

1 Proposed Internet-Draft

S. Isaacson
Novell, Inc.
D. Taylor
Novell, Inc.
M. MacKay
Novell, Inc.
P. Zehler
Xerox Corporation
T. Hastings
Xerox Corporation
C. Manros
Xerox Corporation
November 1996

17 LDPA - Lightweight Document Printing Application
18 Version 0.8, October 31, 1996

20 Status of this Memo

21
22 This document is a working version of a protocol specification. It
23 will eventually become an Internet-Draft by following well defined
24 IETF procedures. At that time, the following paragraphs must be
25 included:

26
27 This document is an Internet-Draft. Internet-Drafts are working
28 documents of the Internet Engineering Task Force (IETF), its areas,
29 and its working groups. Note that other groups may also distribute
30 working documents as Internet-Drafts.

31
32 Internet-Drafts are draft documents valid for a maximum of six months
33 and may be updated, replaced, or obsoleted by other documents at any
34 time. It is inappropriate to use Internet-Drafts as reference
35 material or to cite them other than as ``work in progress.''

36
37 To learn the current status of any Internet-Draft, please check the
38 ``lid-abstracts.txt' listing contained in the Internet-Drafts Shadow
39 Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe),
40 munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or
41 ftp.isi.edu (US West Coast).

43 Abstract

44
45 This Internet-Draft specifies a Lightweight Document Printing
46 Application (LDPA) protocol for the Internet. This protocol is a
47 subset of the semantic operations and attributes defined in ISO/IEC
48 10175 Document Printing Application (DPA) parts 1 and 3. It also
49 incorporates some of the implementation and interoperability lessons
50 learned from other printing related standards such as POSIX System
51 Administration - Part 4 (POSIX 1378.4) and X/Open A Printing System
52 Interoperability Specification (PSIS).

53
54 LDPA is defined as a set of abstract data types and operations. The
55 operations are implemented using the Internet standard remote
56 procedure call mechanisms defined in RFC 1831 (RPC: Remote Procedure
57 Call Specification Version 2).

58
59 The LDPA protocol covers only user operations on basic print service

60 objects. Authentication, Access Control, Device Management, and
61 Service Management are all outside the scope of this protocol. These
62 areas are covered by other protocols include methods and operations
63 for service creation, management, and administration. The SNMP
64 Printer MIB [1] is an example of one of these. In the areas where
65 there are no existing standards, many are being worked in other
66 distributed service forums (management, security, etc.). As these
67 services become more standardized, this document (and hence the
68 protocol) can be updated to reflect the integration and relationships
69 with these other standards.
70

Table of Contents

1. Introduction	5
2. Distributed Printing	6
2.1 Components	6
2.2 Objects	6
2.2.1 Printer	7
2.2.2 Job	9
2.2.3 Document	9
2.2.4 Initial Value Job	9
2.2.5 Initial Value Document	9
2.3 Object Relationships	9
2.4 Use of Naming and Directory Services	9
2.4.1 Status	10
2.4.2 Resolution	11
2.4.4 Color Supported	11
2.4.5 Maximum Speed	11
2.4.6 Maximum Speed Units	11
2.4.7 Plug and Play Device Id	11
2.4.8 Model	11
2.4.9 Manufacturer	11
2.4.10 Type	12
2.4.11 PDLs Supported	12
2.4.12 Sides Supported	12
2.5 OIDs	12
3. Internet Printing Model	12
3.1 Object Instances	12
3.2 Limits and Defaults	13
3.3 List Object Attribute Scoping Rules	13
4. Operations	13
4.1 Common Data Structures	13
4.1.1 XDR	13
4.1.2 ASN.1	16
4.2 Errors	17
4.2.1 ASN.1	17
5. Binding and Unbinding	19
5.1 Bind Operation	19
5.1.1 Bind Argument	19
5.1.2 Bind Result	21
5.2 Unbind Operation	22
5.2.1 Unbind Argument	22
5.2.2 Unbind Result	22
6. User Operations	22
6.1 Print Operation	22
6.1.1 Print Argument	22
6.1.2 Print Result	24
6.2 Cancel Job Operation	25
6.2.1 Cancel Job Argument	25
6.2.2 Cancel Job Result	26
6.3 List Object Attributes Operation	26
6.3.1 List Object Attributes Argument	27
6.3.2 List Object Attributes Result	30

128	6.4 Modify Job Operation	31
129	6.4.1 Modify Job Argument	31
130	6.4.2 Modify Job Result	32
131	6.5 Resubmit Job Operation	33
132	6.5.1 Resubmit Job Argument	33
133	6.5.2 Resubmit Job Result	34
134		
135	7. Object Attributes	35
136	7.1 Job Attributes	36
137	7.1.1 Job Informational Attributes	37
138	7.1.2 Printer Selection Attributes	39
139	7.1.3 Job Status Attributes	39
140	7.1.4 Job Results Handling Attributes	43
141	7.1.5 Job Event Handling Attributes	44
142	7.1.6 Job Scheduling Instructions Attributes	45
143	7.2 Document Attributes	47
144	7.2.1 Document Description Attributes	47
145	7.2.2 Document Production Instruction Attributes	48
146	7.2.3 Document Characteristics Attributes	52
147	7.2.4 Document Status Attributes	54
148	7.3 Operation Attributes	54
149	7.4 Printer Attributes	54
150	7.4.1 printer-name	56
151	7.4.2 printer-state	56
152	7.4.3 message	56
153	7.4.4 printer-initial-value-job	56
154	7.4.5 printer-initial-value-document	57
155	7.4.6 fonts-supported	58
156	7.4.7 fonts-ready	58
157	7.4.8 media-supported	58
158	7.4.9 media-ready	58
159	7.4.10 printer-associated-printers	58
160	7.4.11 document-formats-supported	58
161	7.4.12 numbers-up-supported	59
162	7.4.13 finishings-supported	59
163	7.4.14 sides-supported	59
164	7.4.15 job-sheets-supported	59
165	7.4.16 document-sheets-supported	60
166	7.4.17 maximum-copies-supported	60
167	7.4.18 notification-delivery-methods-supported	61
168	7.4.19 physical-printers-supported	61
169	7.4.20 Logical-printers-supported	61
170	7.4.21 events-supported	61
171	7.4.22 transfer-methods-supported	61
172	7.4.23 locales-supported	61
173	7.4.24 multiple-documents-supported	61
174	7.4.28 cancel-individual-document-supported	62
175	7.4.29 modify-individual-document-supported	62
176	7.5 Initial Value Job Attributes	62
177	7.6 Initial Value Document Attributes	62
178	7.7 Relationship to ISO/IEC 10175 Conformance Levels	62
179		
180	8. Security Considerations	63
181		
182	9. References	63
183		
184	10. Author's Address	63

185 1. Introduction

186
187 This document is a Proposed Internet-Draft. It is a specification for
188 a protocol that can be used for distributed printing on the Internet.
189 This protocol, Lightweight Document Printing Application (LDPA), is
190 based on the printing model introduced in the Document Printing
191 Application (ISO/IEC 10175 DPA) standard. The DPA model describes a
192 distributed printing service made up of cooperating networked
193 entities. DPA also identifies user and administrative roles and
194 operations. These ideas and concepts, when unified with other
195 Internet protocols and services, realizes a distributed print service
196 for the Internet.

197
198 This specification obsoletes RFC 1179 "Line Printer Daemon Protocol"
199 [10]. "lpr" was designed a long time ago with line printers in mind.
200 It does not fit with current page oriented printing technologies and
201 most printer vendors have made their own proprietary extensions to
202 "lpr" to try and get by with current needs. Unfortunately, these
203 extensions are mutually incompatible, which means that many enhanced
204 "lpr" implementations cannot interwork. In the X/Open PSIS project
205 [6], these differences were documented and appropriate gateway
206 solutions between DPA and each vendor (Digital, HP, IBM, SCO, Sun,
207 and Xerox) described. LDPA introduces several improvements over RFC
208 1179:

- 209 - It uses Object Identifiers (OIDs) for interoperability and
210 extensibility.
- 211 - It models the more complex, multiple content object page oriented
212 languages and printers.
- 213 - It is based on a well known and well tested ISO standard.
- 214 - It supports desktop printing models as well as current market
215 trends for distributed systems which are as diverse as they are
216 spread throughout the globe.

217
218 This document assumes a distributed computing environment where print
219 service users (clients, applications, drivers, etc.) cooperate and
220 interact with print service providers (servers, printers, gateways,
221 etc.) to realize the print service.

222
223 The actual protocol consists of abstract data types representing the
224 distributed print service and its components as well as operations
225 which define and give semantics to the interaction between these
226 components. The operations defined for this protocol are defined as
227 RFC 1831 [2] compliant remote procedure calls. These operations are
228 defined in sections 4, 5, and 6. The objects and their attributes
229 are defined in section 7.

230
231 NOTE: The abstract data types could be defined using a syntax such as
232 RPC language [2] or ASN.1 [8]. The actual encoding of the abstract
233 data types could be realized using either XDR [3] or BER [9]. The
234 actual syntax and encoding mechanism must be finalized during the
235 standards process. Only the operations and attribute semantics are
236 defined at this time. This document does not yet contain the small
237 subset of syntax definitions for the proposed attributes. These can
238 be added as a short appendix to this document. For the operations,
239 this document currently shows operations defined in both ASN.1 and
240 XDR. The full ASN.1 for ISO/IEC 10175 parts 1 and 3 (attributes,
241 syntaxes, and operations) can be found at

"ftp://ftp.pwg.org/pub/pwg/snmpmib/dpa". A proposal for the XDR version of the operation, attributes, and syntaxes can be found at "ftp://ftp.pwg.org/pub/pwg/netprint/ldpa".

2. Distributed Printing

The distributed printing service is defined as a collection of coordinating and cooperating entities in a distributed computing environment. The model assumed by this protocol is a n-tier client/server model. A service requester (client) makes service requests of service providers (servers). A given instance of a service provider (server) may in turn be a service requester (client) of some other service via its service providers (servers).

A client is able to access the services offered by a server by invoking one or more operations associated with the server. Each operation has associated arguments and results. The arguments provide additional data which is passed from the client to the server. The results return the status and outcome of the desired operation back to the client from the server.

2.1 Components

In the distributed printing service the entities or components are:

- One or more humans or agents acting on behalf of humans. Humans (or their agents) act in the role of Users, Operators, Managers, or Administrators.
- One or more clients. Client are computer network nodes with which end users interact in order to manipulate the distributed print service. A client implements the LDPA protocol.
- One or more print service providers. An instance of a print service provider implements the LDPA protocol and performs and responds to LDPA operations. A given instance of a print service provider can be a "client" of yet another instance of a print service provider. A print service provider can either be physical or logical (physical if it represents a physical printer or some other document production device, or logical if it represents one or more print service providers each of which may be logical or physical).

This LDPA specification only defines the operation used by Users. The operations used by Operators, Managers, and Administrators are not within the scope of this standard.

2.2 Objects

To accomplish the action(s) requested via an operation, the print service provider manages and manipulates data objects. These are simply convenient collections of data that may represent other objects (real life or computer system) elsewhere. A client supplies arguments in the form of attribute values for some of these objects. A server informs the client of the status or outcome of an operation by also providing attribute values for the objects involved in the operations. These objects are not encapsulations of both data and

behavior as in other object oriented models, but are simple collections of attribute/value pairs.

The objects which are relevant to this protocol are:

- Printer
- Job
- Document
- Initial Value Job
- Initial Value Document

LDPA defines the operations that interact with and affect the real-life objects represented by the protocol's object definition.

2.2.1 Printer

One of the most significant components within the distributed printing service is a Printer. A Printer is an instance of a print service provider that provides access to both Logical and Physical output devices. A Printer object is a composition of some of the functionality that has traditionally been tied to other components within the printing system. A Printer can support the functionality of spooling, job management, device management, server, as well as more traditional device components.

A Printer can be in one of two modes:

Public Access: The Printer is not restricted with any access control checks. The Printer uses a simple, name only (no password or credential) form of binding.

Controlled Access: The Printer may have some restrictions based on some authentication and authorization scheme. The Printer uses some form of credential based binding.

A Printer object represents an instance of a print service provider which implements the LDPA protocol. This allows the Printer to provide a common interface for all types of disparate and diverse physical devices or as well as a gateway interface for other non-Internet based printing systems.

To a print service user, a Printer has the "looks and feel" of a any typical physical printer. Jobs are submitted to and managed at the Printer. The Printer can accept or reject submitted jobs based on job attributes which are sent along with the print job. The Printer tracks all jobs that have been submitted to it. The Printer can be modified to indicated a corresponding behavior change at the device level (either manually or automatically). In the Controlled Access mode, the Printer has an identity with a security or credential service.

The Printer can be a service provider for any of the following configurations:

- The Printer can be a Physical Printer which represents and controls a print device using a device specific interface (possibly embedded).

- The Printer can be a Gateway to some other printing system (for example LPD).
- The Printer can be a Logical Printer which feeds other Printers (either Physical or Logical or Gateway).

