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

Abstract

This document specifies an OPTIONAL attribute syntax called 'collection' for use with the Internet Printing Protocol/1.0 (IPP) [RFC2565, RFC2566], IPP/1.1 [ipp-mod, ipp-pro], and subsequent versions. 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.

The full 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 (this document)
Internet Printing Protocol/1.1: Encoding and Transport [IPP-PRO]
Internet Printing Protocol/1.1: Implementer's Guide [IPP-IIG]
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. A few OPTIONAL operator operations have been added to IPP/1.1.

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 working group's major decisions.

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 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 a new scheme named 'ipp' 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.

Table of Contents

1 Problem Statement

The IPP Model and Semantics [ipp-mod] 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 must allow for optional members and subsequent extension of the collection.

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

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 ~~IPP~~ 'collection' attribute syntax is a container holding one or more named values (i.e., attributes), which are called member attributes. Each collection attribute ~~is named and its specification~~ 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 collection attribute definition document specifies whether the attribute is single-value (collection) or multi-valued (1setOf collection).

The name of each member attribute MUST be unique, but MAY be the same as the name of a member attribute in another collection type 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 ~~a collection~~, 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 contains 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 reject the request with the 'client-error-bad-request' status code (see section 13.1.4.1)

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

3 Definition of a Collection Attribute ~~Type~~

This section describes the requirements for any collection attribute definition.

3.1 Member Attribute Naming Rules

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

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

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

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

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

if any of the "media-supported" (1setOf (type3 keyword | name)) Printer attribute values could be specified for job sheets. The reason that the "job-sheet-format" member attribute isn't named simply, "job-sheet", is because its values only indicate the format, and don't imply any media, while the "job-sheets" (type2 keyword | name) Job Template attribute do imply a media. This example illustrates when a member attribute can be the same as another attribute (in this case a Job Template attribute) and when the member attribute MUST have a different name.

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

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

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

3.2 Remaining rules for a ~~Specification of a~~ collection attribute definition

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

1. The name of the collection attribute MUST be specified. (e.g. "xxx")

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. The member attributes MUST be defined. For each member attribute the definition document MUST provide the following ~~MUST be provided~~:~~.~~

   a) The member attribute's name, "aaa", MUST either (1) reuse the attribute name of another attribute if the member attribute shares the syntax and semantics with the other attribute or (2) 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 is the same as another attribute (case of option a-1 above), it MUST have the same attribute syntax, including cardinality (1setOf or not)~~This MAY be expressed with a reference to the associated attribute in the case of option a-1 above~~.

e) The semantics of the "aaa" member attribute. The semantic definition ~~should~~ MUST include a description of any constraint or boundary conditions the member attribute places on the associated attribute, especially if the attribute is the same as another attribute used in a different context (case of option a-1 above)

f) ~~its~~ the supported values for the "aaa" member attribute, either enumerated explicitly or specified by the values of a referenced attribute which may be specified by either:

– the attribute’s definition

– a Printer~~n~~ attribute, such as "~~yyy~~aaa-supported", which contains the explicit values supported. The "~~yyy~~aaa-supported" attribute is a Printer attribute and not in a collection. For example, if a collection contains the "media" attribute and its supported values are specified by the "media-supported" attribute, the "media-supported" attribute is the same Printer attribute that the "media" attribute uses.

g) the default value of "~~yyy~~aaa" member attribute if it is OPTIONAL for a client to supply the "~~yyy~~aaa" member attribute in a request. The default value is specified by either:

– the attribute’s definition

– a Printer~~n~~ attribute, such as "~~yyy~~aaa-default", which may have a collection value

– or an implementation defined algorithm that takes into account the values of the other member attributes of the collection value

h) ~~For any member attribute of a job template collection the syntax of "aaa-supported" MUST be specified. See section a) below.~~Depending on the collection attributes context, it MUST follow the additional rules specified below for the various contexts.

3.3 Nested Collections

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

1. The name of the collection attribute, e.g., "yyy"

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

3. The member attributes MUST be defined. For each member attribute the definition document MUST provide the following:~~For each member attribute the following MUST be provided~~

a) The member attribute's name, "bbb", MUST either (1) reuse the attribute name of another attribute if the member attribute shares the syntax and semantics with the other attribute or (2) 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 is the same as another attribute (case of option a-1 above), it MUST have the same attribute syntax, including cardinality (1setOf or not)~~This MAY be expressed with a reference to the associated attribute in the case of option a-1 above.~~

e) The semantics of the member attribute. The semantic definition ~~should~~ MUST include a description of any constraint or boundary conditions the member attribute places on the associated attribute, especially if the attribute is the same as another attribute used in a different context (case of option a-1 above)

f)

g) Depending on the collection attributes context, it MUST follow the additional rules specified below for the various contexts.~~For any member attribute of a job template collection the syntax of "bbb-supported" MUST be specified. See section a) below.~~

3.4 Collection Attributes as Operation Attributes

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

a) Define in which operation requests the collection attribute is intended to be used.

b) Define in which operation responses the collection attribute is intended to be used.

