Proposal S. Isaacson
Novell, Inc.
D. Taylor
Novell, Inc.
M. MacKay
Novell, Inc.
August 1996

LDPA Data Types Using XDR
Version 0.8, October 31, 1996

Status of this Memo

Draft Proposal

Abstract

This document is a full description of the LDPA protocol using XDR as defined by RFC 1831 and RFC 1832.

Table of Contents

1. ldpaop.x

```
/*
//*****************************************************************
//* Source module name: ldpaop.x                                  *
//*****************************************************************
*/
/* DESCRIPTION
**
**  This is an RPCL specification for LDPA operations
**  taken from ISO 10175 Document Printing Application (DPA)
**  (including part 3).
*/

#include "ldpaty.x"


/**************************************************
 **  Bind and BindPrinter Operation Parameters       **
 **************************************************/

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<>;
   };

struct BindArgument {
   Credentials       credentials;
   nint32            retrieveRestrictionsOption;
   opaque            bindSecurityOption<>;
   };

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

/*********************************************
 **          Unbind Operation Parameters     **
 *********************************************/

struct UnbindArgument {
   nint32            sessionHandle;
   };

struct UnbindResult {
   ErrorReturn      *errorReturnOptionPtr;
  };

/*********************************************
 **          Print Operation Parameters      **
 *********************************************/

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;
        };


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

     /**********************************************
      **          Modify Operation Parameters     **
      **********************************************/

     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;
        };

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

```
/*********************************************
 **          Cancel Operation Parameters     **
 *********************************************/

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

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


/*********************************************
 **          List Object Attributes          **
 *********************************************/

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;
       };

    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;
       };


    /**********************************************
     **       PromoteJob Operation Parameters    **
     **********************************************/

    struct PromoteJobArgument {
       nint32                  sessionHandle;
       PrtContainedObjectId    jobIdentifier;
       NameOrOid              *messageOptionPtr;
       CommonArguments         commonArgumentsOption;
       };

    struct PromoteJobResult {
       AttributeSet            jobStatusOption;
       ErrorReturn            *errorReturnOptionPtr;
       };
```

```
/*********************************************
 **       Pause Operation Parameters        **
 *********************************************/

enum PauseJobEnum  {
   PAUSE_JOB_ID,                              /* (0) */
   PAUSE_PRINTER_NM                           /* (1) */
   };

union PauseJob switch(PauseJobEnum designator) {
   case PAUSE_JOB_ID:
        PrtContainedObjectId     jobIdentifier;
   case PAUSE_PRINTER_NM:
        QualifiedName           printerName;
   };

struct PauseArgument {
   nint32                   sessionHandle;
   PauseJob                 pausedJob;
   NameOrOid               *pauseMessageOptionPtr;
   CommonArguments          commonArgumentsOption;
   };

struct PauseResult {
   PrtContainedObjectId pausedJobId;
   AttributeSet             jobStatusOption;
   ErrorReturn             *errorReturnOptionPtr;
  };

/*********************************************
 **       Resume Operation Parameters       **
 *********************************************/

struct ResumeArgument {
   nint32                   sessionHandle;
   PrtContainedObjectId     resumedJob;
   NameOrOid               *resumeMessageOptionPtr;
   CommonArguments          commonArgumentsOption;
   };

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



/************************************************
 **            ResubmitJobs parameters         **
 ************************************************/

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;
       };

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

     typedef ResubmitJobResult ResubmitJobResultSet<>;

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


     program PRINTPROGRAM {
       version PRINTVERS {

        BindResult
        BindPrinter ( BindPrinterArgument
        ) = 1;

        UnbindResult
        Unbind ( UnbindArgument
        ) = 2;

        PrintResult
        Print ( PrintArgument
        ) = 3;

        ModifyJobResult
        ModifyJob ( ModifyJobArgument
        ) = 4;

        CancelJobResult
        CancelJob ( CancelJobArgument
```

```
        ) = 5;

        ListObjectAttrsResult
        ListObjectAttributes ( ListObjectAttrsArgument
        ) = 6;

        PromoteJobResult
        PromoteJob ( PromoteJobArgument
        ) = 7;

        PauseResult
        Pause ( PauseArgument
        ) = 8;

        ResumeResult
        Resume (ResumeArgument
        ) = 9;

        ResubmitJobsResult
        ResubmitJobs ( ResubmitJobsArgument
        ) = 10;