Figure 1 shows the some of the typical configurations of Printers:

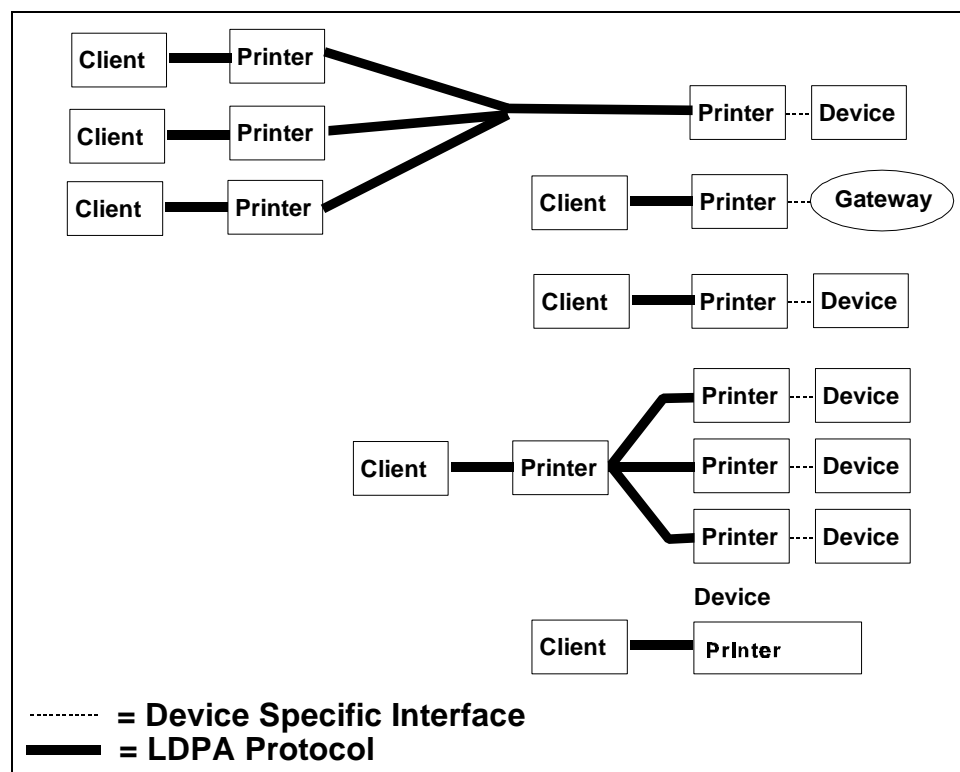


Figure 1 Printer Configurations

The following questions may arise:

1. How can I submit a job to a queue and have that job be routed to any available physical printer?

In the LDPA model, there are two kinds of Printers. One that directly drives a print device by using a device specific protocol. We call that Printer a Physical Printer. The second kind of Printer is one that communicates with other Printers using the LDPA protocol. The Physical Printer only implements the server side of the LDPA protocol. The Logical Printer implements both the client and the server side of the LDPA protocol.

2. How can I achieve both "fan out" and "fan in" of client to Printers and from Printers to physical print devices?

For "fan in", an administrator may set up one or more non-spooling Logical Printers that feed a given Printer. These Logical Printers are used to define different defaults, capabilities, and access

rights. For "fan out", an administrator may set up a Logical Printer which can fan out to one or more downstream Printers for load balancing and reliability.

3. How can a Printer be configured to service multiple spooling Logical Printers?

The system administrator establishes which Logical and Physical printers spool and which do not. If multiple spooling Logical Printers are used to feed the same Printer, their schedulers are not coordinated. Although this configuration is possible, it is not recommended.

2.2.2 Job

A Job object is used to model a job. A job can consist of one or more documents. There are certain job attributes that pertain to the running and scheduling of the entire job (all documents). There are other job attributes that define global behavior or defaults for all contained jobs.

2.2.3 Document

A Document object is used to model a document. There are attributes that describe the contents of the document as well as the processing and handling of the document.

2.2.4 Initial Value Job

An Initial Value Job object is used to model job defaults. These are essentially job attributes that are used as default attributes for each job that is submitted.

2.2.5 Initial Value Document

An Initial Value Document is used to model document defaults. These are essentially document attributes that are used as default attributes for each document that is submitted as part of a job.

2.3 Object Relationships

Instances of objects within the system have relationships which must be maintained persistently along with the persistent storage of the objects themselves.

An instance of a print service provider is a Printer. The Printer is represented via a Printer object. A Printer can contain zero, one, or more Job objects. A Job object contains one or more Document objects. The following relationships are examples:

- "Document object D1 belongs to Job object J1", or
- "Job object J1 belongs to Printer object P1".

2.4 Use of Naming and Directory Services

Any distributed service uses some sort of naming and/or directory service for (X.500, NDS, DCE naming, DNS). It is outside the scope of this protocol to define which name service to use or what the protocol is for using that name service, but the following discussion helps to clarify how the name service is intended to be used.

To distributed printing system, the instances of print service providers are represented by objects of type Printer. These same instances are also registered to the name/directory service. There is one entry in the name service for each Printer.

That is, instances of print service providers are represented to LDBA as Printer objects. These objects represent their real-life counterparts, the print service provider (software, hardware, or firmware). However, for directory lookup, there is an entry in the naming service that also represents the Printer.

It is important to remember that a Printer object represents the current status and configuration information of a certain print service provider. The Printer object contains attributes and values that describe the characteristics and capabilities of the logical or physical print device. However, a few of the most important attributes from the Printer object are duplicated in the entry in the directory. These attributes are used for filtered directory lookups. The results of these searches enable a user to select an appropriate printer. It is the responsibility of the Printer itself to keep these attributes consistent and accurate. This requirement frees the directory or some directory agent from continually polling registered entities for configuration changes.

The following attributes are in the directory entry:

- Fully Distinguished Name (the name within the directory's name space)
- Description
- Location
- Owner
- Address
- Status
- Resolution
- Color Supported
- Maximum Speed
- Maximum Speed Units
- Device Id
- Model
- Manufacturer
- Type
- PDLs Supported
- Sides Supported

The final set of name service entry attributes needs to be finalized and rationalized with the PSIS name service recommendations [6] as well as implementation experience.

2.4.1 Status

The printer status field in the directory entry is really a "summary" attribute of the true printer state. The following mapping takes place between the Printer Status attribute in the directory entry and the printer-state attribute in the Printer object:

```
"Not Connected"
    STATE_NOT_CONNECTED
    STATE_PAUSED_NOT_CONNECTED
"Shutdown"
    STATE_SHUTDOWN
"Active"
    STATE_IDLE
    STATE_PAUSED
    STATE_PRINTING
"Stopped"
    STATE_STOPPED
    STATE_PAUSED_STOPPED
```

Even though the Printer may not be up and running, the directory entry still exists in the directory. In this case, the directory entry represents the fact that it may begin running at some future time.

2.4.2 Resolution

This is a single valued, maximum resolution in either the horizontal or vertical direction of the print device in dpi.

2.4.4 Color Supported

This is a BOOLEAN for either yes, color printing is supported, or no color printing is not supported.

2.4.5 Maximum Speed

This is the maximum speed of the printer in the units defined in Maximum Speed Units

2.4.6 Maximum Speed Units

This is the units of the maximum speed rating of the print device. This can be: pages per minute, sheets per minutes, characters per second, etc.

2.4.7 Plug and Play Device Id

This attribute can be used for automatic driver download and other automatic configuration tasks.

2.4.8 Model

This is a simple text string defined by the manufacturer.

2.4.9 Manufacturer

This is a simple text string defined by the manufacturer. There is no registration, and there is a possibility of overlap, but the goal

is to keep this simple, not too complex.

2.4.10 Type

This is the printing mechanism of the print device: laser, ink jet, thermal, etc.

2.4.11 PDLs Supported

This is a list of all of the page description languages (PDLs) that the printer and/or its interpreter(s) support.

2.4.12 Sides Supported

This is either a 1 or a 2 to indicate the maximum number of sides on which the printer can automatically print.

2.5 OIDs

This protocol makes use of Object Identifiers (OIDs). All OIDs used in this protocol are defined encoded using the OBJECT IDENTIFIER ASN.1 syntax and the BER encoding of OBJECT IDENTIFIER.

This specification does not introduce any new OIDs. The following rules are used:

- Since LDPA is a small subset of DPA, for all attributes and values which are OIDs and are defined as within the scope of this specification, the OIDs from DPA will be used.
- For any extensions to this specification that fall within DPA operations or semantics, OIDs from DPA will be used.
- For any vendor specific extensions, OIDs from the appropriate enterprise arcs in the OID tree will be used.

3. Internet Printing Model

3.1 Object Instances

All instances of all objects have an identifier attribute that makes them unique so that they can be unambiguously referenced. In the object-oriented model, these are the globally unique object references which are created by factories or constructors.

The following objects have the following mandatory identifier attributes:

Object	Identifier	Containing Object
Printer	printer-name	None
Job	job-identifier	Printer
Document	document-sequence-number	Job
Initial Value Job	TBD	Printer
Initial Value Document	TBD	Printer

3.2 Limits and Defaults

This LDPA specification does not include any mechanism for specifying for enforcing "limits" or any other kinds constraints. However, defaults are achieved through the implementation of Initial Value Job and Initial Value Document objects.

3.3 List Object Attribute Scoping Rules

The LIST-OBJECT-ATTRIBUTES operation is used for various reasons. The first of which, is to list contained jobs under a given Printer. Listing jobs works in this manner, according to the designator in the LOA (LIST-OBJECT-ATTRIBUTES) request:

1. LIST_OP_ORDERED_JOBS
 - Lists scheduled jobs. Does not include retained jobs.
2. List objects (job class) with specified instances. Includes retained jobs.
 - Lists specified jobs. If a job is not found as a current job, LOA looks for it as a retained job.
3. List objects (job class) without specified instances. Includes retained jobs; retained jobs are listed after current jobs.
 - If the client is bound to a printer agent, lists all jobs for that printer agent.
 - If only retained jobs are desired, the retained job state may be specified in a filter.

The second reason that the LIST-OBJECT-ATTRIBUTES operation is used is to query the database to find out about contained object relationships such as "What are the initial value objects for a given Printer?". The rules for these types of operations are:

1. List objects without specified instances.
 - Lists all contained objects

4. Operations

LDPA defines 7 operations:

- Bind
- Unbind
- Print
- Cancel Job
- List Object Attributes
- Modify Job
- Resubmit Job

4.1 Common Data Structures

This section describes the common data structures that are used by two or more operations.