3.5 Collections as Job Template Attributes

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

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

For the following rules assume the "xxx" (collection) example from section 3.2 is a job template attribute.

1) There ~~are two~~MUST be two associated printer attributes. The attributes are "xxx-supported" and "xxx-default"

2) The "xxx-default" is a collection with a syntax identical to the "xxx" specification in section 3.2 .

   - Each member attribute has the same name as in the "xxx" definition.

   - A Get-Printer-Attributes operation MUST return the "xxx-default" (collection) Printer attribute and all the member attributes. Any default values that have been set MUST be returned. Any default values that have not been set MUST return an out of band attribute of 'no-value'.

3. If the definition of the collection does not mention an "xxx-ready" attribute than it is assumed that one is not defined, though implementer's are free to support an "xxx-ready" as an extension.

4. The collection attribute definition document MUST define an "xxx-supported" ~~an~~ attribute with either a syntax of '1setOf type2 keyword' or '1setOf collection':

   - If the definition uses the '1setOf type2 keyword' attribute syntax, it MUST be the attribute keyword names of all of the member attributes that the Printer implementation supports in a Job Creation operation. Furthermore, the definition MUST include corresponding definitions of each of the "aaa-supported" attributes that correspond to each "aaa" member attribute. Then a client can determine the supported values of each member attribute in the Job Template collection attribute

   - If the definition uses the '1setOf collection' attribute syntax, then the values are the supported instances of the "xxx" (collection) attribute that a client can supply in a Job Creation operation. It is expected that this second approach will be used for small collections whether the number of possible collection values is small. For example, a "media-size" (collection) member attribute in which the member attributes are "x-dimension" (integer) and "y-dimension" (integer). The pairs of integers are just like keywords as far as the client localization is concerned, except that if the client doesn't recognize a size pair of numbers, it can display the numbers.

~~ISSUE 03 - For certain small collections where all member attributes MUST be supplied and supported, such as "media.size" (collection) where the collection is "media.size.x" and "media.size.y", it would be useful to allow the "xxx-supported" (1setOf collection) to show the possible combinations of x and y dimensions. Thus this rule should be amended to allow either form in a definition. The pairs of integers are just like keywords as far as the client localization is concerned, except that if the client doesn't recognize a size pair of numbers, it can display the numbers.~~

   a) The keywords returned lists all the contained member attribute names. This example would return the "aaa" keyword.

b) The list is recursive and lists all the member attributes of the contained collections. In section 3.3 the printer would return "aaa" and "bbb" for collection "xxx"

c) The encoding convention allows the reconstruction of the collection structure. The will allow the client to reconstruct the collections. The client would know that "aaa" is a member of collection "xxx". It can also be derived that collection "bbb" is a member of collection "yyy". See section 7 for more information on encoding.

d) To obtain the supported values for any member attribute a client performs a Get-Printer-Attributes operation explicitly requesting the member attribute name with the suffix "supported". If a member attribute is itself a collection rule 4 above applies to member attribute.

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

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

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

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

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

4 New Out-of-band value

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.

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

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

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

- the definition of the attribute specifies 'none' MUST be supported AND

- the definition of the attribute specifies 'none' MAY be supported and it is a value of the attribute "xxx-supported".

5 Unsupported Values

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

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 [ipp-mod] section 4.1) in the Unsupported Attributes Group.

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 [ipp-mod] section 4.1). The unsupported member attributes and conflicting values have their unsupported or conflicting values.

6 Sample specification

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

Name: media-–col

Syntax: collection

Member Attributes:

Name: "media-color"

Syntax: type3 keyword | name

Mandatory

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

"media-color-supported" syntax: 1setO~~o~~f (type2 keyword | name)

Name: "media-size"

Syntax: collection

Member Attributes:

Name: "x-dimension"

Syntax: integer

Mandatory

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

x-supported syntax: rangeOfInteger

Name: "y-dimension"

Syntax: integer

Mandatory

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

y-supported syntax: rangeOfInteger

Name: name

Syntax: See job template attribute "media"

Optional

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

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

7 Encoding

This section is still under construction.