      } = 1;
    } = 0x00000000;   /* the real program number */

    /* ************ end of LDPAOP.x *********************** */
```

2. ldpaty.x

```
   /*
   //**************************************************************
   //* Source module name: ldpaty.x                               *
   //**************************************************************
   */

   /*
    * =================== BASIC TYPES ===================
    */


   #include "ldpaav.x"



   /* ------ octet string ------ */

   typedef opaque OctetString<>;

   /* ------ string ------ */

   struct IntegerOption {
      nint32      flag;                  /* 0 or 1 */
      nint32      value;
      };


   /* ------ object identifier ------ */
   **   The generated code for the definition of the
   **   ObjectIdentifier in this specification is in the form:
   **   typedef struct {
   **           u_int ObjectIdentifier_len;
   **           byte *ObjectIdentifier_val;
   **   } ObjectIdentifier;
   **   the *ObjectIdentifier_val sequence generated at runtime, which
   **   conforms to the ASN.1 BER specification for the OBJECT
   **   IDENTIFER,
   **   consists a 0x06 byte (the BER tag), a length (assume 8-bit), and
   **   then the data bytes
   */

   typedef opaque ObjectIdentifier<>;


   /* ------ text ------ */

   /*
   // Note:   Text can be stored in XDR structures in Unicode,
   //         to eliminate problems in comparing disparate forms
   //         of text.  Unicode characters should always kept in
   //         low-high byte order.
   //
   //         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<>;


/* ------ name or OID ------ */
/* this definition is placed here to eliminate forward reference */
enum NameOrOidEnum {
   NAME_OR_OID_GLOBAL,                  /* (0) */
   NAME_OR_OID_LOCAL                    /* (1) */
   };

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

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

/*
// The QualifiedName union is intended to handle object names
//  with context
//  information.  Example:  NDS Tree Name
//  For now, provision is made for a distinguished name with no tree
name.
*/

enum QualifiedNameEnum {
   QUALIFIED_NAME_NONE,
   QUALIFIED_NAME_OTHER
   };

union QualifiedName switch (QualifiedNameEnum designator) {
   case QUALIFIED_NAME_NONE:
      void;
   case QUALIFIED_NAME_OTHER:
      Text      otherName;
   };

typedef QualifiedName QualifiedNameSet<>;

/* ------ distinguishedNameStringSequence ------ */
typedef DistinguishedNameString DistinguishedNameStringSeq<>;


/* ------ integer sequence ------ */
typedef nint32 IntegerSeq<>;

/* ------ cardinal sequence ------ */
```

```
typedef nuint32 CardinalSeq<>;

/* ------ integer range ------ */

struct IntegerRange {
   nint32      lowerBound;
   nint32      upperBound;
   };

/* ------ cardinal range ------ */

struct CardinalRange {
   nuint32     lowerBound;
   nuint32     upperBound;
};


/* ------ cardinal64 range ------ */
struct Cardinal64Range {
   nuint64      lowerBound;
   nuint64      upperBound;
   };


/* ------ object identifier sequence  ------ */
/* NOTE: SET OF and SEQUENCE OF semantics have to be enforced
** programmatically at runtime
*/
typedef ObjectIdentifier ObjectIdentifierSet < >;


/* ------ name or OID sequence ------ */
typedef NameOrOid NameOrOidSet<>;

/* ------ RDN sequence  ------ */
typedef Text RDNSequence<>;


/* ------ realization ------ */
enum RealizationEnum {
   REALIZATION_LOGICAL,
   REALIZATION_PHYSICAL,
   REALIZATION_LOG_AND_PHYS
   };

/* ------ medium dimensions ------- */
struct XYRealDimensions {
   nreal64     xDimension;
   nreal64     yDimension;
   };

/* ------ dimension -------- */

enum DimValueEnum {
   DIMENSION_NUMERIC,                    /* (0) */
   DIMENSION_NAMED                       /* (1) */
   };
```

```
   union DimValue switch(DimValueEnum designator) {
      case DIMENSION_NUMERIC:
         nreal64     numericValue;
      case DIMENSION_NAMED:
         NameOrOid  namedValue;
      };

   struct Dimension {
      DimValue       value;
      RealOption     toleranceOption;
      };