4.1.1 XDR

```
670  /*
671  // Note: Text is stored in XDR structures in Unicode, to eliminate
672  // problems in comparing disparate forms of text.
673  // Unicode characters are always kept in low-high byte order
674  // structures.
675  //
676  // The Text structure is defined as opaque, for efficiency
677  // in marshalling / unmarshalling operations.
678  // This means that the array item count contains the number
679  // of bytes, rather than the number of 16-bit characters.
680  */
681
682  typedef opaque Text<>;
683
684
685  /* ----- job identifier ----- */
686  struct PrtContainedObjectId {
687      Text          printerName;
688      nuint32       localIdentifier;
689  };
690  typedef PrtContainedObjectId PrtContainedObjectIdSet<>;
691
692  /* ----- document identifier ----- */
693  struct DocumentIdentifier {
694      PrtContainedObjectId jobIdentifier;
695      nuint32              documentNumber;
696  } ;
697
698
699  /*
700  // Often times it is necessary for an object to have an
701  // attribute whose value is the "identifier" of another object.
702  // These attributes used an attribute syntax as defined below
703  */
704
705  enum ObjectIdentificationEnum {
706      OBJ_ID_PRT_CONTAINED_OBJ_ID = 0,
707      OBJ_ID_DOCUMENT_IDENTIFIER = 1,
708      OBJ_ID_OBJECT_IDENTIFIER = 2,
709      OBJ_ID_OBJECT_NAME = 3,
710      OBJ_ID_NAME_OR_OID = 4,
711      OBJ_ID_SIMPLE_NAME = 5,
712      OBJ_ID_PRT_CONFIG_OBJ_ID = 6
713  };
714  typedef enum ObjectIdentificationEnum ObjectIdentificationEnum;
715
716  struct ObjectIdentification {
717      ObjectIdentificationEnum designator;
718      union {
719          PrtContainedObjectId prtContainedObjectId;
720          DocumentIdentifier documentIdentifier;
721          ObjectIdentifier objectIdentifier;
722          DistinguishedNameString objectName;
723          NameOrOid nameOrOid;
724          Text simpleName;
725          PrtConfigObjectId prtConfigObjectId;
726      } ObjectIdentification_u;
```

```
727     };
728     typedef struct ObjectIdentification ObjectIdentification;
729
730     typedef AttributeValue AttributeValueSet<>;
731
732     /*
733     ** NOTE:
734     **     Sending an empty sequence for values allows an attribute
735     **     to be set as if it was not specified. This is primarily
736     **     for use in the modify function.
737     */
738
739     /* ----- attribute ----- */
740     struct Attribute {
741         ObjectIdentifier    attributeId;
742         AttributeValueSet    valueSet;
743         nuint32              qualifier;
744     };
745
746     typedef Attribute AttributeSet<>;
747
748     typedef Attribute CommonArguments<>;
749
750     enum NameOrOidEnum {
751         NAME_OR_OID_NONE,          /* 0 */
752         NAME_OR_OID_GLOBAL,        /* 1 */
753         NAME_OR_OID_LOCAL          /* 2 */
754     };
755
756     union NameOrOid switch(NameOrOidEnum designator) {
757         case NAME_OR_OID_NONE:
758             void;
759         case NAME_OR_OID_GLOBAL:
760             ObjectIdentifier globalForm;
761         case NAME_OR_OID_LOCAL:
762             Text              localForm;
763     } ;
764
765     /* ----- distinguishedNameString 9.1.5.7 ----- */
766     struct DistinguishedNameString {
767         Text              name;
768         NameOrOid         *syntaxOptionPtr;
769     };
770
771     enum QualifiedNameEnum {
772         QUALIFIED_NAME_NONE,
773         QUALIFIED_NAME_SIMPLE,
774         QUALIFIED_NAME_OTHER
775     };
776
777     struct OtherName {
778         Text              object;
779         Text              otherOption;
780     };
781
782     union QualifiedName switch (QualifiedNameEnum designator) {
783         case QUALIFIED_NAME_NONE:
```

```
784         void;
785     case QUALIFIED_NAME_SIMPLE:
786         Text                simpleName;
787     case QUALIFIED_NAME_OTHER:
788         OtherName            otherName;
789     };
790
791     typedef QualifiedName QualifiedNameSet<>;
792
793     typedef DistinguishedNameString DistinguishedNameStrSeq<>;
794
795     /*
796     // Note: The value syntax for time attributes is
797     // implemented as Cardinal.
798     */
799
800
801 4.1.2 ASN.1
802
803
804     -- The following constants are used in later ASN.1 data types
805     --
806     -- ub-integer    = 2147483647 - biggest int = 2**31-1
807     -- ub-message-string = 4095
808     -- ub-name-string  = 255
809     -- ub-octet-string = 255
810     --
811
812     SimpleName ::= CHOICE {
813         iso-646-irv    [0] VisibleString(SIZE(0..ub-name-string)),
814         ccitt-t-61     [1] T61String(SIZE(0..ub-name-string)),
815         iso-latin1     [2] Latin1String(SIZE(0..ub-name-string)),
816         iso-ucs-2      [3] UCS2Level2String(SIZE(0..ub-name-string))}
817
818     AttributeId ::= OBJECT IDENTIFIER
819
820     Attribute    ::= SEQUENCE {
821         attribute-id      [0] AttributeId,
822         attribute-values  [1] SET OF ANY -- DEFINED BY attribute-id -- }
823
824     CommonArguments ::= SET OF Attribute
825
826     JobIdentifier ::= PrintableString (SIZE (1..255))
827
828     Message ::= CHOICE {
829         iso-646-irv    [0] VisibleString(SIZE(0..ub-message-string)),
830         ccitt-t-61     [1] T61String(SIZE(0..ub-message-string))
831         iso-latin1     [2] Latin1String(SIZE(0..ub-message-string)),
832         iso-ucs-2      [3] UCS2Level2String(SIZE(0..ub-message-string))}
833
834     PositiveInteger ::= INTEGER (1..ub-integer)
835
836     DeltaTime ::= INTEGER (0..ub-integer)
837
838     Cardinal ::= INTEGER (0..ub-integer)
839
840     NameOrOid ::= CHOICE {
```



```
841     global-form  [0] OBJECT IDENTIFIER,
842     local-form   [1] SimpleName }
843
844 DistinguishedNameString ::= SEQUENCE {
845     name          [0] Text,
846     name-syntax   [1] NameOrOid OPTIONAL }
847
848 Global-Name
849     FROM ISO-STANDARD-9541-FONT-RESOURCE
850     { iso(1) standard(0) 9541 2 1 }
851
852 FontReference ::= CHOICE {
853     simple-font-name  [0] SimpleName,
854     iso-9541-font-name [1] Global-Name }
855
856 AttributeValueAssertion ::= SEQUENCE {
857     attribute-id      [0] AttributeId,
858     attribute-values  [1] SET OF ANY -- DEFINED BY attribute-id
859
860
861 GeneralizedTime ::= from ISO 8824
862
863
864 ErrorMessage ::= SEQUENCE {
865     data          [0] CHOICE {
866         iso-646-irv  [0] VisibleString(SIZE(0..ub-message-string)),
867         ccitt-t-61   [1] T61String(SIZE(0..ub-message-string)),
868         iso-latin-1  [2] Latin1String(SIZE(0..ub-message-string)),
869         iso-ucs-2    [3] UCS2Level2String(SIZE(0..ub-message-string)),
870         other-code-set [4] OCTET STRING(SIZE(0..ub-message-string)) },
871
872 4.2 Errors
873
874 This section identifies each of the individual error that might be
875 returned in any of the operation results.
876
877 4.2.1 ASN.1
878
879 AccessProblem ::= CHOICE {
880     standard-problem      ENUMERATED {
881         inappropriate-object-class (1),
882         insufficient-access-rights (2),
883         cannot-interrupt-job      (3),
884         inappropriate-object-state (4) },
885     extended-problem      OBJECT IDENTIFIER }
886
887 AccessErrorSequence ::= SEQUENCE OF SEQUENCE {
888     object-identification [0] ObjectIdentification,
889     problem               [1] AccessProblem,
890     error-message         [2] ErrorMessage }
891
892 AttributeProblem ::= CHOICE {
893     standard-problem      ENUMERATED {
894         invalid-attribute-syntax (2),
895         undefined-attribute-type (3),
896         inappropriate-matching  (4),
897         constraint-violation     (5),
```

```
898         unsupported-attribute-type           (6),
899         illegal-modification                   (7),
900         inconsistent-with-other-attributes     (8),
901         undefined-attribute-value              (9),
902         unsupported-attribute-value            (10),
903         invalid-non-compulsory-attribute-modification (11),
904         per-job-attribute-inadmissible         (12),
905         not-multi-valued                       (13),
906         mandatory-attribute-omitted           (14),
907         attribute-illegal-for-object-class     (15) },
908     extended-problem OBJECT IDENTIFIER }
909
910 AttributeErrorSequence ::= SEQUENCE {
911     object-identification [0] ObjectIdentification OPTIONAL,
912     problems               [1] SEQUENCE OF SEQUENCE {
913         problem            [0] AttributeProblem,
914         attribute          [1] Attribute,
915         error-message      [2] ErrorMessage } }
916
917 DocumentAccessProblem ::= CHOICE {
918     standard-problem      ENUMERATED {
919         document-not-available (1),
920         referent-modified     (2),
921         access-denied         (3),
922         unknown-document      (4),
923         no-documents-in-job   (5) },
924     extended-problem      OBJECT IDENTIFIER }
925
926 DocumentAccessErrorSequence ::= SEQUENCE {
927     problem                [0] DocumentAccessProblem,
928     object-identification [1] ObjectIdentification,
929     error-message          [2] ErrorMessage }
930
931 PrinterProblem ::= CHOICE {
932     standard-problem      ENUMERATED {
933         printer-error          (1),
934         printer-needs-attention (2),
935         printer-needs-key-operator (3) },
936     extended-problem      OBJECT IDENTIFIER }
937
938 PrinterErrorSequence ::= SEQUENCE {
939     problem                [0] PrinterProblem,
940     object-identification [1] ObjectIdentification,
941     error-message          [2] ErrorMessage }
942
943 SecurityProblem ::= CHOICE {
944     standard-problem      ENUMERATED {
945         inappropriate-authentication (1),
946         invalid-credentials          (2),
947         insufficient-operation-rights (3),
948         invalid-pac                  (4) },
949     extended-problem      OBJECT IDENTIFIER }
950
951 SecurityErrorSequence ::= SEQUENCE {
952     problem                [0] SecurityProblem,
953     error-message          [1] ErrorMessage }
954
955 SelectionProblem ::= CHOICE {
956     standard-problem      ENUMERATED {
957         invalid-identification (1),
958         unknown-identification (2),
959         object-already-exists   (3) }
```

```
955         extended-problem          OBJECT IDENTIFIER }
956 SelectionErrorSequence ::= SEQUENCE OF
957     SEQUENCE {
958         problem                    [0] SelectionProblem,
959         attribute                  [1] Attribute OPTIONAL,
960         object-identification      [2] ObjectIdentification,
961         error-message              [3] ErrorMessage }
962 ServiceProblem ::= CHOICE {
963     standard-problem ENUMERATED {
964         server-busy                (1),
965         server-unavailable         (2),
966         operation-too-complex     (3),
967         resource-limit-exceeded   (4),
968         unclassified-server-error (5),
969         too-many-items-in-list    (6),
970         compulsory-resource-not-available (7),
971         cancel-document-unsupported (8),
972         modify-document-unsupported (9),
973         print-multiple-documents-unsupported (10),
974         unsupported-parameter-value (11),
975         invalid-checkpoint        (12),
976         invalid-continuation-context (13),
977         pause-limit-exceeded     (14),
978         unsupported-operation     (15) },
979     extended-problem          OBJECT IDENTIFIER }
980 ServiceErrorSequence ::= SEQUENCE OF
981     SEQUENCE {
982         problem                    [0] ServiceProblem,
983         attribute                  [1] Attribute OPTIONAL,
984         object-identification      [2] ObjectIdentification,
985         error-message              [3] ErrorMessage }
986 UpdateProblem ::= CHOICE {
987     standard-problem ENUMERATED {
988         no-modifications-allowed (1),
989         insufficient-update-rights (2),
990         previous-operation-incomplete (4),
991         cancellation-not-possible (5) },
992     extended-problem          OBJECT IDENTIFIER }
993 UpdateErrorSequence ::= SEQUENCE {
994     problem                    [0] UpdateProblem,
995     object-identification      [1] ObjectIdentification,
996     error-message              [2] ErrorMessage }
997
```

998 5. Binding and Unbinding

1000 There are two special operations that are defined for establishing a
1001 "session" between a client and a server. These are the BIND and the
1002 UNBIND operations.

1004 5.1 Bind Operation

1006 5.1.1 Bind Argument

1008 The following abstract data types are part of the Bind Argument:

1011	Printer Name	The name instance of the Print Service Provider (Printer object) to which the bind is being done.
1012	Credentials	These can simple (name of the client performing the Bind) or the actual opaque Credential from some security/authorization service. All LDPA implemenations must support at least the simple option.
1013 1014	Other Security Info	Optional additional opaque security information if needed for a given security/authorization service.

5.1.1.1 XDR

```

1015
1016
1017
1018     struct Creds {
1019         Text          name;
1020         opaque        password<>;
1021     };
1022
1023     struct Other1 {
1024         string        serverNamePtr<>;
1025         nuint16       connection;
1026     };
1027
1028     struct Othern {
1029         nuint16       othern;
1030     };
1031
1032     enum CredentialsEnum {
1033         CREDENTIALS_SIMPLE,           /* (0) */
1034         CREDENTIALS_CERTIFIED,        /* (1) */
1035         CREDENTIALS_OTHER_1,          /* (2) */
1036         CREDENTIALS_OTHER_2,          /* (3) */
1037         /* ... */
1038         CREDENTIALS_OTHER_n           /* (n) */
1039     };
1040
1041     union Credentials switch(CredentialsEnum designator) {
1042     case CREDENTIALS_SIMPLE:
1043         Creds          simple;
1044     case CREDENTIALS_CERTIFIED:
1045         opaque        certified<>;
1046     case CREDENTIALS_OTHER1:
1047         struct Other1  other1;
1048     };
1049
1050     struct BindPrinterArgument {
1051         QualifiedName  printerId;
1052         Credentials    credentials;
1053         nint32         retrieveRestrictionsOption;
1054         opaque        bindSecurityOption<>;
1055     };
1056

```

5.1.1.1 ASN.1

```

1059 PrivilegeAttributeCertificate ::= EXTERNAL
1060
1061 Creds ::= SEQUENCE {
1062     name          [0] DistinguishedNameString,
1063     password      [1] OCTET STRING }
1064
1065 Credentials ::= CHOICE {
1066     simple          [0] Creds,      -- used for initial
1067                                   -- authentication --
1068     certified       [1] PrivilegeAttributeCertificate }
1069     -- used when initial authentication has already taken place
1070     -- external to the DP-Server -
1071
1072 Restrictions ::= SET {
1073     maximum-result-length [1] ResultLength OPTIONAL }
1074     -- default is no restriction --
1075
1076 ResultLength ::= INTEGER (1..ub-integer)
1077
1078 BindSecurity ::= EXTERNAL
1079
1080 DpBindArgument ::= SEQUENCE {
1081     credentials      [0] Credentials,
1082     retrieve-restrictions [1] Restrictions OPTIONAL,
1083                                   -- default is none--
1084     bind-security    [2] BindSecurity OPTIONAL }
1085

```

5.1.2 Bind Result

The following abstract data types are part of the Bind Result:

Results	The authentication attributes
Errors	Optional Error information
Session Handle	Session Handle

5.1.2.1 XDR

```

1096 struct BindResult {
1097     OctetString      authAttributeSet<>;
1098     ErrorReturn      *errorReturnOptionPtr;
1099     nint32           sessionHandle;
1100 };
1101

```

5.1.2.2 ASN.1

```

1104 AuthenticationAttribute ::= EXTERNAL
1105
1106 DpBindResult ::= SET {
1107     authentication-attributes [0] SET OF AuthenticationAttribute }
1108
1109 DpBindError ::= CHOICE {
1110     service-error [0] ServiceProblem,
1111     security-error [1] SecurityProblem }
1112

```

5.2 Unbind Operation

5.2.1 Unbind Argument

The following abstract data types are part of the Unbind Argument:

Session Handle	Session Handle
----------------	----------------

5.2.1.1 XDR

```
struct UnbindArgument {
    nint32      sessionHandle;
};
```

5.2.1.2 ASN.1

```
DpUnbind ::= ABSTRACT-UNBIND
FROM { dp-user[S], dp-administration[S] }
```

5.2.2 Unbind Result

The following abstract data types are part of the Unbind Argument:

Errors	Optional Error Information
--------	----------------------------

5.2.2.1 XDR

```
struct UnbindResult {
    ErrorReturn      *errorReturnOptionPtr;
};
```

5.2.2.2 ASN.1

No arguments or errors associated with Unbind.

6. User Operations

6.1 Print Operation

6.1.1 Print Argument

The following abstract data types are part of the Print Argument:

Session Handle	The handle for this session.
Create Job	One of the three modes for the Print Arguments (Create Job, Add Document, Close Job). If it is a Create Job, the Job Id is returned in the Print Results.
Printer Name	
Job Submission Complete	

1162 1163	Job Attributes	
1164 1165 1166	First Document Description	Transfer method, content, type, and Document Attributes
1167	Add Document	
1168	Job Id	The job to which this document is added.
1169 1170 1171	Job Submission Complete	
1172 1173 1174	First Document Description	Transfer method, content, type, and Document Attributes
1175	Close Job	
1176	Job Id	The job to close (no more documents can be added)
1177	Common Arguments	Common to all three forms of Print Argument

6.1.1.1 XDR

```

1182     struct DocumentDescription {
1183         ObjectIdentifier      transferMethod;
1184         DocumentContent       *documentContentOptionPtr;
1185         ObjectIdentifier      documentType;
1186         AttributeSet          documentAttributes;
1187     };
1188
1189     struct CreateJob {
1190         QualifiedName          printerName;
1191         bool                   jobSubmissionComplete;
1192         AttributeSet          jobAttributes;
1193         DocumentDescription    *firstDocumentOptionPtr;
1194         CommonArguments        commonArgumentsOption;
1195     };
1196
1197     struct AddDocument {
1198         PrtContainedObjectId    existingJob;
1199         bool                   jobSubmissionComplete;
1200         DocumentDescription    *newDocumentPtr;
1201         CommonArguments        commonArgumentsOption;
1202     };
1203
1204     struct CloseJob {
1205         PrtContainedObjectId    existingJob;
1206         CommonArguments        commonArgumentsOption;
1207     };
1208
1209     enum PrintArgEnum {
1210         PRINT_ARG_CREATE_JOB,          /* (0) */

