668 Webster, NY 14580
669 Phone: (716) 442-4832
670 EMail: kirk.ocke@usa.xerox.com

671
672 Peter Zehler
673 Xerox Corp.
674 800 Phillips Rd
675 M/S 139-05A
676 Webster, NY 14580
677 Phone: (716) 265-8755
678 EMail: peter.zehler@usa.xerox.com

679 14 Appendix A: Full Copyright Statement

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

681 This document and translations of it may be copied and furnished to others, and derivative works that
682 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
683 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
684 this paragraph are included on all such copies and derivative works. However, this document itself may not
685 be modified in any way, such as by removing the copyright notice or references to the Internet Society or
686 other Internet organizations, except as needed for the purpose of developing Internet standards in which
687 case the procedures for copyrights defined in the Internet Standards process must be followed, or as
688 required to translate it into languages other than English.

689 The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its
690 successors or assigns.

691 This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET
692 SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES,
693 EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE
694 OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
695 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

696