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

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   This document specifies an OPTIONAL attribute syntax called
   'collection' for use with the Internet Printing Protocol (IPP/1.0 and
   IPP/1.1). A 'collection' is a container holding one or more named
   values, which are called "member" attributes. A collection allows
   data to be grouped like a PostScript dictionary or a Java Map. This
   document also specifies the conformance requirements for a definition
   document that defines a collection attribute. Finally, this document
   gives some illustrative example collection attribute definitions that
   are not intended as actual attribute specifications.

Table of Contents

Table of Tables

1 Introduction

   This document is an OPTIONAL extension to IPP/1.0 [RFC2565, RFC2566]
   and IPP/1.1 [RFC2911, RFC2910]. For a description of the base IPP
   documents see Appendix D.

1.1 Problem Statement

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

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

1.2 Solution

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

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

   As with any attribute syntax, the document that defines a collection
   attribute specifies whether the attribute is single-valued
   (collection) or multi-valued (1setOf collection). If the attribute
   is multi-valued (1setOf collection), each collection value MUST be a
   separate instance of a single definition of a collection, i.e., it
   MUST have the same member attributes except for OPTIONAL member
   attributes. If we view each collection definition as a separate
   syntax type, this rule continues the IPP/1.1 notion that each

attribute has a single type or pattern (e.g., "keyword | name" is a
pattern). Without this rule, the supported values would be more
difficult to describe and the mechanism defined in item 4 of section
3.1 would not be sufficient.

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

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

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

A collection value MUST NOT contain two or more member attributes
with the same attribute name. Such a collection is mal-formed.
Clients MUST NOT submit such malformed requests and Printers MUST NOT
return such malformed responses. If such a malformed request is
submitted to a Printer, the Printer MUST (depending on
implementation) either (1) reject the request with the 'client-
error-bad-request' status code (see [RFC2911] section 13.1.4.1), or
(2) accept the request and use only one of each duplicate member
attributes.

2 Terminology

   This section defines terminology used throughout this document.

2.1 Conformance Terminology

   Capitalized terms, such as MUST, MUST NOT, REQUIRED, SHOULD, SHOULD
   NOT, MAY, NEED NOT, and OPTIONAL, have special meaning relating to
   conformance as defined in BCP 14, RFC 2119 [RFC2119] and [RFC2911],
   section 12.1. If an implementation supports the extension defined in
   this document, then these terms apply; otherwise, they do not. These
   terms define conformance to this document only; they do not affect
   conformance to other documents, unless explicitly stated otherwise.

2.2 Other terminology

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

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

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

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

3 Definition of a Collection Attribute

   This section describes the requirements for any collection attribute
   definition.

3.1 Information to Include

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

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

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

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

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

   a) "xxx-supported" is a "1setOf collection", which enumerates all
      of the supported collection values of "xxx". If a collection
      of this form contains a nested collection, it MUST be of the
      same form.

      For example, "media-size-supported" might have the values {{x-
      dimension:210, y-dimension:297},{x-dimension:297, y-
      dimension:420}} to show that it supports two values of "media
      size": A4 (210x297) and A3 (297x420). It does not support
      other combinations of "x-dimension" and "y-dimension" member
      attributes, such as 210x420 or 297x297, and it does not support
      non-enumerated values, such as 420x595.

   b) "xxx-supported" is a "1setOf type2 keyword", which enumerates
      the names of all of the member attributes of "xxx": "aaa1",
      "aaa2", ..., "aaaN". If a collection of this form contains a
      nested collection, it MAY be of either form. See item 5f)
      below for details on supported values of member attributes.

      For example, "media-col-supported" might have the keyword
      values: "media-size" and "media-color".

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

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

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

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

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

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

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

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

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

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

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

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

           For example, "media-col-supported" might have the keyword
           values: "media-size" and "media-color". Using the first
           method for defining supported values (an "aaaI-supported"
           attribute), the collection values of "media-col" are
           combinations of values of "media-size-supported" and
           "media-color-supported". If "media-size-supported" has
           the values of '210x297' and '297x420' and "media-color-
           supported" has the values of 'white' and 'pink', the

Printer might support only the combinations 'white-
210x297', 'pink-210x297' and 'white-297x420', and not
'pink-297x420'.

If a collection contains a member "aaaI", whose syntax
type is "text", the supported values would probably be
defined by the definition of "xxx" rather than by the
attribute "aaaI-supported".