```

```

1211     PRINT_ARG_ADD_DOCUMENT,          /* (1) */
1212     PRINT_ARG_CLOSE_JOB              /* (2) */
1213 };
1214
1215 union PrintOperation switch(PrintArgEnum designator) {
1216 case PRINT_ARG_CREATE_JOB:
1217     CreateJob      createJob;
1218 case PRINT_ARG_ADD_DOCUMENT:
1219     AddDocument    addDocument;
1220 case PRINT_ARG_CLOSE_JOB:
1221     CloseJob       closeJob;
1222 };
1223
1224 struct PrintArgument {
1225     nint32          sessionHandle;
1226     PrintOperation  printOperation;
1227 };
1228
1229 6.1.1.2 ASN.1
1230
1231 DocumentDescription ::= SEQUENCE {
1232     transfer-method      [0] OBJECT IDENTIFIER
1233         DEFAULT id-val-transfer-method-with-request,
1234     document-content     [1] DocumentContent OPTIONAL,
1235     document-type        [2] OBJECT IDENTIFIER
1236         DEFAULT id-val-document-type-printable,
1237     document-attributes  [3] SET OF Attribute OPTIONAL
1238     -- Contains any document attributes valid for the document,
1239     -- except any document-status attributes.
1240     -- document-type = printable, font, or resource.
1241     -- If document-type is font, a font-identifier attribute is
1242     -- required in the document-attributes element.
1243     -- If document type is resource, a resource-name attribute
1244     -- is required in the document-attributes element. }
1245
1246 PrintArgument ::= CHOICE {
1247     create-job           [0] SEQUENCE {
1248         printer-name      [0] SimpleName,
1249         job-submission-complete [1] BOOLEAN DEFAULT TRUE,
1250         job-attributes    [2] SET OF Attribute OPTIONAL,
1251         -- may include any job attribute, except
1252         -- id-att-job-identifier,
1253         -- id-att-printer-name-requested, and
1254         -- any job-status attribute
1255         first-document    [3] DocumentDescription OPTIONAL,
1256         common-arguments  [4] CommonArguments OPTIONAL },
1257     add-document         [1] SEQUENCE {
1258         existing-job      [0] JobIdentifier,
1259         job-submission-complete [1] BOOLEAN DEFAULT TRUE,
1260         new-document      [3] DocumentDescription,
1261         common-arguments  [4] CommonArguments OPTIONAL },
1262     close-job            [2] SEQUENCE {
1263         existing-job      [0] JobIdentifier,
1264         common-arguments  [4] CommonArguments OPTIONAL } }
1265
1266 6.1.2 Print Result
1267

```


The following abstract data types are part of the Print Result:

Job Id	Used for all other operations on this Job.
Server State	Optional state information about the Print Service Provider
Message	Optional message
Document Status	Optional document status information
Job Status	Job state information
Errors	Optional Error Information

6.1.2.1 XDR

```

struct PrintResult {
    PrtContainedObjectId    jobIdentification;
    ObjectIdentifier        serverStateOption;
    NameOrOid               *serverMessageOptionPtr;
    AttributeSet            documentStatusOption;
    AttributeSet            jobStatus;
    ErrorReturn              *errorReturnOptionPtr;
};

```

6.1.2.2 ASN.1

```

PrintResult ::= SEQUENCE {
    job-identification    [0] JobIdentifier,
                        -- value of id-att-job-identifier
    server-state           [1] OBJECT IDENTIFIER OPTIONAL,
                        -- value of id-att-server-state
    server-message         [2] Message OPTIONAL,
                        -- value of server's id-att-message
    document-status        [3] SET OF Attribute OPTIONAL,
                        -- may include id-att-document-state,
                        -- id-att-document-sequence-number,
                        -- id-att-file-reference, and
                        -- id-att-copies-completed.
    job-status             [4] SET OF Attribute
                        -- See document-status attributes subclause.
                        -- may include any job-status attributes
                        -- See job-status attributes subclause.
}

```

6.2 Cancel Job Operation

6.2.1 Cancel Job Argument

The following abstract data types are part of the Cancel Job Argument:

Session Handle	The handle for this session.
Job Id	The identifier of the job to be canceled.

Document Number	Optional document number of the document to cancel within a given job.
Message	Optional message to the operator.
Retention Period	Optional period for retaining the cancelled job.
Common Arguments	

6.2.1.1 XDR

```

struct CancelJobArgument {
    nint32      sessionHandle;
    PrtContainedObjectId  jobIdentifier;
    nuint32     documentNumberOption;
    NameOrOid   *cancelMessageOptionPtr;
    IntegerOption  retentionPeriodOption;
    CommonArguments  commonArgumentsOption;
};

```

6.2.1.2 ASN.1

```

CancelJobArgument ::= SEQUENCE {
    job-identification      [0] JobIdentifier,
    document-number         [1] PositiveInteger OPTIONAL,
                           -- required for addressing individual
                           -- documents in a multiple document print-job
    cancel-message          [2] Message OPTIONAL,
                           -- sets value of id-att-job-message-from-administrator
    retention-period        [3] DeltaTime OPTIONAL,
    common-arguments        [4] CommonArguments OPTIONAL }

```

6.2.2 Cancel Job Result

The following abstract data types are part of the Cancel Job Result:

Job Status	Optional Job status information
Errors	Optional Error Information

6.2.2.1 XDR

```

struct CancelJobResult {
    AttributeSet      jobStatusOption;
    ErrorReturn       *errorReturnOptionPtr;
};

```

6.2.2.2 ASN.1

```

CancelJobResult ::= SEQUENCE {
    status            [0] SET OF Attribute OPTIONAL
                           -- any job-status or document-status attributes }

```

6.3 List Object Attributes Operation

6.3.1 List Object Attributes Argument

The following abstract data types are part of the List Object Attributes Argument:

Session Handle	Handle for this session.
Operation	CONTINUE or SPECIFICATION
SPECIFICATION	
Class	The class type for which this operation is being performed (Printer, Job, Document, etc.)
Scope	Levels of object containment to report
Selector	A set of object identifiers (possibly wild carded), optional filter information, time limits, and count limits.
Requested Attributes	A set of attributes in which the requestor is interested
Operation	ATTRIBUTES or ORDERED_JOBS if requesting Jobs contained by a given Printer.
CONTINUATION	
Context	Context for continuing
Abort	Should the operation be aborted? (boolean)
Common Arguments	

6.3.1.1 XDR

```

struct Selector {
    ObjectIdentificationSeq objectIdentificationSeqOption;
    Filter                  *objectFilterOptionPtr;
    uint32                  timeLimitOption;
    uint32                  countLimitOption;
};

enum ListOperatorEnum {
    LIST_OP_ATTRIBUTES,          /* (0) */
    LIST_OP_ORDERED_JOBS = 2    /* (1) */
};

struct ListSpecification {
    ObjectIdentifier    objectClass;
    uint32              scope;          /* default 0; */
    Selector            *selectorOptionPtr;
    ObjectIdentifierSet *requestedAttrsOptionPtr;
    ListOperatorEnum    listOperator;
    /* default DpaReturnAttributes */
    CommonArguments     commonArgumentsOption;
};

```

```

1411 struct ListContinuation {
1412     OctetString      context;
1413     bool             abort;
1414     CommonArguments   commonArgumentsOption;
1415 };
1416
1417 enum ListAttrsArgEnum {
1418     LIST_ATTRIBUTES_ARG_CONTINUE,      /* (0) */
1419     LIST_ATTRIBUTES_ARG_SPEC           /* (1) */
1420 };
1421
1422 union ListAttrsOperation switch(ListAttrsArgEnum designator) {
1423 case LIST_ATTRIBUTES_ARG_CONTINUE:
1424     ListContinuation      continuation;
1425 case LIST_ATTRIBUTES_ARG_SPEC:
1426     ListSpecification     specification;
1427 };
1428
1429 struct ListObjectAttrsArgument {
1430     nint32              sessionHandle;
1431     ListAttrsOperation   listAttrsOperation;
1432 };
1433

```

6.3.1.2 ASN.1

```

1436 SubstringMatchCriteria ::= ENUMERATED {
1437     exact                (0),
1438     case-insensitive     (1),
1439     same-letter          (2), -- ignoring accents, case, etc.
1440     approximate         (3) -- implementation-defined -- }
1441
1442 FilterItem ::= CHOICE {
1443     equality              [0] AttributeValueAssertion,
1444     substrings           [1] SEQUENCE {
1445         attribute-id      [0] AttributeId,
1446         match-criteria    [1] SubstringMatchCriteria,
1447         initial-string     [2] ANY OPTIONAL,
1448                             -- DEFINED BY attribute-id
1449         any-string        [3] SEQUENCE OF ANY OPTIONAL,
1450                             -- DEFINED BY attribute-id
1451         final-string      [4] ANY OPTIONAL },
1452                             -- DEFINED BY attribute-id
1453     greater-or-equal     [2] AttributeValueAssertion,
1454                             -- asserted value is greater than or equal to
1455                             -- the attribute value
1456     less-or-equal        [3] AttributeValueAssertion,
1457                             -- asserted value is less than or equal to
1458                             -- the attribute value
1459     present              [4] AttributeId,
1460                             -- asserted attribute is present (with any value)
1461     subset-of            [5] AttributeValueAssertion,
1462                             -- asserted value is a subset of attribute value
1463     superset-of          [6] AttributeValueAssertion,
1464                             -- asserted value is a superset of attribute value
1465     non-null-set-intersection [7] Attribute
1466                             -- at least one of the members of the asserted --
1467                             -- value is present in the attribute value -- }

```

```
1468 Filter ::= CHOICE {
1469     item      [0] FilterItem,
1470     and       [1] SET OF Filter,
1471     or        [2] SET OF Filter,
1472     not       [3] Filter }
1473
1474 SubstringMatchCriteria ::= ENUMERATED {
1475     exact      (0),
1476     case-insensitive (1),
1477     same-letter (2), -- ignoring accents, case, etc.
1478     approximate (3) -- implementation-defined -- }
1479
1480 ContinuationContext ::= OCTET STRING
1481                        -- implementation-specific information
1482
1483 Selector ::= SET {
1484     object-identification [0] SEQUENCE OF ObjectIdentification
1485                                OPTIONAL,
1486                                -- should not be omitted if class is id-oc-document
1487     object-filter         [1] Filter OPTIONAL,
1488     time-limit            [2] DeltaTime OPTIONAL,
1489     count-limit           [3] PositiveInteger OPTIONAL }
1490
1491 ObjectIdentification ::= CHOICE {
1492     job-identifier        [0] JobIdentifier,
1493     document-identifier   [1] DocumentIdentifier,
1494     object-identifier     [2] OBJECT IDENTIFIER,
1495     object-name           [3] DistinguishedNameString,
1496     font-reference        [4] FontReference,
1497     name-or-oid           [6] NameOrOid,
1498     simple-name           [7] SimpleName }
1499
1500 DocumentIdentifier ::= SEQUENCE {
1501     job-identifier        [0] JobIdentifier,
1502     document-number       [1] PositiveInteger OPTIONAL }
1503                        -- document sequence number
1504
1505 ListOperator ::= ENUMERATED {
1506     get-attributes        (0),
1507     get-ordered-jobs      (2) }
1508
1509 ListObjectAttributesArgument ::= SEQUENCE {
1510     CHOICE {
1511         continuation      [0] SEQUENCE {
1512             context        [0] ContinuationContext,
1513             abort           [1] BOOLEAN DEFAULT FALSE,
1514             common-arguments [2] CommonArguments OPTIONAL },
1515         specification      [1] SEQUENCE {
1516             class           [0] OBJECT IDENTIFIER, -- id-oc-xxx
1517             scope           [1] Cardinal DEFAULT 0,
1518             -- scope is contained objects in levels 0 through n
1519             -- where 0 means the base object specified
1520             -- by the object-identification
1521             selector        [2] Selector OPTIONAL,
1522             -- should not be omitted if class is id-oc-document
1523             requested-attributes [3] SET OF AttributeId OPTIONAL,
1524             list-operator      [4] ListOperator
1525                                 DEFAULT get-attributes,
1526             common-arguments  [5] CommonArguments OPTIONAL } } }
```

6.3.2 List Object Attributes Result

The following abstract data types are part of the List Object Attributes Result:

Time	The operation can take an indeterminate amount of time to process. The results to a single Argument can be returned in multiple phases. This Result of for one of those phases. This processing time element is the time required for this phase of the operation.
Continuation Context	Optional opaque context information for performing another argument request on the next phase of the same operation.
Limit Encountered	Information on the type of limit that was encountered which forces the end of the operation even if there is a potential for more results. Values include TIME, COUNT, ERRORS.
Result Attributes	Attribute set containing the returned results.
Errors	Optional Error Information

6.3.2.1 XDR

```

enum LimitEncounteredEnum {
    LIMIT_ENCOUNTERED_TIME,           /* (0) */
    LIMIT_ENCOUNTERED_COUNT,          /* (1) */
    LIMIT_ENCOUNTERED_ERROR           /* (2) */
};

struct LimitEncounteredOption {
    nint32      length;                /* 0 or 1 */
    LimitEncounteredEnum value;
};

struct ObjectResult {
    ObjectIdentification  objectIdentification;
    AttributeSet          attributes;
    ObjectIdentifier      objectClass;
};

typedef ObjectResult ObjectResultSet<>;

struct ListObjectAttrsResult {
    nuint32      answerTime;
    OctetString  continuationOption;
    LimitEncounteredOption limitEncounteredOption;
    ObjectResultSet resultSet;
    ErrorReturn  *errorReturnOptionPtr;
};

```

6.3.2.2 ASN.1

```

ContinuationContext ::= OCTET STRING
                        -- implementation-specific information

LimitEncountered ::= ENUMERATED {
    time-limit          (0),
    count-limit         (1),
    error-limit         (2) }

ObjectResult ::= SEQUENCE {
    object-identification [0] ObjectIdentification,
    attributes            [1] SET OF Attribute
    object-class          [2] OBJECT IDENTIFIER },
                        -- id-oc-xxx

ListObjectAttributesResult ::= SEQUENCE {
    answer-time          [1] GeneralizedTime,
    continuation        [2] ContinuationContext OPTIONAL,
    limit-encountered    [3] LimitEncountered OPTIONAL,
    result-set          [4] SEQUENCE OF ObjectResult }

```

6.4 Modify Job Operation

6.4.1 Modify Job Argument

The following abstract data types are part of the Modify Job Argument:

