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

LDPA - Lightweight Document Printing Application

Version 0.8, October 31, 1996

Status of this Memo

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

Abstract

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

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

The LDPA protocol covers only user operations on basic print service

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

Table of Contents

1. Introduction

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

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

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

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

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

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

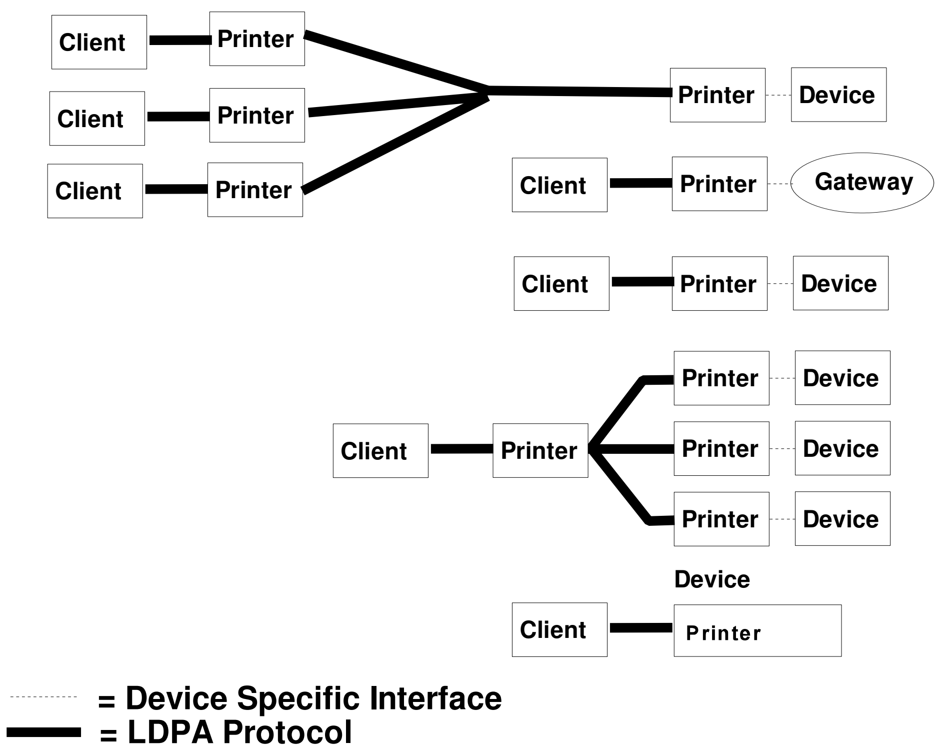

Figure 1 Printer Configurations

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 ojbect 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 LDPA 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

```
/*
//  Note: Text is stored in XDR structures in Unicode, to eliminate
//  problems in comparing disparate forms of text.
//  Unicode characters are always kept in low-high byte order
//  structures.
//
//  The Text structure is defined as opaque, for efficiency
//    in marshalling / unmarshalling operations.
//    This means that the array item count contains the number
//    of bytes, rather than the number of 16-bit characters.
*/

typedef opaque Text<>;


/* ------ job identifier ------ */
struct PrtContainedObjectId {
    Text          printerName;
    nuint32       localIdentifier;
    };
typedef PrtContainedObjectId PrtContainedObjectIdSet<>;

/* ------ document identifier ------ */
struct DocumentIdentifier {
   PrtContainedObjectId jobIdentifier;
   nuint32              documentNumber;
} ;


/*
// Often times it is necessary for an object to have an
// attribute whose value is  the "identifier" of another object.
// These attributes used an attribute syntax as defined below
*/

enum ObjectIdentificationEnum {
   OBJ_ID_PRT_CONTAIND_OBJ_ID = 0,
   OBJ_ID_DOCUMENT_IDENTIFIER = 1,
   OBJ_ID_OBJECT_IDENTIFIER = 2,
   OBJ_ID_OBJECT_NAME = 3,
   OBJ_ID_NAME_OR_OID = 4,
   OBJ_ID_SIMPLE_NAME = 5,
   OBJ_ID_PRT_CONFIG_OBJ_ID = 6
};
typedef enum ObjectIdentificationEnum ObjectIdentificationEnum;

struct ObjectIdentification {
   ObjectIdentificationEnum designator;
   union {
      PrtContainedObjectId prtContainedObjectId;
      DocumentIdentifier documentIdentifier;
      ObjectIdentifier objectIdentifier;
      DistinguishedNameString objectName;
      NameOrOid nameOrOid;
      Text simpleName;
      PrtConfigObjectId prtConfigObjectId;
   } ObjectIdentification_u;
```

```
   };
   typedef struct ObjectIdentification ObjectIdentification;

   typedef AttributeValue AttributeValueSet<>;

   /*
   ** NOTE:
   **      Sending an empty sequence for values allows an attribute
   **      to be set as if it was not specified. This is primarily
   **      for use in the modify function.
   */

   /* ------ attribute -------------- */
   struct Attribute {
      ObjectIdentifier     attributeId;
      AttributeValueSet    valueSet;
      nuint32              qualifier;
      };

   typedef Attribute AttributeSet<>;

   typedef Attribute CommonArguments<>;

   enum NameOrOidEnum {
      NAME_OR_OID_NONE,                  /* 0 */
      NAME_OR_OID_GLOBAL,                /* 1 */
      NAME_OR_OID_LOCAL                  /* 2 */