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

        i) a fixed default

       ii) a mechanism by which the Printer determines default

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

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

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

     a) a fixed default

     b) a mechanism by which the Printer determines default

     c) an indefinite default that is left to the implementation

     d) a Printer attribute "xxx-default" which is a collection with
        the same member attributes as "xxx". If optional member
        attributes are absent, the Printer uses the defaulting rules of
        item 5g) above.

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

3.2 Nested Collections

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

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

5.1 media-col (collection)

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

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

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

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

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

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

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

5.1.2 media-size (collection)

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

          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

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

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

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

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

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

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

5.2 media-col-default (collection)

The "media-col-default" Printer attribute specifies the media that
the Printer uses, if any, if the client omits the "media-col" and
"media". Job Template attributes in the Job Creation operation and
the PDL do not include a media specification. The member attributes

are defined in Table 1. A Printer MUST support the same member
attributes for this default collection attribute as it supports for
the corresponding "media-col" Job Template attribute.

5.3 media-col-ready (1setOf collection)

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

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

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

6 A Second Example Definition Of A Collection Attribute

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.

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

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

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

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

7 Encoding

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

7.1 Additional tags defined for representing a collection attribute
    value

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

   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

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

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

2. Each member attribute is encoded as a sequence of two or more
   values that appear to be part of a single multi-valued attribute,
   i.e., 1setOf. The first value after the 'begCollection' value has
   the attribute syntax, 'memberAttrName' (0x4A), and its value holds
   the name of the first member attribute (e.g., "aaa"). The second
   value holds the first member's attribute value, which can be of
   any attribute syntax, except 'memberAttrName' or 'endCollection'.
   If the first member's attribute value is multi-valued, the third
   value holds the second value of the first member's value.
   Otherwise, the third value holds the name of second member
   attribute (e.g., "bbb"), and its attribute syntax is
   'memberAttrName'. In this case, the fourth member's value is the
   value of "bbb".

   Note that the technique of encoding a 'collection' as a '1setOf'
   makes it easy for a Printer that doesn't support a particular
   collection attribute (or the collection attribute syntax at all)
   to simply skip over the entire collection value.

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

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

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

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

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

Additional examples have been included in the appendices.

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

```
   "media-col" =
     { "media-color" =  'blue';
        "media-size" =
        {    "x-dimension" = 6;
             "y-dimension" = 4
          }
      },
```

The full encoding is in table 5. A simplified view of the encoding
looks like this:

          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	""	""

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
			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)

Octets	Symbolic Value	Protocol field	comments
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"

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 "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
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

Octets	Symbolic Value	Protocol field	comments
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

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

9 IANA Considerations

The following table provides registration for the 'collection'
attribute syntax defined in this document. This is to be registered
according to the procedures in RFC 2911 [RFC2911] section 6.3.

Tag value:	Attribute Syntaxes:	Ref.	Section:
collection		RFC 3382	3
0x34	begCollection	RFC 3382	7.1
0x37	endCollection	RFC 3382	7.1
0x4A	memberAttrName	RFC 3382	7.1

The resulting attribute syntax registration will be published in the
http://www.iana.org/assignments/ipp-registrations registry.

10 Internationalization Considerations

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

11 Security Considerations

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

12 References

12.1 Normative References

   [RFC2910] Herriot, R., Butler, S., Moore, P. and R. Turner, "Internet
             Printing Protocol/1.1: Encoding and Transport", RFC 2910,
             September 2000.

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

12.2 Informative References

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

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

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

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

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

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

   [RFC3196] Hastings, T., Manros, C., Zehler, P., Kugler, C. and H.
             Holst, "Internet Printing Protocol/1.1: Implementor's
             Guide", RFC 3196, November 2001.