Session Handle	Handle for this session.
Job Id	Which job to modify.
Document Number	Optionally the document to modify if not modifying a job attribute.
Job Attributes	Attribute set for Job attributes. Values can be modified in any of the following ways: ADD_ATTRIBUTE, REPLACE, ADD_VALUES, REMOVE_VALUES, SET_TO_DEFAULT, or REMOVE_ATTRIBUTE
Document Attributes	Attribute set for Document attributes.
Message	Optional Message.
Common Arguments	

6.4.1.1 XDR

```

enum ModifyOperatorEnum {
    MODIFY_OP_NULL,          /* (0) */
    MODIFY_OP_REPLACE,       /* (1) */
    MODIFY_OP_ADD_VALUES,    /* (2) */
    MODIFY_OP_REMOVE_VALUES, /* (3) */
    MODIFY_OP_SET_TO_DEFAULT, /* (4) */
}

```

```

1615     MODIFY_OP_REMOVE_ATTRIBUTE          /* (5) */
1616 };

```

```

1617
1618 struct ModifyJobArgument {
1619     nint32      sessionHandle;
1620     PrtContainedObjectId  jobIdentification;
1621     nuint32     documentNumberOption;
1622     AttributeSet  jobAttrModificationSet;
1623     AttributeSet  docAttrModificationSet;
1624     NameOrOid     *modifyMessageOptionPtr;
1625     CommonArguments  commonArgumentsOption;
1626 };

```

6.4.1.2 ASN.1

```

1631 JobAttrModification ::= SEQUENCE {
1632     attribute-id      [0] AttributeId,
1633     -- Any job attributes, except:
1634     -- id-att-job-identifier,
1635     -- id-att-job-owner, id-att-job-originator,
1636     -- id-att-printer-name-requested,
1637     -- id-att-initial-value-job,
1638     -- any access-and-accounting attributes,
1639     -- any job-security attributes, and
1640     -- any job-status attributes.
1641     -- Any document attributes, except:
1642     -- id-att-transfer-method, id-att-document-content,
1643     -- id-att-initial-value-document, and
1644     -- any document-status attributes
1645     attribute-values   [1] SET OF ANY
1646     -- DEFINED BY attribute-id -- OPTIONAL,
1647     -- omitted for set-to-default
1648     modify-operator    [2] ModifyOperator DEFAULT replace }

```

```

1650 ModifyOperator ::= ENUMERATED {
1651     replace            (0),
1652     add-values         (1),
1653     remove-values      (2),
1654     set-to-default     (3) }

```

```

1656 ModifyJobArgument ::= SEQUENCE {
1657     job-identification  [0] JobIdentifier,
1658     document-number     [1] PositiveInteger OPTIONAL,
1659     -- required for addressing individual
1660     -- documents in a multiple document print-job
1661     job-attr-modification  [2] SEQUENCE OF JobAttrModification,
1662     modify-message        [3] Message OPTIONAL,
1663     -- sets value of id-att-job-message-from-administrator
1664     common-arguments      [4] CommonArguments OPTIONAL }

```

6.4.2 Modify Job Result

The following abstract data types are part of the Modify Job Result:

Modify Status	Modify result attributes.
Errors	Optional Error Information

6.4.2.1 XDR

```

struct ModifyJobResult {
    AttributeSet          statusOption;
    ErrorReturn          *errorReturnOptionPtr;
};

```

6.4.2.2 ASN.1

```

ModifyJobResult ::= SEQUENCE {
    status          [0] SET OF Attribute OPTIONAL
    -- any job-status or document-status attributes }

```

6.5 Resubmit Job Operation

6.5.1 Resubmit Job Argument

The following abstract data types are part of the Resubmit Argument:

Session Handle	Handle for this session.
Destination Printer Name	Optional name of the destination printer.
Destination Printer Address	The address of the destination printer (can be used instead of the name).
Operation	MOVE or COPY
Job Set	A set of jobs to move or copy. Each entry in the set has: Job Id, Document Number, Job attributes, and Document attributes.
Message	Optional Message
Common Arguments	

6.5.1.1 XDR

```

enum ResubmitOpEnum {
    RESUBMIT_OP_COPY,          /* (0) */
    RESUBMIT_OP_MOVE          /* (1) */
};

/*
// If documentNumber is 0, docAttrSet is applied to all documents
*/

struct ResubmitJob {
    PrtContainedObjectId      jobId;
    nuint32                  documentNumber;
    AttributeSet              jobAttrSet;
};

```

```

1718     AttributeSet          docAttrSet;
1719 };
1720
1721 typedef ResubmitJob ResubmitJobSet<>;
1722
1723 struct ResubmitJobsArgument {
1724     nint32                sessionHandle;
1725     QualifiedName         destPrinterNameOption;
1726     NetAddress            *destPrinterNetAddressPtr;
1727     ResubmitOpEnum        operation;
1728     ResubmitJobSet        resubmitJobSet;
1729     NameOrOid             *resubmitMessageOptionPtr;
1730     CommonArguments       commonArgumentsOption;
1731 };

```

6.5.1.2 ASN.1

```

1735 ResubmitJobArgument ::= SEQUENCE {
1736     object-class           [0] OBJECT IDENTIFIER,
1737                           -- id-oc-job, id-oc-printer,
1738                           -- id-oc-server
1739     object-identification [1] ObjectIdentification,
1740     printer               [2] DistinguishedNameString,
1741     message               [3] Message OPTIONAL,
1742     common-arguments      [4] CommonArguments OPTIONAL }
1743

```

6.5.2 Resubmit Job Result

The following abstract data types are part of the Resubmit Job Result:

Resubmit Job Set	A set of jobs that were resubmitted. Each element in the set has: Old Job Id, New Job Id, and an attribute set with info about the results of the move or copy.
Errors	Optional Error Information

6.5.2.1 XDR

```

1754 struct ResubmitJobResult {
1755     PrtContainedObjectId  oldJobIdentifier;
1756     PrtContainedObjectId  newJobIdentifier;
1757     AttributeSet         jobStatusOption;
1758 };
1759
1760 typedef ResubmitJobResult ResubmitJobResultSet<>;
1761
1762 struct ResubmitJobsResult {
1763     ResubmitJobResultSet  resubmitJobResultSet;
1764     ErrorReturn           *errorReturnOptionPtr;
1765 };

```

6.5.2.2 ASN.1

```

1769 ObjectStatus ::= SEQUENCE {

```

```
1770     object-status [0] SET OF Attribute OPTIONAL }
1771     -- job-identifier and new-job-identifier shall be
1772     -- returned at least. For any jobs that could not
1773     -- be resubmitted, the new-job-identifier attribute
1774     -- shall be omitted as the only error indication.
1775
1776     ResubmitJobResult ::= SEQUENCE {
1777         result-set      [0] SEQUENCE OF ObjectStatus }
1778         -- one result-set for each job resubmitted
1779         -- (or for each job attempted to be resubmitted)
```

7. Object Attributes

This section describes the attributes and their associated values that are part of the LDPA protocol. The list below shows the objects and their attributes that are included within the scope of this protocol:

Job Attributes

Job Informational Attributes

- job-identifier
- job-owner
- job-name

Printer Selection Attributes

- printer-name-requested

Job Status Attributes

- current-job-state
- printers-assigned
- submission-time
- print-checkpoint
- job-message-from-administrator
- completion-time
- job-state-reasons
- number-of-documents
- job-submission-complete

Job Results Handling Attributes

- job-sheets
- document-sheets

Job Event Handling Attributes

- notification-profile

Job Scheduling Instructions Attributes

- job-hold
- job-priority
- job-print-after
- job-retention-period

Document Attributes

Document Description Attributes

- document-format
- document-content
- transfer-method

Document Production Instruction Attributes

- default-font
- default-medium
- number-up
- finishing
- sides

1827 copy-count
1828 reset-printer
1829 Document Characteristics Attributes
1830 fonts-used
1831 media-used
1832 Document Status Attributes
1833 document-sequence-number
1834 Operation Attributes
1835 operation-locale
1836 default-delivery-addresses
1837 Printer Attributes
1838 printer-name
1839 printer-state
1840 message
1841 printer-initial-value-job
1842 printer-initial-value-document
1843 fonts-supported
1844 fonts-ready
1845 media-supported
1846 media-ready
1847 printer-associated-printers
1848 document-formats-supported
1849 numbers-up-supported
1850 finishings-supported
1851 sides-supported
1852 job-sheets-supported
1853 document-sheets-supported
1854 maximum-copies-supported
1855 notification-delivery-methods-supported
1856 server-name
1857 server-state
1858 physical-printers-supported
1859 logical-printers-supported
1860 events-supported
1861 transfer-methods-supported
1862 locales-supported
1863 multiple-documents-supported
1864 cancel-individual-document-supported
1865 modify-individual-document-supported
1866 Initial Value Job Attributes
1867 Initial Value Document Attributes

1868
1869 In the following sections, most of the text has been taken word for
1870 word from ISO/IEC 10175 DPA (Final, June 1996).

1871 1872 7.1 Job Attributes

1873
1874 A job object contains a set of job attributes and one or more
1875 document objects. The server shall create a printable job object in
1876 response to a client that invokes one or more Print
1877 abstract-operations. In addition, initial-value-job objects are
1878 created in a server by means outside the scope of this part of
1879 ISO/IEC 10175 in order to represent complete sets of default values
1880 for job attributes (see the initial-value-job object class).

1881
1882 In addition to the attributes specifically defined for the job and
1883 initial-value-job objects, certain of the generic attributes may also

be associated with these objects. For example, when requesting a list of attribute values for an object of these classes, the client may identify one or more of the generic attributes in the following table, for which the server shall return values if the attributes are implemented.

There are no notification profiles included in this LDPA specification.

There is a table for each attribute that shows its: name, syntax, multi or single valuedness (S or M), and any relevant notes.

7.1.1 Job Informational Attributes

These attributes provide information to identify a print-job.

The client may specify job-information attributes in:

- a) Print: all, except id-att-job-identifier
- b) ModifyJob: all, except id-att-job-identifier, id-att-job-owner
- c) ListObjectAttributes: all

7.1.1.1 job-identifier

job-identifier	jobIdentifierSyntax	S	
----------------	---------------------	---	--

This attribute provides the job-identifier for this job on the server. The server shall generate a job-identifier value that is unique on that server, but need not be unique across the distributed environment.

The value of the job-identifier attribute shall be returned by the server as part of the PrintResult in the first Print abstract-operation for the job. The client shall pass its value as part of the argument in subsequent abstract-operations for the same job.

7.1.1.2 job-owner

job-owner	distinguishedNameStringSyntax	S	
-----------	-------------------------------	---	--

Attribute value types that specify the name of an object, file, or person as a string that can be either (1) a simple name by itself or (2) a simple name qualified with a path name employ this generic data type and syntax. If the path name is included, an optional name-syntax element may be used to specify the syntax of the path name, i.e., to identify the name syntax of the service being used. If the name-syntax element is omitted, the server shall assume the name-syntax is identified by some other means.

The following standard values are defined for use in the name-syntax element to identify the syntax of names:

Descriptive Name	Object Identifier	Descriptor Text
------------------	-------------------	-----------------

1938	automatic	id-val-dn-syntax-automatic	server recognizes the
1939			syntax
1940	X-500	id-val-dn-syntax-x-500	ISO/IEC 9594
1941			Directory Service
1942	XFN	id-val-dn-syntax-xfn	X/OPEN Federated
1943			Names
1944	DCE	id-val-dn-syntax-dce	Distributed Computing
1945			Environment -
1946			includes X.500 and
1947			CDS
1948			
1949	CDS	id-val-dn-syntax-cds	Cell Directory
1950			Service - part of DCE
1951	NIS	id-val-dn-syntax-nis	Network Information
1952			Service
1953	DNS	id-val-dn-syntax-dns	Domain Name Service
1954	DEC-NS	id-val-dn-syntax-dec-ns	Digital Name Service
1955	Internet-mail	id-val-dn-syntax-internet-mail	Internet Mail
1956			address
1957	XNS	id-val-dn-syntax-xns	Xerox Network System
1958	Bindery	id-val-dn-syntax-bindery	
1959	NDS	id-val-dn-syntax-nds	Novell Directory
1960			Service
1961	URL	id-val-dn-syntax-url	HTTP Universal
1962			Resource Locator
1963	POSIX	id-val-dn-syntax-posix	POSIX file name
1964			(ISO/IEC 9945-1)
1965	UNIX	id-val-dn-syntax-unix	UNIX(TM) file name
1966	OS/2	id-val-dn-syntax-os2	OS/2 file name
1967	PC-DOS	id-val-dn-syntax-pc-dos	PC DOS file name
1968	NT	id-val-dn-syntax-nt	NT file name
1969	MVS	id-val-dn-syntax-mvs	MVS file name
1970	VM	id-val-dn-syntax-vm	VM file name
1971	OS/400	id-val-dn-syntax-os400	OS/400 file name
1972	VMS	id-val-dn-syntax-vms	VMS file name
1973	UNC	id-val-dn-syntax-unc	Microsoft Universal
1974			Name Convention
1975	NetWare	id-val-dn-syntax-netware	NetWare file path
1976			name

As with any NameOrOid, implementors may use their own object identifiers or simple names (if they have not assigned an OID) for implementation-defined name-syntaxes.

This attribute supplies the name of the human owner of the print-job.

The value of job-owner will often be the same as job-originator. The job-owner will be different from job-originator when the job has been submitted by the originator on behalf of the owner.

If this attribute is not specified, the value of user-name or job-originator should be used for any circumstances which require a value for job-owner.

7.1.1.3 TBD

7.1.1.4 job-name

job-name	simpleNameSyntax	S	
----------	------------------	---	--

This attribute supplies a human readable string for the print-job. This string is used for naming the print-job in human-readable "free-form" fashion.

This attribute is intended for enabling a user or the user's application to convey a job name that may be printed on a start sheet, returned in a ListObjectAttributes result, or used in notification or logging messages.

If this attribute is not specified, no job name is assumed, but implementation specific defaults are allowed, such as the value of the document-name attribute of the first document in the job.

7.1.2 Printer Selection Attributes

These attributes provide information to help select a particular printer. If more than one printer-selection attribute is specified, the server shall select a printer that meets all of the criteria.