   };

   union NameOrOid switch(NameOrOidEnum designator)  {
      case NAME_OR_OID_NONE:
         void;
      case NAME_OR_OID_GLOBAL:
         ObjectIdentifier globalForm;
      case NAME_OR_OID_LOCAL:
         Text              localForm;
      } ;

   /* ------ distinguishedNameString 9.1.5.7 ------ */
   struct DistinguishedNameString {
      Text                name;
      NameOrOid          *syntaxOptionPtr;
      };

   enum QualifiedNameEnum {
      QUALIFIED_NAME_NONE,
      QUALIFIED_NAME_SIMPLE,
      QUALIFIED_NAME_OTHER
      };

   struct OtherName {
      Text                object;
      Text                otherOption;
      };

   union QualifiedName switch (QualifiedNameEnum designator) {
      case QUALIFIED_NAME_NONE:
```

```
            void;
         case QUALIFIED_NAME_SIMPLE:
            Text             simpleName;
         case QUALIFIED_NAME_OTHER:
            OtherName          otherName;
         };

      typedef QualifiedName QualifiedNameSet<>;

      typedef DistinguishedNameString DistinguishedNameStrSeq<>;

      /*
      // Note: The value syntax for time attributes is
      // implemented as Cardinal.
      */
```

4.1.2 ASN.1

```
      -- The following constants are used in later ASN.1 data types
      --
      -- ub-integer  =  2147483647 - biggest int = 2**31-1
      -- ub-message-string  =  4095
      -- ub-name-string  =  255
      -- ub-octet-string  =  255
      --

      SimpleName ::= CHOICE {
         iso-646-irv   [0] VisibleString(SIZE(0..ub-name-string)),
         ccitt-t-61    [1] T61String(SIZE(0..ub-name-string)),
         iso-latin1    [2] Latin1String(SIZE(0..ub-name-string)),
         iso-ucs-2     [3] UCS2Level2String(SIZE(0..ub-name-string))}

      AttributeId ::= OBJECT IDENTIFIER

      Attribute   ::= SEQUENCE {
         attribute-id      [0] AttributeId,
         attribute-values  [1] SET OF ANY -- DEFINED BY attribute-id -- }

      CommonArguments  ::=  SET OF Attribute

      JobIdentifier ::= PrintableString (SIZE (1..255))

      Message ::= CHOICE {
         iso-646-irv   [0] VisibleString(SIZE(0..ub-message-string)),
         ccitt-t-61    [1] T61String(SIZE(0..ub-message-string))
         iso-latin1    [2] Latin1String(SIZE(0..ub-message-string)),
         iso-ucs-2     [3] UCS2Level2String(SIZE(0..ub-message-string))}

      PositiveInteger ::= INTEGER (1..ub-integer)

      DeltaTime ::= INTEGER (0..ub-integer)

      Cardinal ::= INTEGER (0..ub-integer)

      NameOrOid ::= CHOICE {
```

```
     global-form  [0] OBJECT IDENTIFIER,
     local-form   [1] SimpleName }

   DistinguishedNameString ::= SEQUENCE {
      name          [0] Text,
      name-syntax   [1] NameOrOid OPTIONAL }

   Global-Name
       FROM ISO-STANDARD-9541-FONT-RESOURCE
           { iso(1) standard(0) 9541 2 1 }

   FontReference ::= CHOICE {
      simple-font-name    [0] SimpleName,
      iso-9541-font-name  [1] Global-Name }

   AttributeValueAssertion ::= SEQUENCE {
      attribute-id      [0] AttributeId,
      attribute-values  [1] SET OF ANY -- DEFINED BY attribute-id


   GeneralizedTime ::=  from ISO 8824


   ErrorMessage ::= SEQUENCE {
      data          [0] CHOICE {
         iso-646-irv  [0] VisibleString(SIZE(0..ub-message-string)),
         ccitt-t-61   [1] T61String(SIZE(0..ub-message-string)),
         iso-latin-1  [2] Latin1String(SIZE(0..ub-message-string)),
         iso-ucs-2    [3] UCS2Level2String(SIZE(0..ub-message-string)),
         other-code-set [4] OCTET STRING(SIZE(0..ub-message-string)) },
```