Appendix A: Encoding Example of a Simple Collection (Informative)

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

      "media-size" =
        { "x-dimension" = 6;
           "y-dimension" = 4
        }

   A simplified view of the encoding would look like this:

           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   ""   ""

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

           Table 7 - Example Encoding of simple collection

      Octets   Symbolic Value Protocol   comments
                                      field

      0x34   begCollection   value-tag   beginning of the "media-
                                                   size" collection attribute

      0x000A   name-   length of (collection)
                                      length   attribute name

      media-size   media-size   name   name of (collection)
                                                   attribute

Octets	Symbolic Value	Protocol field	comments
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 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)

Octets	Symbolic Value	Protocol field	comments
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)

Appendix B: Encoding Example of 1setOf Collection (Informative)

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

```
      "media-size-supported" =
        {  "x-dimension" = 6;
           "y-dimension" = 4
        },
        {  "x-dimension" = 3;
           "y-dimension" = 5
        };
```

   A simplified view of the encoding would look like this:

          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	""	""

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)

Octets	Symbolic Value	Protocol field	comments
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"
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)

Octets	Symbolic Value	Protocol field	comments
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

Octets	Symbolic Value	Protocol field	comments
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
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

Octets	Symbolic Value	Protocol field	comments
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)

Appendix C: Encoding Example of Collection containing 1setOf XXX
           attribute (Informative)

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

      "wagons" =
        { "colors" = red, blue;
           "sizes" = 4, 6, 8
        }

   A simplified view of the encoding would look like this:

      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	""	""

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	colors	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	

Octets	Symbolic Value	Protocol field	comments
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

Octets	Symbolic Value	Protocol field	comments
0x0004		value	1st 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
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

```
        Octets         Symbolic Value  Protocol    comments
                                        field

                                                    no value (since value-
                                                    length was 0)
```

Appendix D: Description of the Base IPP Documents (Informative)

    The base set of IPP documents includes:

        Design Goals for an Internet Printing Protocol [RFC2567]
        Rationale for the Structure and Model and Protocol for the
        Internet Printing Protocol [RFC2568]
        Internet Printing Protocol/1.1: Model and Semantics [RFC2911]
        Internet Printing Protocol/1.1: Encoding and Transport [RFC2910]
        Internet Printing Protocol/1.1: Implementer's Guide [RFC3196]
        Mapping between LPD and IPP Protocols [RFC2569]

    The "Design Goals for an Internet Printing Protocol" document takes a
    broad look at distributed printing functionality, and it enumerates
    real-life scenarios that help to clarify the features that need to be
    included in a printing protocol for the Internet. It identifies
    requirements for three types of users: end users, operators, and
    administrators. It calls out a subset of end user requirements that
    are satisfied in IPP/1.0 [RFC2566, RFC2565]. A few OPTIONAL operator
    operations have been added to IPP/1.1 [RFC2911, RFC2910].

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

    The "Internet Printing Protocol/1.1: Model and Semantics" document
    describes a simplified model with abstract objects, their attributes,
    and their operations. The model introduces a Printer and a Job. The
    Job supports multiple documents per Job. The model document also
    addresses how security, internationalization, and directory issues
    are addressed.

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

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

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

Authors' Addresses

    Roger deBry
    Utah Valley State College
    Orem, UT 84058

    Phone: (801) 222-8000
    EMail: debryro@uvsc.edu

    Tom Hastings
    Xerox Corporation
    737 Hawaii St. ESAE 231
    El Segundo, CA 90245

    Phone: 310-333-6413
    Fax: 310-333-5514
    EMail: hastings@cp10.es.xerox.com

    Robert Herriot
    Consultant
    706 Colorado Ave
    Palo Alto, CA 94303

    Phone: 650-327-4466
    Fax: 650-327-4466
    EMail: bob@herriot.com

Kirk Ocke
Xerox Corp.
800 Phillips Rd
M/S 128-30E
Webster, NY 14580

Phone: (585) 442-4832
EMail: KOcke@crt.xerox.com

Peter Zehler
Xerox Corp.
800 Phillips Rd
M/S 128-30E
Webster, NY 14580

Phone: (585) 265-8755
EMail: PZehler@crt.xerox.com

IPP Web Page: http://www.pwg.org/ipp/
IPP Mailing List: ipp@pwg.org

To subscribe to the ipp mailing list, send the following email:

  1) send it to majordomo@pwg.org
  2) leave the subject line blank
  3) put the following two lines in the message body:
       subscribe ipp
       end

Implementers of this specification document are encouraged to join
the IPP Mailing List in order to participate in any discussions of
clarification issues and review of registration proposals for
additional attributes and values. In order to reduce spam the
mailing list rejects mail from non-subscribers, so you must subscribe
to the mailing list in order to send a question or comment to the
mailing list.

Full Copyright Statement

   Copyright (C) The Internet Society (2002). All Rights Reserved.

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

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

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

Acknowledgement

   Funding for the RFC Editor function is currently provided by the
   Internet Society.