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

a) must be simple

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

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

The two encodings are:

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

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

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

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

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

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

a) a one byte tag

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

c) "n" bytes of a name

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

e) "v" bytes of a value

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

There are 3 encoding examples for each solution:

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

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

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

Solution 1a)

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	""

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

"media-size-supported"	end-collection	""
"media-size-supported"	begin-collection	""
"media-size-supported.x"	integer	850
"media-size-supported.y"	integer	1400
"media-size-supported"	end-collection	""

Name	syntax-type	value
"job-notify"	begin-collection	""
"job-notify.notify-recipients"	url	"mailto://bill@foo.com"
"job-notify.notify-events"	keyword	job-completed
""	keyword	job-created
""	keyword	job-state-changed
"job-notify"	end-collection	""

Solution 1b)

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

Name	syntax-type	value
"media-size-supported"	begin-collection	""
"x-"	integer	850
"y-"	integer	1100
""	end-collection	""
""	begin-collection	""
"x-"	integer	850
"y-"	integer	1400
""	end-collection	""

Name	syntax-type	value
"job-notify"	begin-collection	""
"notify-recipients-"	url	"mailto://bill@foo.com"
"notify-events-"	keyword	"job-completed"
""	keyword	"job-created"
""	keyword	"job-state-changed"
"job-notify"	end-collection	""

Solution 2)

Name	syntax-type	value

"media-col"	begin-collection	""
""	attribute-name	"media-color"
""	keyword	white
""	attribute-name	"media-size"
""	begin-collection	""
""	attribute-name	"x"
""	integer	850
""	attribute-name	"y"
""	integer	1100
""	end-collection	""
""	end-collection	""

Name	syntax-type	value
"media-size-supported"	begin-collection	""
""	attribute-name	"x"
""	integer	850
""	attribute-name	"y"
""	integer	1100
""	end-collection	""
""	begin-collection	""
""	attribute-name	"x"
""	integer	850
""	attribute-name	"y"
""	integer	1400
""	end-collection	""

Name	syntax-type	value
"job-notify"	begin-collection	""
""	attribute-name	"notify-recipients"
""	url	mailto://bill@foo.com"
""	attribute-name	"notify-events"
""	keyword	"job-completed"
""	keyword	"job-created"
""	keyword	"job-state-changed"
""	end-collection	""

Observations:

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

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

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

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

7.1 encoding of a collection (using solution 1a)

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

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

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

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

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

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

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

4. A null collection value length (e.g. 0x00)

5. The attributes are encoded as with any other attribute. It is valid to have a collection a member of a collection. The modifications necessary for encoding member attributes of a collection are as follows.

    a) The name of the member attribute MUST be prepended with the collection name and a period.

    b) The length of the member attribute name MUST be adjusted appropriately.

6. The end of the collection is indicated with a value tag that MUST be syntax type 'endCollection' (e.g. 0x37).

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

8. The collection name (e.g. "xxx")

9. A null collection value length (e.g. 0x00)

~~7.1~~7.2 Sample Encoding (using solution 1a)

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

This section defines the encoding of a collection syntax type using solution 1a. The collection specified in section 6 is used. The encoding is of an implementation that does not support any optional attributes. A 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.

A collection value is encoded as a sequence of attribute values preceded by a beginCollection attribute and followed by an endCollection attribute. The name field of a beginCollection and an endCollection both contain the name of the collection type, i.e., the keyword name of the collection attribute, which is a string of ASCII characters. The value field contains the prefix used for all subordinate member attributes. The following example is written in the style of the IPP/1.1 "Encoding and Transport" document [ipp-pro]. The following example is for a media collection attribute. The media collection contains 2 member attributes. One member is "color" that contains a keyword for the media's color. The second attribute is a collection that gives the media's size. The size collection has two integer attributes "x" and "y" that gives the media's 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	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
0x0010		name-length	Length of member attribute name
media-col.size.x	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	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	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	media-col	Name	Sub-collection's name
0x0000		Value-length	

~~7.2~~7.3 1setOf Collection encoding (using solution 1a)

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

~~7.3~~7.4 Sample 1setOf Collection encoding (using solution 1a)

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

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

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		name-length	Length of member attribute name
media-col.size.x	media-col.size.x	Name	Name of member attribute
0x0004		value-length	
0x0004		Value	
0x37	endCollection	value-tag	end 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	
			Second collection in set
0x34	beginCollection	value-tag	Beginning of the collection
0x0000		name-length	Indicates continuation of set
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
0x0003		value-length	
red	red	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
0x0010		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
0x0010		name-length	Length of member attribute name
media-col.size.x	media-col.size.x	Name	Name of member attribute
0x0004		value-length	
0x0004		Value	
0x37	endCollection	value-tag	end of the sub-collection

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

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 will look like ordinary attributes, but since they each are encoded with a unique name that can't be the same as a top level attribute, each of the member attributes will also look like attributes that the recipient doesn't support and so should ignore.

9 IANA Considerations

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

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

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 [ipp-mod].

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 [ipp-mod].

12 References

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

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

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

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

[RFC2566]
R. deBry, T. Hastings, R. Herriot, S. Isaacson, 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., Martin, J., "Mapping between LPD and IPP Protocols", RFC 2569, April 1999.

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

13 Author's 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
e-mail: hastings@cp10.es.xerox.com

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

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

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

14 Appendix A: Full Copyright Statement