4.2 Errors

This section identifies each of the individual error that might be returned in any of the operation results.

4.2.1 ASN.1

```
   AccessProblem ::= CHOICE {
       standard-problem        ENUMERATED {
           inappropriate-object-class      (1),
           insufficient-access-rights      (2),
           cannot-interrupt-job            (3),
           inappropriate-object-state      (4) },
       extended-problem        OBJECT IDENTIFIER }

   AccessErrorSequence ::= SEQUENCE OF SEQUENCE {
           object-identification [0] ObjectIdentification,
           problem               [1] AccessProblem,
           error-message         [2] ErrorMessage }

   AttributeProblem ::= CHOICE {
       standard-problem        ENUMERATED {
           invalid-attribute-syntax                (2),
           undefined-attribute-type                (3),
           inappropriate-matching                  (4),
           constraint-violation                    (5),
```

```
            unsupported-attribute-type              (6),
            illegal-modification                    (7),
            inconsistent-with-other-attributes      (8),
            undefined-attribute-value               (9),
            unsupported-attribute-value             (10),
            invalid-non-compulsory-attribute-modification (11),
            per-job-attribute-inadmissible          (12),
            not-multi-valued                        (13),
            mandatory-attribute-omitted             (14),
            attribute-illegal-for-object-class      (15) },
       extended-problem OBJECT IDENTIFIER }

    AttributeErrorSequence ::= SEQUENCE {
            object-identification [0] ObjectIdentification OPTIONAL,
            problems              [1] SEQUENCE OF SEQUENCE {
                problem               [0] AttributeProblem,
                attribute             [1] Attribute,
                error-message         [2] ErrorMessage } }

    DocumentAccessProblem ::= CHOICE {
        standard-problem        ENUMERATED {
            document-not-available  (1),
            referent-modified       (2),
            access-denied           (3),
            unknown-document        (4),
            no-documents-in-job     (5) },
        extended-problem        OBJECT IDENTIFIER  }

    DocumentAccessErrorSequence ::= SEQUENCE {
            problem                 [0] DocumentAccessProblem,
            object-identification   [1] ObjectIdentification,
            error-message           [2] ErrorMessage }
    PrinterProblem ::= CHOICE {
       standard-problem     ENUMERATED {
          printer-error                      (1),
          printer-needs-attention       (2),
          printer-needs-key-operator    (3) },
       extended-problem           OBJECT IDENTIFIER }
    PrinterErrorSequence  ::= SEQUENCE {
          problem                    [0] PrinterProblem,
          object-identification   [1] ObjectIdentification,
          error-message           [2] ErrorMessage }
    SecurityProblem ::= CHOICE {
       standard-problem  ENUMERATED {
          inappropriate-authentication     (1),
          invalid-credentials              (2),
          insufficient-operation-rights    (3),
          invalid-pac                           (4) },
       extended-problem           OBJECT IDENTIFIER  }
    SecurityErrorSequence ::= SEQUENCE {
          problem              [0] SecurityProblem,
          error-message     [1] ErrorMessage }
    SelectionProblem ::= CHOICE {
       standard-problem  ENUMERATED {
          invalid-identification     (1),
          unknown-identification     (2) ,
          object-already-exists         (3) }
```

```
      extended-problem        OBJECT IDENTIFIER }
   SelectionErrorSequence ::=  SEQUENCE OF
      SEQUENCE {
            problem                       [0] SelectionProblem,
            attribute                     [1] Attribute  OPTIONAL,
            object-identification      [2] ObjectIdentification,
               error-message              [3] ErrorMessage }
   ServiceProblem ::= CHOICE {
      standard-problem  ENUMERATED {
         server-busy                                (1),
         server-unavailable                         (2),
         operation-too-complex                      (3),
         resource-limit-exceeded                (4),
         unclassified-server-error              (5),
         too-many-items-in-list                 (6),
         compulsory-resource-not-available          (7),
         cancel-document-unsupported            (8),
         modify-document-unsupported            (9),
         print-multiple-documents-unsupported         (10),
         unsupported-parameter-value            (11),
         invalid-checkpoint                         (12),
         invalid-continuation-context           (13),
         pause-limit-exceeded                   (14),
         unsupported-operation                      (15) },
      extended-problem           OBJECT IDENTIFIER }
   ServiceErrorSequence ::= SEQUENCE OF
         SEQUENCE {
            problem                    [0] ServiceProblem,
            attribute                  [1] Attribute OPTIONAL,
            object-identification   [2] ObjectIdentification,
            error-message           [3] ErrorMessage }
   UpdateProblem ::= CHOICE {
      standard-problem     ENUMERATED {
            no-modifications-allowed         (1),
            insufficient-update-rights    (2),
            previous-operation-incomplete    (4),
            cancellation-not-possible     (5) },
      extended-problem     OBJECT IDENTIFIER }
   UpdateErrorSequence ::= SEQUENCE {
         problem                    [0] UpdateProblem,
         object-identification   [1] ObjectIdentification,
         error-message           [2] ErrorMessage }
```