The client may specify printer-selection attributes in:

- a) Print: all, except the value of printer-name-requested (which shall be passed as an explicit parameter of the first PrintArgument, rather than as an attribute)
- b) ModifyJob: all, except printer-name-requested
- c) ListObjectAttributes: all

7.1.2.1 printer-name-requested

printer-name-requested	simpleNameSyntax	S	
------------------------	------------------	---	--

This attribute identifies the printer to be used for printing the job. The client shall specify the value of this attribute with the first invocation of the Print abstract-operation for the print-job as the explicit printer-name component of the PrintArgument, rather than as an attribute.

NOTES

1 To cause a server to select a printer according to other attributes, the system administrator should define a logical printer that supports the desired set of physical printers.

2 Initial-value-job objects should have the value of their printer-name-requested attribute specified as an empty value in order to indicate that no printer-name is defaulted.

7.1.3 Job Status Attributes

These attributes specify the job status before, during and after the processing of the print-job by the server. The server shall create the job object with these attributes (if implemented) and shall assign appropriate values to each such job-status attribute.

The client may specify job-status attributes in:

- a) Print: none
- b) ModifyJob: none
- c) ListObjectAttributes: all

7.1.3.1 current-job-state

current-job-state	objectIdentifierSyntax	S	
-------------------	------------------------	---	--

This attribute identifies the current state of the job (pending, printing, held, etc.).

The following job state standard values are defined:

id-val-job-state-unknown, id-val-job-state-pre-processing,
id-val-job-state-held, id-val-job-state-pending,
id-val-job-state-processing, id-val-job-state-paused,
id-val-job-state-interrupted, id-val-job-state-terminating,
id-val-job-state-retained, id-val-job-state-completed

The LDPA protocol supports all values for job states, but printers are not required to generate all job states, only those which are appropriate for the particular implementation.

If a printer implementation or policy is to start processing documents before the last print-request (with a TRUE value for the job-submission-complete parameter), the printer may change the job's current-job-state from pre-processing directly to the processing state when the printer begins processing any of the job's documents.

7.1.3.2 printers-assigned

printers-assigned	simpleNameSeqSyntax	S	
-------------------	---------------------	---	--

This attribute identifies the physical printer or printers to which this job has been assigned, if any.

When the job is first submitted and the printer has not yet assigned any printers to the job, the SEQUENCE shall be empty.

If the printer intends to use a single printer for the job, and the printer has assigned a printer to the job, the SEQUENCE shall contain just that printer.

If a printer has split the job into multiple pieces and assigned each piece to a different printer, the SEQUENCE shall contain n elements, one for each assigned printer. A job with multiple job-result-sets is an example of a job that would be easy to split into multiple pieces.

A SEQUENCE with no elements shall be returned if this attribute is supported, but this job has not yet been assigned to any physical printers.

The number of elements in the SEQUENCE for this attribute shall be the same as the number of elements in the SEQUENCE for the associated job attribute printer-state-of-printers-assigned.

In addition, the ith element of the value of printer-state-of-printers-assigned shall be the state of the printer named by the ith element of printers-assigned.

The printers-assigned value shall not be the same as the printer requested by the user if the job's printer-name-requested attribute specified a logical printer that supports one or more different physical printers. The printers-assigned value might differ also if the job has been re-assigned by an operator to ensure successful completion of the job, allowing the user to find out where a job has been re-assigned (when necessary).

The value of the job's printers-assigned attribute shall remain after the job has completed, so that users can determine the physical printer(s) on which the job was printed.

7.1.3.3 submission-time

submission-time	generalizedTimeSyntax	S	
-----------------	-----------------------	---	--

This attribute indicates the time at which the latest print request or this job was accepted by the printer. If the printer does not support the notion of time, the attribute is not stored as part of the job object.

7.1.3.4 print-checkpoint

print-checkpoint	printCheckpointSyntax	S	
------------------	-----------------------	---	--

This attribute indicates the job-copies, document-copies, pages, and octets completed for the document or documents on the specified printer(s) and the local context information at which the last checkpoint was taken.

This attribute allows a print service to provide information about a checkpoint of a job that is printing. This would indicate where a print-server could resume printing of this job - at a page and copy number of a document close to the point at which the job was paused due to malfunction or operator request.

The context-info element shall contain information that would be needed by the server and printer to enable them to resume printing at the last checkpoint. The format of this element identified by the checkpoint-format element. The content of this element is implementation-specific; the intent is that the client would return this element, without alteration, in a ResumeJobArgument in order to resume the job.

NOTE - A server should encode the value of context-info in such a way as to protect against clients submitting ResumeJob requests with

altered context-info.

The checkpoint-format element shall identify the encoding format used for the context-info element. Standard values are defined in the printer attribute checkpoint-format-supported.

Some systems support concurrent printing of a job on multiple printers. In such cases, the server shall return a PrintCheckpoint sequence for each printer currently assigned to the job.

The ability to generate the previous internal state of the job and the printer is dependent on the page-independence supported by the document format. If a document format is not page-independent, it may be possible to emulate the resumption of the job at the checkpoint by processing through the entire document to the checkpoint page without printing any additional pages, then continue printing pages from that point. Some document formats may not support any form of checkpointing.

If a PauseJob operation causes a job to pause in the middle of a document encoded in a document format that does not support checkpointing, the server shall set the checkpoint to a value that will force the system to resume back at the beginning of the current copy. Obviously the ability to checkpoint a job is very implementation-dependent.

7.1.3.5 job-message-from-administrator

job-message-from-administrator	simpleNameSyntax	S	
--------------------------------	------------------	---	--

This attribute provides a message from an operator, system administrator or 'intelligent' process to indicate to the user the reasons for modification or other management action taken on a job.

7.1.3.6 completion-time

completion-time	generalizedTimeSyntax	S	
-----------------	-----------------------	---	--

This attribute indicates the time at which this job completed. Providing this time is useful for jobs which are retained after printing.

7.1.3.7 job-state-reasons

job-state-reasons	objectIdentifierSyntax	M	
-------------------	------------------------	---	--

This attribute identifies the reason or reasons that the job is in the state that it is in (e.g., held, terminating, retained, completed, etc.). The printer shall indicate the particular reason(s) by setting the value of the job-state-reasons attribute.

It is valid for the printer to set the value of the job-state-reasons attribute to the empty set.

The following standard values are defined:

id-val-reasons-documents-needed, id-val-reasons-job-hold-set,
id-val-reasons-job-print-after-specified,
id-val-reasons-required-resources-not-ready,
id-val-reasons-successful completion,
id-val-reasons-completed-with-warnings,
id-val-reasons-completed-with-errors,
id-val-reasons-cancelled-by-user,
id-val-reasons-cancelled-by-operator,
id-val-reasons-aborted-by-system, id-val-reasons-logfile-pending,
id-val-reasons-logfile-transferring

7.1.3.8 number-of-documents

number-of-documents	cardinalSyntax	S	
---------------------	----------------	---	--

This attribute indicates the number of documents in the job. The number indicates how many Print abstract-operations that specified a document (of any document-type) have been submitted for printing until job submission has been completed; at that point this attribute shall then indicate the total number of printable documents, fonts, and resource objects submitted by the client in the job. If the first Print abstract-operation does not contain a first-document component, the value of this attribute shall be 0.

The server shall count fonts and resource objects passed to the server by means of Print abstract-operation invocations, as documents for the purposes of this attribute.

NOTE - the value of the number-of-documents attribute represents the total number of documents that the client has submitted to the server during the course of job submission, regardless of whether or not the client has cancelled any of the documents. See CancelJob abstract-operation.

7.1.3.9 job-submission-complete

job-submission-complete	booleanSyntax	S	
-------------------------	---------------	---	--

This attribute indicates whether all documents of the print-job have been submitted (i.e., all Print abstract-operations have been invoked for the job). The value FALSE indicates that more documents are expected to be submitted for the job, by means of additional print invocations.

7.1.4 Job Results Handling Attributes

These attributes specify the actions to be undertaken after printing of a job has been completed. This includes assembly of documents

into job sets, finishing operations applied to job sets, and delivery of the completed job sets.

The client may specify job-results-handling attributes in:

- a) Print: all
- b) ModifyJob: all
- c) ListObjectAttributes: all

7.1.4.1 job-sheets

job-sheets	nameOrOidSyntax	S	
------------	-----------------	---	--

Attribute value types that encode identifiers that may have either a global-form or a local-form employ this generic syntax or datatype in their definitions.

The global-form is of the object identifier type, and is expected to be used wherever such a value has been defined for the object in question. The local-form is intended for local implementation convenience, for use when a global-form is not available or has not been defined for the object to be identified.

NOTE - It must be stressed that the local-form is not guaranteed to be unique, since there are no procedures in place to control the creation and usage of simple-name types. It is possible for two different sites to create and use the same simple-name to identify two different entities. If those two sites are interconnected subsequently, unexpected results can occur because of this duplication of simple-names. For this reason, the local-form is to be used only for purely local or temporary purposes; the global-form must be used in all other cases.

This attribute specifies the auxiliary-sheets that the server shall insert into the job as separators, covers, and trailers.

7.1.4.2 document-sheets

document-sheets	nameOrOidSyntax	S	
-----------------	-----------------	---	--

This attribute is similar to job-sheets. The difference is that it applies to documents within the job rather than the job itself.

7.1.5 Job Event Handling Attributes

7.1.5.1 notification-profile

notification-profile	eventHandlingProfileSyntax	M	
----------------------	----------------------------	---	--

This attribute is a specification of events about which the user and/or designate are to be notified. In addition, this attribute specifies how the event notifications are to be delivered.

Printers may produce the same information for notification and logging or they may produce different information, depending on implementation.

7.1.6 Job Scheduling Instructions Attributes

These attributes provide additional hints for the scheduling of a print-job. How a print-service uses this information in scheduling jobs is implementation-specific.

The client may specify job-scheduling-instruction attributes in:

- a) Print: all
- b) ModifyJob: all
- c) ListObjectAttributes: all

7.1.6.1 job-hold

job-hold	booleanSyntax	S	
----------	---------------	---	--

This attribute specifies whether the print-job is a candidate for scheduling for printing or not, when the server would otherwise place the job in the pending or processing states. The PauseJob and ResumeJob operations may be used independently of the value of this attribute.

When the value is FALSE, the printer shall not hold the job from being scheduled for printing, unless there are other reasons (see the current-job-state and the job-state-reasons job-status attributes).

When the value is TRUE, the printer shall place the job in the held state and add the job-hold-set value to the job's job-state-reasons attribute and shall not schedule the print-job for printing. If the job enters the held state because its job-hold attribute was TRUE, a client shall reset the job's job-hold attribute to FALSE by means of the ModifyJob operation before the printer can schedule the job for printing. When the value is set to FALSE as a result of the ModifyJob operation, the printer shall remove the job-hold-set value from the job-state-reasons attribute and, if no other reasons remain, shall change the job's current-job-state to pending so that the job becomes a candidate for being scheduled on printer(s).

7.1.6.2 job-priority

priority	prioritySyntax	S	
----------	----------------	---	--

This attribute specifies a priority for scheduling the print-job. It is used by servers that employ a priority-based scheduling algorithm.

A higher value specifies a higher priority. The value 1 is defined to indicate the lowest possible priority (a job which a priority-based scheduling algorithm shall pass over in favour of higher priority jobs). The value 100 is defined to indicate the highest possible priority. Priority is expected to be evenly or 'normally' distributed across this range. The mapping of vendor-defined priority over this

range is implementation-specific. The omission of this attribute implies that the user places no constraints concerning priority on the scheduling of the print-job.

7.1.6.3 job-print-after

job-print-after	generalizedTimeSyntax	S	
-----------------	-----------------------	---	--

This attribute specifies the calendar date and time of day after which the print-job shall become a candidate to be scheduled for printing.

If the value of this attribute is in the future, the server shall set the value of the job's current-job-state to held and add the job-print-after-specified value to the job's job-state-reasons attribute and shall not schedule the print-job for printing until the specified date and time has passed. When the specified date and time arrives, the server shall remove the job-print-after-specified value from the job's job-state-reason attribute and, if no other reasons remain, shall change the job's current-job-state to pending so that the job becomes a candidate for being scheduled on printer(s).

The printer shall assign an empty value to the job-print-after attribute when no print after time has been assigned or when it does not support the notion of time within the printer, so that the job shall be a candidate for scheduling immediately.

7.1.6.4 job-retention-period

job-retention-period	deltaTimeSyntax	S	
----------------------	-----------------	---	--

Delta time provides an integer value for a period of elapsed time, measured in seconds.

This attribute specifies the minimum period of time following the completion of job processing and printing that the server shall keep job attributes, document attributes, and document data. The server may keep these attributes and data longer than the value of the job-retention-period attribute.

Job-retention-period specifies a lower bound on how long job attributes, document attributes and document data shall be retained by a server after printing has completed, whilst job-discard-time sets an upper bound on retention of the job and document attributes independent of whether the job is ever scheduled for, starts or completes printing.

In addition to providing status information to a user after a job has completed printing, the job-retention-period also provides the mechanism for retaining job's document data after it has been printed, so that the job may be printed again, possibly with modified attributes, such as the job-copies component of the job-results attribute.

NOTE - The mechanism to reprint the job is outside the scope of this part of ISO/IEC 10175; part 3 of this International Standard (in preparation) includes a Resubmit abstract-operation to enable this function.

7.2 Document Attributes

A document object contains a set of document attributes, including the document-content attribute which specifies the document data. A document object may be of type printable, font, or resource as specified by the document's document-type attribute. The printer shall create document objects as contained members of job objects in response to a client that performs one or more print-requests (see 8.2.1). In addition, initial-value-document objects are created in a server by means outside the scope of this part of ISO/IEC 10175 in order to represent complete sets of default values for document attributes (see the initial-value-document object class). This subclause of ISO/IEC 10175 specifies the document attributes for both document and initial-value-document objects.

In addition to the attributes specifically defined for the document and initial-value-document objects, certain of the generic attributes may also be associated with these objects.

NOTE -There are no attributes that apply to both the job and document objects. Thus the server may return both job and document attributes mixed together without ambiguity in the ModifyJob and CancelJob requests.

7.2.1 Document Description Attributes

These attributes identify the document and its characteristics and specify the method by which the document is acquired by the print-server.