5. Binding and Unbinding

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

5.1 Bind Operation

5.1.1 Bind Argument

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

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

5.1.1.1 XDR

```
   struct Creds {
      Text               name;
      opaque             password<>;
   };

   struct Other1 {
      string             serverNamePtr<>;
      nuint16            connection;
   };

   struct Othern {
      nuint16            othern;
   };

   enum CredentialsEnum {
      CREDENTIALS_SIMPLE,                   /* (0) */
      CREDENTIALS_CERTIFIED,                /* (1) */
      CREDENTIALS_OTHER_1,                  /* (2) */
      CREDENTIALS_OTHER_2,                  /* (3) */
      /* ... */
      CREDENTIALS_OTHER_n                   /* (n) */
   };

   union Credentials switch(CredentialsEnum designator) {
   case CREDENTIALS_SIMPLE:
      Creds          simple;
   case CREDENTIALS_CERTIFIED:
      opaque         certified<>;
   case CREDENTIALS_OTHER1:
      struct Other1  other1;
   };

   struct BindPrinterArgument {
      QualifiedName      printerId;
      Credentials        credentials;
      nint32             retrieveRestrictionsOption;
      opaque             bindSecurityOption<>;
   };
```

5.1.1.1 ASN.1

```
PrivilegeAttributeCertificate ::= EXTERNAL

Creds ::= SEQUENCE {
   name           [0] DistinguishedNameString,
   password       [1] OCTET STRING  }

Credentials ::= CHOICE {
   simple            [0] Creds,   -- used for initial
                                  -- authentication --
   certified         [1] PrivilegeAttributeCertificate  }
    -- used when initial authentication has already taken place
    -- external to the DP-Server -

Restrictions ::= SET {
   maximum-result-length  [1] ResultLength OPTIONAL }
-- default is no restriction --

ResultLength ::= INTEGER (1..ub-integer)

BindSecurity ::= EXTERNAL

DpBindArgument ::= SEQUENCE {
   credentials              [0] Credentials,
   retrieve-restrictions    [1] Restrictions OPTIONAL,
                                  -- default is none--
   bind-security            [2] BindSecurity OPTIONAL }
```

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

```
struct BindResult {
   OctetString       authAttributeSet<>;
   ErrorReturn      *errorReturnOptionPtr;
   nint32            sessionHandle;
};
```

5.1.2.2 ASN.1

```
AuthenticationAttribute ::= EXTERNAL

DpBindResult ::= SET {
   authentication-attributes [0] SET OF AuthenticationAttribute }

DpBindError ::= CHOICE {
   service-error  [0] ServiceProblem,
   security-error [1] SecurityProblem }
```

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	

Job Attributes	
First Document Description	Transfer method, content, type, and Document Attributes
Add Document	
Job Id	The job to which this document is added.
Job Submission Complete	
First Document Description	Transfer method, content, type, and Document Attributes
Close Job	
Job Id	The job to close (no more documents can be added)
Common Arguments	Common to all three forms of Print Argument

6.1.1.1 XDR

```
      struct DocumentDescription {
      ObjectIdentifier        transferMethod;
      DocumentContent        *documentContentOptionPtr;
      ObjectIdentifier        documentType;
      AttributeSet            documentAttributes;
   };

   struct CreateJob {
      QualifiedName           printerName;
      bool                    jobSubmissionComplete;
      AttributeSet            jobAttributes;
      DocumentDescription    *firstDocumentOptionPtr;
      CommonArguments         commonArgumentsOption;
   };

   struct AddDocument {
      PrtContainedObjectId    existingJob;
      bool                    jobSubmissionComplete;
      DocumentDescription    *newDocumentPtr;
      CommonArguments         commonArgumentsOption;
   };

   struct CloseJob {
      PrtContainedObjectId    existingJob;
      CommonArguments         commonArgumentsOption;
   };

   enum PrintArgEnum {
      PRINT_ARG_CREATE_JOB,                /* (0) */
```

```
      PRINT_ARG_ADD_DOCUMENT,               /* (1) */
      PRINT_ARG_CLOSE_JOB                   /* (2) */
   };

   union PrintOperation switch(PrintArgEnum designator) {
   case PRINT_ARG_CREATE_JOB:
      CreateJob     createJob;
   case PRINT_ARG_ADD_DOCUMENT:
      AddDocument   addDocument;
   case PRINT_ARG_CLOSE_JOB:
      CloseJob      closeJob;
   };

   struct PrintArgument {
      nint32            sessionHandle;
      PrintOperation    printOperation;
   };
```

6.1.1.2 ASN.1

```
   DocumentDescription ::= SEQUENCE {
      transfer-method         [0] OBJECT IDENTIFIER
                     DEFAULT id-val-transfer-method-with-request,
      document-content        [1] DocumentContent OPTIONAL,
      document-type           [2] OBJECT IDENTIFIER
                     DEFAULT id-val-document-type-printable,
      document-attributes     [3] SET OF Attribute OPTIONAL
      -- Contains any document attributes valid for the document,
      -- except any document-status attributes.
      -- document-type = printable, font, or resource.
      -- If document-type is font, a font-identifier attribute is
      -- required in the document-attributes element.
      -- If document type is resource, a resource-name attribute
      -- is required in the document-attributes element. }

   PrintArgument ::= CHOICE {
      create-job     [0] SEQUENCE {
         printer-name             [0] SimpleName,
         job-submission-complete [1] BOOLEAN DEFAULT TRUE,
         job-attributes           [2] SET OF Attribute OPTIONAL,
                             -- may include any job attribute, except
                             -- id-att-job-identifier,
                             -- id-att-printer-name-requested, and
                             -- any job-status attribute
         first-document           [3] DocumentDescription OPTIONAL,
         common-arguments         [4] CommonArguments OPTIONAL },
      add-document   [1] SEQUENCE {
         existing-job             [0] JobIdentifier,
         job-submission-complete [1] BOOLEAN DEFAULT TRUE,
         new-document             [3] DocumentDescription,
         common-arguments         [4] CommonArguments OPTIONAL },
      close-job      [2] SEQUENCE {
         existing-job             [0] JobIdentifier,
         common-arguments         [4] CommonArguments OPTIONAL } }
```

6.1.2 Print Result

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.
                     -- See document-status attributes subclause.
   job-status           [4] SET OF Attribute
                     -- 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;
   nuint32                 timeLimitOption;
   nuint32                 countLimitOption;
};