The client may specify document-description attributes in:

- a) Print: all, except document-type, transfer-method, and document-content shall be passed as explicit parameters of the Print abstract-operation and shall not be passed as attributes.
- b) ModifyJob: all, except id-att-transfer-method, and id-att-document-content
- c) ListObjectAttributes: all, except id-att-document-content

7.2.1.1 document-format

document-format	docFormatSyntax	S	
-----------------	-----------------	---	--

This attribute identifies the overall print document format used for the document. It consists of three elements, a document-format, a document-format-variants and a document-format-version. The latter two elements are optional.

The document-format element identifies a particular family of document formats, of which there may exist several versions or variants. The document-format-variants and document-format-version

elements identify a specific instance of a document format. The variant refers to a particular functional subset of a format. For example, the format PostScript has variants of level 1 and level 2, and the format PCL has several variants, including PCL4 and PCL5.

The version distinguishes among successive releases of the same basic format and variant. For example, successive versions of Xerox Interpress include versions 2.0, 2.1, 3.0, 3.1, etc.

The document-format-variants element consists of a single text string. If it is necessary to identify more than one variant, the respective variant values shall all be contained in the document-format-variants element, separated from one another by commas.

If the client omits the document-format-variants or document-format-version elements, the server may supply a format-specific default.

Proprietary values for the document-format, document-format-variants, and document-format-version elements are assigned by the owners of those formats.

7.2.1.2 document-content

document-content	documentContentSyntax	S	
------------------	-----------------------	---	--

This attribute specifies a transfer-method-specific reference for the document to be transferred. It indicates whether the content is included or referenced. If it is referenced, the reference is of syntax DOR. The DOR datatype (Distinguished Object Reference) is imported from ISO/IEC 10031-2. The DOR datatype may be used for other transfer-methods, e.g., ftam-by-server.

7.2.1.3 Reserved

7.2.1.4 transfer-method

transfer-method	objectIdentifierSyntax	S	
-----------------	------------------------	---	--

This attribute identifies the method by which the document is transferred to or acquired by the print-server.

Standard values are defined as: TBS.

Conforming client and server implementations shall support at least id-val-transfer-method-with-request, which is the default transfer-method.

7.2.2 Document Production Instruction Attributes

These attributes provide information that affect the rendering and finishing of the document and are referred to as document production

instructions (DPI). DPI may also be contained in the document to be printed.

If there is a conflict between the value of one of these attributes, and a corresponding parameter found in the document (either implicit or explicit), the value of the attribute shall take precedence over the document parameter, unless specifically mandated otherwise in the standard defining that document format.

All the default-xxx attributes (e.g. default-medium) specifically allow for the document contents to override the default-xxx attribute under all conditions.

The client may specify document production-instruction attributes in:

- a) Print: all
- b) ModifyJob: all
- c) ListObjectAttributes: all

7.2.2.1 default-font

default-font	nameOrOidSyntax	S	
--------------	-----------------	---	--

This attribute identifies a font that the server shall use as the font default for the pages of the document that require a specification.

Standard values are defined: TBD.

If the document data, itself, specifies fonts, such specification shall override the default-font attribute on a page by page basis. If the document data specifies fonts which are not also values of fonts-used, then a printer may receive a document which requires fonts which are not ready. In such a case, an implementation may either abort the document or try printing the document using some alternative fonts, such as the default font.

7.2.2.2 default-medium

default-medium	nameOrOidSyntax	S	
----------------	-----------------	---	--

This attribute identifies a medium that the server shall use as the medium default for the pages of the document that require a specification.

Standard values are defined: TBD

If the document data, itself, specifies media, such specification shall override the default-medium attribute on a page by page basis. If the document data specifies media which are not also values of media-used, then a printer may receive a document which requires media that are not ready. In such a case, an implementation may either abort the document or try printing the document on some alternative medium, such as the default medium.

A client has numerous ways to specify the media to be used when printing a document and different document pages can be specified in different ways. The client can specify the media in the document contents or with attributes. Some attributes override the document contents, and other attributes may be overridden by the document contents. In addition, the client can specify the media by name or by the input-tray containing it.

Before printing each page of a document, the server determines the medium or input-tray for that page by finding the first condition in the list of numbered steps below that is satisfied. For this discussion, either the medium or the input-tray is sufficient information:

- a) If page-media-select has a medium value for the current page, use that medium, regardless of document contents and other attributes.
- b) If input-tray-select has a value, use that tray.
- c) If the document contents specify a medium, and that medium is the same as the value of one of the original-medium elements in the media-substitution attribute, then use the corresponding substitution-medium in the media-substitution attribute.
- d) If the document contents specify a medium, use that medium.
- e) If the document contents specify an input-tray, use that input-tray.
- f) If the default-medium has a value, and the document format interpreter allows its use, and that medium is the same as the value of one of the original-medium elements in media-substitution attribute, then use the corresponding substitution-medium in the media-substitution attribute.
- g) If the default-medium has a value and the document format interpreter allows its use, use the default-medium.
- h) If the default-input-tray has a value and the document format interpreter allows its use, use the default-input-tray.
- i) Use the medium or input-tray selected by the document format processor in the printer. This selection is implementation-dependent.

7.2.2.3 number-up

number-up	cardinalOrNameOrOidSyntax	S	
-----------	---------------------------	---	--

A CardinalSyntax allows attribute values that can specify either a Cardinal or an OID (that normally names a Cardinal).

This attribute specifies the number of source page-images to impose upon a single instance of a selected medium. The attribute can be specified either by a number directly or by naming an imposition object which specifies some particular number-up imposition.

In general, only certain numeric values are valid for this attribute, depending upon the server and printer implementations to which the print-request is directed. A value of 0 or none shall suppress any server default number up, if any.

This attribute primarily controls the translation, scaling and rotation of page images, but a site may choose to add embellishments, such as borders to each logical page. A site may even choose to add an attribute to control the presence or characteristics of such embellishments.

The following standard values are defined: id-val-generic-none, id-val-imposition-simple-1-up, id-val-imposition-simple-2-up, id-val-imposition-simple-4-up.

NOTE - The value 0 or none specifies that no convenience imposition functions shall be performed; 0 or none is needed to suppress any special number-up operation because a value of 1 for some sites may cause the server to alter the placement, or size of the page image, or to add embellishments, such as borders or to rotate the page depending on content-orientation.

The server may support three values for number-up besides 0 (and id-val-generic-none), namely 1 (and id-val-imposition-1-up), 2 (and id-val-imposition-simple-2-up) and 4 (and id-val-imposition-simple-4-up), which this document will reference by the respective names of 0-up, 1-up, 2-up and 4-up, henceforth. These 1-up, 2-up and 4-up values provide a simple means for users to request the printing of compact documents of a temporary or informal nature.

7.2.2.4 finishing

finishing	finishingSyntax	M	
-----------	-----------------	---	--

This attribute identifies a sequence of one or more finishing-processes to be applied to each copy of the printed document.

Finishing encompasses the operations that may be applied to the media output of a print-job. Examples include stapling, saddle-stitching, hole-drilling, binding with tape, etc.

This attribute allows the requester to specify one or more individual finishing processes may be specified in the finishing attribute. Each of the individual processes is specified by including the required parameters for each of the individual finishing processes in the finishing attribute. Standard values for this attribute are defined: TBD.

7.2.2.5 sides

sides	sidesSyntax	S	
-------	-------------	---	--

This attribute specifies the number of printable surfaces of the medium to be imaged. SidesSyntax is an integer restricted to the range {1..2}.

7.2.2.6 copy-count

copy-count	cardinalSyntax	S	
------------	----------------	---	--

This attribute specifies the number of copies of the documents, or of the selected pages of the document, to be printed.

A value of 1 for copy-count shall generate a single human perceptible copy of the electronic document. If a value of 0 is supplied, then:

- a) if the server supports specification of the value 0, the job shall be processed normally, but no print output shall be produced; or
- b) if the server does not support specification of the value 0, the server shall return an unsupported-attribute-value `AttributeError`.

7.2.2.7 reset-printer

reset-printer	booleanSyntax	S	
---------------	---------------	---	--

This attribute specifies that the interpreter and/or printer be reset after processing this document (in a multiple document job).

This attribute would normally be used to suppress the resetting of non-page-independent interpreters and/or printers so that the previously defined state (e.g. font resources, forms, etc.) is inherited by the next document in the job.

This attribute has no meaning or effect for document formats that are page-independent, such as SPDL.

A server shall ensure that a printer is always reset after the last document in a job, independent of whether reset-printer is TRUE or FALSE for the last document, so that jobs are independent of one another.

7.2.3 Document Characteristics Attributes

This group of attributes describes the characteristics of the document to be printed.

The values provided by these attributes are intended to assist the print-server in validating and scheduling the print-job. Providing these attributes independent of the document allows the server to schedule a job or to validate the resources required to print the document without interpreting the contents of the document. This provides the opportunity for a server to support a broad set of document formats yet still support fast efficient scheduling and validation of each job. The values provided by these attributes are also intended to provide parameters to print-server services, such as a text formatter or imposition's number-up procedure.

The values of these attributes are hints to the server about production instructions and resources needed to print a document, but the printer does not use these attributes during the actual printing of a document. The values of these attributes are intended to come

from the document content, but some may come from intentions of the client. The values of these attributes are assigned as follows:

- a) First the client may, at its option, either omit these attributes, or assign values to any of them based on the document content or client intent.
- b) The server may then, at its option, either leave any of these attributes unchanged, or assign values to any of them based on its own analysis of the document contents. If the document contains incomplete information or no information about the attribute, or the server cannot ascertain the information, the server may choose to assign some default value. For xxx-used attributes which have a corresponding default-xxx, the server shall use the value of the default-xxx as the default. When a default is used with a MULTI VALUE attribute, it may be one of several values in the attribute, e.g. some pages may have an explicit medium, others may use a default.
- c) Finally, the server may choose to assign a value to these attributes only when the client does not supply a value, or the server may choose to override whatever the client supplies, or the server may also choose to do nothing, regardless of what the client supplies.

If the client performs the ModifyJob operation on any of these attributes, the server shall follow rules b and c, above, for the modified attributes. Thus, in effect, the server shall have control over whether to honor a client's requested change.

For validation and scheduling, the server shall use these attributes and shall not examine the document contents. However, according to the rules above, the server may have examined the document contents earlier to assign values to these attributes.

Processes, such as text formatting and number-up may use some of these attributes as parameters, or they may do their own independent analysis during the procedure.

The client may specify document-characteristic attributes in:

- a) Print: all
- b) ModifyJob: all
- c) ListObjectAttributes: all

7.2.3.1 fonts-used

fonts-used	fontReferenceSyntax	M	
------------	---------------------	---	--

This attribute identifies the font resources specified in the document.

7.2.3.2 media-used

media-used	nameOrOidSequenceSyntax	S	
------------	-------------------------	---	--

This attribute identifies the media specified in the document.

The values in this attribute should contain the actual media required for printing the document, taking into account the results of interpreting the document contents, and applying the attributes: page-media-select, input-tray-select, media-substitution, default-medium and default-input-tray.

Standard values for this attribute are defined: TBD

This attribute contains a SEQUENCE of values rather than a SET because the ith element of this attribute corresponds to the ith attribute of the assured-reproduction-areas-used attribute.

This attribute is intended for scheduling and validation. The server uses this attribute with the printer attributes media-supported for validation and media-ready for scheduling.

7.2.4 Document Status Attributes

These attributes specify the document status, before, during, and after processing of the document by the server. The server shall create the document object with these attributes (if implemented) and shall assign appropriate values to each such document-status attribute.

The client may specify document-status attributes in:

- a) Print: none
- b) ModifyJob: none
- c) ListObjectAttributes: all

7.2.4.1 document-sequence-number

document-sequence-number	cardinalSyntax	S	
--------------------------	----------------	---	--

This attribute specifies the number of this document in relation to the set of documents in this job. The first document in the job is numbered 1.

The document-sequence-number is not passed as an input attribute in the Print abstract-operation. Documents are assumed to be submitted in order (i.e., document number 1 followed by document number 2, etc.).

A server shall return a value of 0 for this attribute if the first Print abstract-operation has not submitted a document (e.g., the first-document element is omitted in the create-job element of the Print abstract-operation).

7.3 Operation Attributes

TBD

7.4 Printer Attributes

A printer object may represent either a physical printer or a logical printer, or both.

A physical printer is a printer object containing a set of printer object attributes that represent an output device capable of rendering a document in visible form. Examples include electronic and electro-mechanical printers such as laser printers, ink-jet printers, and various kinds of impact printers, but may include other types of output devices such as microfiche imagers and plotters as well.

A logical printer is a printer object containing a set of printer object attributes that have been grouped under one name in order to represent some class of printer or printing effect. For example, an administrator might define a single logical printer to represent all of the physical printers of the same type and capability in a single location, associated with a particular server. A user/client would normally send a print-job to a logical printer, and allow the server to assign the job to a particular physical printer based on the relative load and availability of the printers under its control, thus providing a load balancing service. However, ISO/IEC 10175 does not preclude a user/client from sending a print-job to a physical printer. Such a restriction is up to the policy of the system administrator and the access control that the administrator specifies.

Logical and physical printer objects may be defined to specify that a particular set of default values for job and document attributes are to be assumed when a client identifies that printer. Such things as default media, fonts, finishing operations, etc., may be specified for a job simply by sending the job to a particular logical or physical printer. When the client identifies a logical printer, the server shall assign the job to a particular physical-printer that the administrator has explicitly associated with the logical-printer. Depending on implementation, the server may assign the job when the job is received, or the server may delay the assignment, until a physical printer is free, thereby achieving more dynamic load balancing between several physical printers.

A printer object shall have one of three realizations, as specified by the value of its printer-realization attribute logical, physical, or logical-and-physical. The logical-and-physical value is used in the simple and frequent case when the system administrator creates a single printer object to represent both a logical printer and a physical printer. This would be the case when a single physical-printer is associated with a single print-server or when the administrator has decided not to offer additional sets of defaults for the physical printer. In order to create more than one set of defaults for a physical printer, the system manager shall create an associated logical printer and sets its printer-realization to logical.

If the printer-realization attribute is not implemented, the server shall treat all printer objects as if the printer-realization attribute had the value logical-and-physical.