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

struct  ListSpecification {
   ObjectIdentifier        objectClass;
   nuint32                 scope;              /* default 0; */
   Selector               *selectorOptionPtr;
   ObjectIdentifierSet    *requestedAttrsOptionPtr;
   ListOperatorEnum        listOperator;
            /* default DpaReturnAttributes */
   CommonArguments         commonArgumentsOption;
};
```

```
   struct ListContinuation {
      OctetString             context;
      bool                    abort;
      CommonArguments         commonArgumentsOption;
   };

   enum ListAttrsArgEnum  {
      LIST_ATTRIBUTES_ARG_CONTINUE,             /* (0) */
      LIST_ATTRIBUTES_ARG_SPEC                  /* (1) */
   };

   union ListAttrsOperation switch(ListAttrsArgEnum designator) {
   case LIST_ATTRIBUTES_ARG_CONTINUE:
      ListContinuation     continuation;
   case LIST_ATTRIBUTES_ARG_SPEC:
      ListSpecification    specification;
   };

   struct ListObjectAttrsArgument {
      nint32                  sessionHandle;
      ListAttrsOperation      listAttrsOperation;
   };
```

6.3.1.2 ASN.1

```
   SubstringMatchCriteria ::= ENUMERATED {
    exact            (0),
    case-insensitive (1),
    same-letter      (2), -- ignoring accents, case, etc.
    approximate      (3)  -- implementation-defined -- }

   FilterItem ::= CHOICE {
     equality           [0] AttributeValueAssertion,
     substrings         [1] SEQUENCE {
         attribute-id       [0] AttributeId,
         match-criteria     [1] SubstringMatchCriteria,
         initial-string     [2] ANY OPTIONAL,
                                  -- DEFINED BY attribute-id
         any-string         [3] SEQUENCE OF ANY OPTIONAL,
                             -- DEFINED BY attribute-id
         final-string       [4] ANY OPTIONAL },
                                  -- DEFINED BY attribute-id
     greater-or-equal   [2] AttributeValueAssertion,
                     -- asserted value is greater than or equal to
                     -- the attribute value
     less-or-equal      [3] AttributeValueAssertion,
                         -- asserted value is less than or equal to
                         -- the attribute value
     present            [4] AttributeId,
                  -- asserted attribute is present (with any value)
     subset-of          [5] AttributeValueAssertion,
                   -- asserted value is a subset of attribute value
     superset-of        [6] AttributeValueAssertion,
                 -- asserted value is a superset of attribute value
     non-null-set-intersection [7] Attribute
                  -- at least one of the members of the asserted --
                  -- value is present in the attribute value  -- }
```

```
   Filter ::= CHOICE {
      item     [0] FilterItem,
      and      [1] SET OF Filter,
      or       [2] SET OF Filter,
      not      [3] Filter }

   SubstringMatchCriteria ::= ENUMERATED {
      exact            (0),
      case-insensitive (1),
      same-letter      (2), -- ignoring accents, case, etc.
      approximate      (3)  -- implementation-defined -- }

   ContinuationContext ::= OCTET STRING
                           -- implementation-specific information
   Selector ::= SET {
      object-identification [0] SEQUENCE OF ObjectIdentification
                                 OPTIONAL,
                  -- should not be omitted if class is id-oc-document
      object-filter           [1] Filter OPTIONAL,
      time-limit              [2] DeltaTime OPTIONAL,
      count-limit             [3] PositiveInteger OPTIONAL }

   ObjectIdentification ::= CHOICE {
      job-identifier          [0] JobIdentifier,
      document-identifier     [1] DocumentIdentifier,
      object-identifier       [2] OBJECT IDENTIFIER,
      object-name             [3] DistinguishedNameString,
      font-reference          [4] FontReference,
      name-or-oid             [6] NameOrOid,
      simple-name             [7] SimpleName }

   DocumentIdentifier ::= SEQUENCE {
      job-identifier          [0] JobIdentifier,
      document-number         [1] PositiveInteger OPTIONAL }
                                  -- document sequence number
   ListOperator ::= ENUMERATED {
      get-attributes                (0),
      get-ordered-jobs              (2) }

   ListObjectAttributesArgument ::= SEQUENCE {
      CHOICE {
         continuation      [0] SEQUENCE {
            context              [0] ContinuationContext,
            abort                [1] BOOLEAN DEFAULT FALSE,
            common-arguments     [2] CommonArguments OPTIONAL },
         specification     [1] SEQUENCE {
            class                [0] OBJECT IDENTIFIER, -- id-oc-xxx
            scope                [1] Cardinal DEFAULT 0,
                  -- scope is contained objects in levels 0 through n
                  -- where 0 means the base object specified
                  -- by the object-identification
            selector             [2] Selector OPTIONAL,
                  -- should not be omitted if class is id-oc-document
            requested-attributes [3] SET OF AttributeId OPTIONAL,
            list-operator        [4] ListOperator
                                     DEFAULT get-attributes,
            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 opration.
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) */
```

```
      MODIFY_OP_REMOVE_ATTRIBUTE             /* (5) */
   };

   struct ModifyJobArgument {
      nint32                  sessionHandle;
      PrtContainedObjectId    jobIdentification;
      nuint32                 documentNumberOption;
      AttributeSet            jobAttrModificationSet;
      AttributeSet            docAttrModificationSet;
      NameOrOid              *modifyMessageOptionPtr;
      CommonArguments         commonArgumentsOption;
   };
```

6.4.1.2 ASN.1

```
   JobAttrModification ::= SEQUENCE {
      attribute-id   [0] AttributeId,
               -- Any job attributes, except:
           --   id-att-job-identifier,
           --   id-att-job-owner, id-att-job-originator,
           --   id-att-printer-name-requested,
           --   id-att-initial-value-job,
           --   any access-and-accounting attributes,
           --   any job-security attributes, and
           --   any job-status attributes.
           -- Any document attributes, except:
           --   id-att-transfer-method, id-att-document-content,
           --   id-att-initial-value-document, and
           --   any document-status attributes
      attribute-values  [1] SET OF ANY
                     -- DEFINED BY attribute-id -- OPTIONAL,
                     -- omitted for set-to-default
      modify-operator      [2] ModifyOperator DEFAULT replace }

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

   ModifyJobArgument ::= SEQUENCE {
      job-identification      [0] JobIdentifier,
      document-number         [1] PositiveInteger OPTIONAL,
                     -- required for addressing individual
                     -- documents in a multiple document print-job
      job-attr-modification   [2] SEQUENCE OF JobAttrModification,
         modify-message       [3] Message OPTIONAL,
            -- sets value of id-att-job-message-from-administrator
         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;
```

```
      AttributeSet            docAttrSet;
   };

   typedef ResubmitJob ResubmitJobSet<>;

   struct ResubmitJobsArgument {
      nint32                      sessionHandle;
      QualifiedName               destPrinterNameOption;
      NetAddress                 *destPrinterNetAddressPtr;
      ResubmitOpEnum              operation;
      ResubmitJobSet              resubmitJobSet;
      NameOrOid                  *resubmitMessageOptionPtr;
      CommonArguments             commonArgumentsOption;
   };
```

6.5.1.2 ASN.1

```
   ResubmitJobArgument ::= SEQUENCE {
      object-class          [0] OBJECT IDENTIFIER,
                             -- id-oc-job, id-oc-printer,
                             -- id-oc-server
      object-identification [1] ObjectIdentification,
      printer               [2] DistinguishedNameString,
      message               [3] Message OPTIONAL,
      common-arguments      [4] CommonArguments OPTIONAL }
```

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

```
   struct ResubmitJobResult {
      PrtContainedObjectId    oldJobIdentifier;
      PrtContainedObjectId    newJobIdentifier;
      AttributeSet            jobStatusOption;
   };

   typedef ResubmitJobResult ResubmitJobResultSet<>;

   struct ResubmitJobsResult {
      ResubmitJobResultSet    resubmitJobResultSet;
      ErrorReturn            *errorReturnOptionPtr;
   };
```

6.5.2.2 ASN.1

```
   ObjectStatus ::= SEQUENCE {
```

```
      object-status [0] SET OF Attribute OPTIONAL }
             -- job-identifier and new-job-identifier shall be
             -- returned at least.  For any jobs that could not
             -- be resubmitted, the new-job-identifier attribute
             -- shall be omitted as the only error indication.