Throughout ISO/IEC 10175, the term printer shall refer to both logical and physical printers, and shall be used when no distinction is being made between logical and physical printers. The term

logical printer shall be used for a printer object whose printer-realization attribute has the value logical or logical-and-physical. The term physical printer shall be used for a printer object whose printer-realization attribute has the value physical or logical-and-physical.

The attributes defined in this subclause provide information about a particular logical or physical printer; all of the attributes apply to logical and physical printers.

In addition to the attributes specifically defined for this object, certain of the generic attributes may also be associated with this object. For example, when requesting a list of attribute values for an object of this class, the client may identify one or more of the generic attributes in the following table, for which the server shall return values if the attributes are implemented.

7.4.1 printer-name

printer-name	simpleNameSyntax	S	
--------------	------------------	---	--

This attribute uniquely identifies the printer.

7.4.2 printer-state

printer-state	objectIdentifierSyntax	S	
---------------	------------------------	---	--

This attribute identifies the current state of the printer. The LDPA protocol support all values for printer states, however printers are not required to generate all the printer states, only those which are appropriate for the particular implementation.

The following standard values are defined:

id-val-printer-state-unknown, id-val-printer-state-idle,
id-val-printer-state-printing, id-val-printer-state-needs-attention,
id-val-printer-state-paused, id-val-printer-state-shutdown,
id-val-printer-state-job-start-wait,
id-val-printer-state-job-end-wait,
id-val-printer-state-job-password-wait,
id-val-printer-state-needs-key-operator,
id-val-printer-state-connecting-to-printer,
id-val-printer-state-timed-out

7.4.3 message

message	messageSyntax	S	
---------	---------------	---	--

This attribute provides a message from an operator, system administrator or 'intelligent' process to indicate to the user the reasons for modification or other management action taken on a job.

7.4.4 printer-initial-value-job

printer-initial-value-job	nameOrOidSyntax	S	
---------------------------	-----------------	---	--

This attribute identifies an initial-value-job object in the server for this printer. An initial-value-job object contains those attributes that the server shall default when a print-job is submitted, if the client does not specify an initial-value-job attribute with the print-request, the server shall use the initial-value-job object specified by the printer's printer-initial-value-job attribute to initialize the job object when the job is submitted and set the job's initial-value-job attribute to the value of the printer-initial-value-job attribute.

In an initial-value-job object, SINGLE VALUE attributes (1) shall contain one attribute-value or (2) may specify no attribute values, i.e., an empty attribute-value (see DPA 9.1.2). MULTI VALUE attributes shall contain zero or more attribute-values. Attributes containing no values either (1) are not supported by the printer, or (2) are expected to be defaulted by the printer hardware itself.

LDPA requires that a printer shall implement the printer-initial-value-job attribute. This requirement is important, so that the server defaulting mechanism shall permit a client to submit a print-job with many attributes omitted, and the server supplies default values.

7.4.5 printer-initial-value-document

printer-initial-value-document	nameOrOidSyntax	S	
--------------------------------	-----------------	---	--

This attribute identifies an initial-value-document object in the server for this printer. If the client does not specify an initial-value-document attribute with the print-request, the server shall use the initial-value-document object specified by the printer's printer-initial-value-document attribute to initialize the document object when the document is submitted and set the document's initial-value-document attribute to the value of the printer-initial-value-document attribute.

A printer may specify only one initial-value-document object, which will be used to initialize all document object instances targeted at this printer unless overridden by the initial-value-document attribute as described above. Each document in a job may therefore use a different initial-value-document object even though the printer may specify only one.

In an initial-value-document object, SINGLE VALUE attributes (1) shall contain one attribute value or (2) may specify no attribute values, i.e. an empty attribute-value (DPA see 9.1.2). MULTI VALUE attributes shall contain zero or more attribute-values. Attributes containing no values either (1) are not supported by the printer, or

(2) are expected to be defaulted by the printer hardware itself.

LDPA requires that a printer shall implement the printer-initial-value-document attribute. This requirement is important so that the server defaulting mechanism shall permit a client to submit a document print-request with many attributes omitted, and the server supplies default values.

7.4.6 fonts-supported

fonts-supported	fontReferenceSyntax	M	
-----------------	---------------------	---	--

This attribute identifies the font resources supported by this printer.

7.4.7 fonts-ready

fonts-ready	fontReferenceSyntax	M	
-------------	---------------------	---	--

This attribute identifies the font resources currently ready to be used on this printer.

7.4.8 media-supported

media-supported	nameOrOidSyntax	M	
-----------------	-----------------	---	--

This attribute identifies the media supported by this printer.

7.4.9 media-ready

media-ready	nameOrOidSyntax	M	
-------------	-----------------	---	--

This attribute identifies the media currently ready to be used on this printer.

7.4.10 printer-associated-printers

printer-associated-printers	distinguishedNameStringSyntax	M	
-----------------------------	-------------------------------	---	--

This attribute identifies the logical/physical printers associated with this physical/logical printer.

7.4.11 document-formats-supported

document-formats-supported	docFormatSyntax	M	
----------------------------	-----------------	---	--

This attribute identifies the document-formats, including the

document-format-variants and document-format-versions, supported by the output device and the server software collectively. This set includes both the formats that are native to the output device and those formats that the server software can translate to one that is native to the output device. From the client's point of view, this set contains all formats in which documents can be submitted to this printer.

Proprietary document format identifiers, variants, and versions are assigned by the owners of those formats.

7.4.12 numbers-up-supported

numbers-up-supported	numbersUpSupportedSyntax	S	
----------------------	--------------------------	---	--

This attribute identifies the number-up values and imposition objects supported by this printer. The cardinal-range is an alternative (shorthand) way of specifying consecutive cardinal-values.

There are no standard values defined.

7.4.13 finishings-supported

finishings-supported	nameOrOidSyntax	S	
----------------------	-----------------	---	--

This attribute identifies the per-document finishing objects supported by this printer, that is the server-installed finishing objects that may be used as values of the finishing document attribute.

NOTE: What are the values of this attribute since we have no Finishing objects.

7.4.14 sides-supported

sides-supported	sidesSyntax	M	
-----------------	-------------	---	--

This attribute indicates the values of the sides attribute supported by this printer, i.e., the different numbers of surfaces of a medium that can be imaged by this printer.

7.4.15 job-sheets-supported

job-sheets-supported	nameOrOidSyntax	M	
----------------------	-----------------	---	--

This attribute identifies the auxiliary-sheet-s values supported by this printer.

To allow no job sheets, the system administrator shall include the

value id-val-generic-none as a value for this attribute. The client specifies that there are no job sheets by using the value id-val-generic-none as the value of the job-sheets attribute.

If the job-sheets attribute is not specified or contains a value which the printer does not support, and the job-sheets value is non-compulsory (so that the server accepts the job), then the server may select from among the values of this attribute. The server shall not select the value id-val-generic-none unless it is the only value specified for the job-sheets-supported attribute.

NOTE - It is preferable for the server to produce some job auxiliary-sheet, even if not the desired one, rather than produce none at all.

7.4.16 document-sheets-supported

document-sheets-supported	nameOrOidSyntax	M	
---------------------------	-----------------	---	--

This attribute identifies the auxiliary-sheets values supported by this printer.

To allow no document sheets, the system administrator shall include the value id-val-generic-none as a value for this attribute. The client specifies that there are no document sheets by using the value id-val-generic-none as the value of the document-sheets attribute.

If the document-sheets attribute is not specified or contains a value which the printer does not support, and the document-sheets value is non-compulsory (so that the server accepts the job), then the server may select from among the values of this attribute. The server shall not select the value id-val-generic-none unless it is the only value specified for the document-sheets-supported attribute.

NOTE - It is preferable for the server to produce some job auxiliary-sheet, even if not the desired one, rather than produce none at all.

7.4.17 maximum-copies-supported

maximum-copies-supported	cardinalSyntax	S	
--------------------------	----------------	---	--

This attribute indicates the maximum number of copies of a document that can be rendered by this printer in a single print-job.

A server shall ensure that neither a document's copy-count attribute nor any single job-copies element of a ResultsProfile exceeds the value specified in this attribute. A server may ensure that for each document the product of the document's copy-count and the sum of all job-copies in all result-sets does not exceed this value.

A value of 0 shall indicate there is no limit on the maximum number

of document copies for this printer.

7.4.18 notification-delivery-methods-supported

notification-delivery-methods-supported	TBD	S	
---	-----	---	--

7.4.19 physical-printers-supported

physical-printers-supported	distinguishedNameStringSyntax	M	
-----------------------------	-------------------------------	---	--

This attribute identifies the physical printers (printer's realization attribute is either physical or logical-and-physical) supported by this server.

7.4.20 Logical-printers-supported

logical-printers-supported	distinguishedNameStringSyntax	M	
----------------------------	-------------------------------	---	--

This attribute identifies the logical printers (printer's realization attribute is either logical or logical-and-physical) supported by this server.

7.4.21 events-supported

events-supported	objectIdentifierSyntax	S	
------------------	------------------------	---	--

This attribute identifies the event types and event classes supported by this printer.

7.4.22 transfer-methods-supported

transfer-methods-supported	objectIdentifierSyntax	M	
----------------------------	------------------------	---	--

This attribute identifies the transfer-methods supported by this server.

7.4.23 locales-supported

TBD

7.4.24 multiple-documents-supported

multiple-documents-supported	booleanSyntax	S	
------------------------------	---------------	---	--

This attribute indicates whether this object (printer or server) is capable of processing and printing multiple documents per job.

This printer shall not support any operation involving multiple documents unless this attribute has the value TRUE. In spite of this requirement, it is still a printer driver implementation option of whether to support modifying and/or cancelling individual documents within a multi-document job or not.

7.4.28 cancel-individual-document-supported

cancel-individual-document-supported	booleanSyntax	S	
--------------------------------------	---------------	---	--

This attribute indicates whether this object (printer or server) is capable of cancelling the printing of individual documents within a multiple document job.

7.4.29 modify-individual-document-supported

modify-individual-document-supported	booleanSyntax	S	
--------------------------------------	---------------	---	--

This attribute indicates whether the server is capable of modifying the print-request parameters for individual documents within a multiple document job.

7.5 Initial Value Job Attributes

The attributes for an Initial Value Job object can be any of the Job object attributes defined in section 7.1.

7.6 Initial Value Document Attributes

The attributes for an Initial Value Document object can be any of the Document object attributes defined in section 7.2.

7.7 Relationship to ISO/IEC 10175 Conformance Levels

In ISO/IEC 10175 DPA Appendix E, three Conformance Levels are defined. For levels 1 and 2, an additional set of attributes for multiple-document job support are defined. These additional levels are indicated by the letter M. Thus, level 2M indicates support for a basic set of operations and attributes with additional support for multiple-document jobs. The scope of LDPA is essentially the same as level 2M as defined by DPA.

LDPA is explicitly designed to be extensible. This means that in addition to the attributes defined in this specification, specific implementation instances may support not only the basic protocol as defined in this specification, but might add vendor specific extensions.

Also, for the core set of attributes listed in this specification, it is not required that a conforming server support all (standard) values of all supported attributes. For example, it is not required that a printer implement all finishing methods indicated by the standard values.

The explicit requirement of the term "supported", with respect to one of the attributes that deal with printer functions or resources, is that the server shall recognize the attribute and those values that are supported, and shall be able to respond to a query about which values that printer does, in fact, support.

8. Security Considerations

This protocol does not identify any new security mechanisms. The authentication mechanisms (as well as extensions) built into the RPC infrastructure are recommended. Also, the Bind operation described in section 5 supports the notion of authentication via simple or credential based arguments.

9. References

- [1] Smith, R., Wright, F., Hastings, T., Zilles, S., and Gyllenskog, J., "Printer MIB", RFC 1759, March 1995.
- [2] Srinivasan, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1831, August 1995.
- [3] Srinivasan, R., "XDR: External Data Representation Standard", RFC 1832, August 1995.
- [4] Postel, J., "Instructions to RFC Authors", RFC 1543, October 1993.
- [5] ISO/IEC 10175 Document Printing Application (DPA)
- [6] Herriot, R. (editor), X/Open A Printing System Interoperability Specification (PSIS), August 1995.
- [7] Kirk, M. (editor), POSIX System Administration -- Part 4: Printing Interfaces, POSIX 1387.4 D8, 1994.
- [8] ISO 8824 "Abstract Syntax Notation One ASN.1"
- [9] ISO 8825 "Basic Encoding Rules BER"
- [10] McLaughlin, L. III, (editor), "Line Printer Daemon Protocol" RFC 1179, August 1990.

10. Author's Address

3365
3366 Scott A. Isaacson
3367 Novell, Inc.
3368 122 E 1700 S
3369 Provo, UT 84606
3370
3371 Phone: 801-861-7366
3372 Fax: 801-861-4025
3373 EMail: scott_isaacson@novell.com
3374
3375 F. Devon Taylor
3376 Novell, Inc.
3377 122 E 1700 S
3378 Provo, UT 84606
3379
3380 Phone: 801-861-7179
3381 Fax: 801-861-4025
3382 EMail: devon_taylor@novell.com
3383
3384 Mike MacKay
3385 Novell, Inc.
3386 2180 Fortune Dr.
3387 San Jose, CA 95131
3388
3389 Phone: 408-577-6368
3390 Fax: 408-577-5151
3391 EMail: mmackay@novell.com
3392
3393 Peter Zehler
3394 Xerox Corporation
3395 800 Phillips Rd. M/S 111-01X
3396 Webster, NY 14580
3397
3398 Phone: 716-265-8755
3399 Fax: 716-265-8792
3400 EMail: peter_zehler@wb.xerox.com
3401
3402 Carl-Uno Manros
3403 Xerox Corporation
3404 701 S. Aviation Blvd.
3405 El Segundo, CA 90245
3406
3407 Phone: 310-333-8273
3408 Fax: 310-333-5514
3409 EMail: cmanros@cp10.es.xerox.com
3410
3411 Tom Hastings
3412 Xerox Corporation
3413 701 S. Aviation Blvd.
3414 El Segundo, CA 90245
3415
3416 Phone: 310-333-6413
3417 Fax: 310-333-5514
3418 EMail: hastings@cp10.es.xerox.com
3419
3420