   ResubmitJobResult ::= SEQUENCE {
      result-set    [0] SEQUENCE OF ObjectStatus }
              -- one result-set for each job resubmitted
              -- (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
Documenet Attributes
   Document Description Attributes
      document-format
      document-content
      transfer-method
   Document Production Instruction Attributes
      default-font
      sefault-medium
      number-up
      finishing
      sides
```

copy-count
reset-printer
Document Characteristics Attributes
fonts-used
media-used
Document Status Attributes
document-sequence-number
Operation Attributes
operation-locale
default-delivery-addresses
Printer Attributes
printer-name
printer-state
message
printer-initial-value-job
printer-initial-value-document
fonts-supported
fonts-ready
media-supported
media-ready
printer-associated-printers
document-formats-supported
numbers-up-supported
finishings-supported
sides-supported
job-sheets-supported
document-sheets-supported
maximum-copies-supported
notification-delivery-methods-supported
server-name
server-state
physical-printers-supported
logical-printers-supported
events-supported
transfer-methods-supported
locales-supported
multiple-documents-supported
cancel-individual-document-supported
modify-individual-document-supported
Initial Value Job Attributes
Initial Value Document Attributes

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

7.1 Job Attributes

A job object contains a set of job attributes and one or more document objects. The server shall create a printable job object in response to a client that invokes one or more Print abstract-operations. In addition, initial-value-job 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 job attributes (see the initial-value-job object class).

In addition to the attributes specifically defined for the job and 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

automatic	id-val-dn-syntax-automatic	server recognizes the syntax
X-500	id-val-dn-syntax-x-500	ISO/IEC 9594 Directory Service
XFN	id-val-dn-syntax-xfn	X/OPEN Federated Names
DCE	id-val-dn-syntax-dce	Distributed Computing Environment - includes X.500 and CDS
CDS	id-val-dn-syntax-cds	Cell Directory Service - part of DCE
NIS	id-val-dn-syntax-nis	Network Information Service
DNS	id-val-dn-syntax-dns	Domain Name Service
DEC-NS	id-val-dn-syntax-dec-ns	Digital Name Service
Internet-mail	id-val-dn-syntax-internet-mail	Internet Mail address
XNS	id-val-dn-syntax-xns	Xerox Network System
Bindery	id-val-dn-syntax-bindery	
NDS	id-val-dn-syntax-nds	Novell Directory Service
URL	id-val-dn-syntax-url	HTTP Universal Resource Locator
POSIX	id-val-dn-syntax-posix	POSIX file name (ISO/IEC 9945-1)
UNIX	id-val-dn-syntax-unix	UNIX(TM) file name
OS/2	id-val-dn-syntax-os2	OS/2 file name
PC-DOS	id-val-dn-syntax-pc-dos	PC DOS file name
NT	id-val-dn-syntax-nt	NT file name
MVS	id-val-dn-syntax-mvs	MVS file name
VM	id-val-dn-syntax-vm	VM file name
OS/400	id-val-dn-syntax-os400	OS/400 file name
VMS	id-val-dn-syntax-vms	VMS file name
UNC	id-val-dn-syntax-unc	Microsoft Universal Name Convention
NetWare	id-val-dn-syntax-netware	NetWare file path 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 consiststs 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, ther 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 printershall 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 extentions.

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 extentions) 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

Scott A. Isaacson
Novell, Inc.
122 E 1700 S
Provo, UT   84606

Phone: 801-861-7366
Fax: 801-861-4025
EMail: scott_isaacson@novell.com

F. Devon Taylor
Novell, Inc.
122 E 1700 S
Provo, UT   84606

Phone: 801-861-7179
Fax: 801-861-4025
EMail: devon_taylor@novell.com

Mike MacKay
Novell, Inc.
2180 Fortune Dr.
San Jose, CA   95131

Phone: 408-577-6368
Fax: 408-577-5151
EMail: mmackay@novell.com

Peter Zehler
Xerox Corporation
800 Phillips Rd. M/S 111-01X
Webster, NY 14580

Phone: 716-265-8755
Fax: 716-265-8792
EMail: peter_zehler@wb.xerox.com

Carl-Uno Manros
Xerox Corporation
701 S. Aviation Blvd.
El Segundo, CA   90245

Phone: 310-333-8273
Fax: 310-333-5514
EMail: cmanros@cp10.es.xerox.com

Tom Hastings
Xerox Corporation
701 S. Aviation Blvd.
El Segundo, CA   90245

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