   /* ------ xy dimensions -------- */

   struct XYCardinalDimensions {
      nuint32     xDimension;
      nuint32     yDimension;
      };

   enum XYDimensionsValueEnum {
      DIM_XY_REAL,                            /* (0) */
      DIM_XY_NAMED,                           /* (1) */
      DIM_XY_CARDINAL                         /* (2) */
      };

   union XYDimensionsValue switch(XYDimensionsValueEnum designator) {
      case DIM_XY_REAL:
         XYRealDimensions     realValue;
      case DIM_XY_NAMED:
         NameOrOid            namedValue;
      case DIM_XY_CARDINAL:
         XYCardinalDimensions cardinalValue;
      };

   struct XYDimensions {
      XYDimensionsValue    value;
      RealOption           toleranceOption;
      };

   /* ------ locations --------- */

   enum LocationValueEnum {
      LOCATION_NUMERIC,                       /* (0) */
      LOCATION_NAMED                          /* (1) */
      };

   union LocationValue switch(LocationValueEnum designator) {
      case LOCATION_NUMERIC:
         RealSet     numericValueSeq;
      case LOCATION_NAMED:
         NameOrOid   namedValue;
      };

   struct Locations {
      LocationValue  value;
      RealOption     toleranceOption;
      };
```

```
/* ------ area ------ */
/*  4 reals, non-negative  */

struct Area {
   nreal64     minimumX;
   nreal64     maximumX;
   nreal64     minimumY;
   nreal64     maximumY;
   } ;

/* ------ area Sequence ------ */
typedef Area AreaSeq<>;

/* ------ edge ------ */

enum Edge {
   EDGE_BOTTOM,        /* (0) */
   EDGE_RIGHT,         /* (1) */
   EDGE_TOP,           /* (2) */
   EDGE_LEFT           /* (3) */
   } ;

struct EdgeOption  {
   nint32         flag;
   Edge           value;
   } ;

/* ------ cardinal or oid ------ */
enum CardinalOrOidEnum {
   CARDINAL_OR_OID_NUMBER,                /* (0) */
   CARDINAL_OR_OID_ID                     /* (1) */
   };

union CardinalOrOid switch (CardinalOrOidEnum designator) {
   case CARDINAL_OR_OID_NUMBER:
      nint32            cardinal;
   case CARDINAL_OR_OID_ID:
      ObjectIdentifier  oid;
   };

/* ------ oid cardinal map ------ */
struct OidCardinalMap {
   ObjectIdentifier  oidOption;
   nuint32           cardinal;
   };

/* ------ cardinal or name or oid ------ */
enum CardinalOrNameOrOidEnum {
   CARDINAL_OR_NAM_OR_OID_NUM,            /* (0) */
   CARDINAL_OR_NAME_OR_OID_ID             /* (1) */
   };

union CardinalOrNameOrOid switch (CardinalOrNameOrOidEnum designator)
{
   case CARDINAL_OR_NAM_OR_OID_NUM:
      nint32            cardinal;
   case CARDINAL_OR_NAME_OR_OID_ID:
```

```
          NameOrOid         nameOrOid;
       };

   /* ------ positiveIntegerOrOid type  ------ */

   enum PositiveIntegerOrOidEnum {
      INT_OR_OID_ID,                          /* (0) */
      INT_OR_OID_NUMBER                       /* (1) */
      };

   union PositiveIntegerOrOid switch (
            PositiveIntegerOrOidEnum designator) {
      case INT_OR_OID_ID:
         ObjectIdentifier  oid;
      case INT_OR_OID_NUMBER:
         nint32            integer;
      };

   /* ------ job identifier ------ */
   /* defined here to eliminate forward reference */

   struct PrtContainedObjectId {
       Text          printerName;
       nuint32       localIdentifier;
       };
   typedef PrtContainedObjectId PrtContainedObjectIdSet<>;

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

   enum DeliveryAddressEnum {
      DA_OR_ADDR_AND_OR_DIR_NAME,             /* (0) */
      DA_DISTINGUISHED_NAME,                  /* (1) */
      DA_TEXT,                                /* (2) */
      DA_OCTET_STRING,                        /* (3) */
      DA_DIST_NAME_STRING,                    /* (4) */
      DA_RPC_ADDRESS,                         /* (5) */
      DA_QUALIFIED_NAME                       /* (6) */
      };

   union DeliveryAddress switch(DeliveryAddressEnum designator) {
      case DA_OR_ADDR_AND_OR_DIR_NAME:
         Text     mhsAddress;
      case DA_DISTINGUISHED_NAME:
         Text     distinguishedName;
      case DA_TEXT:
         Text     text;
      case DA_OCTET_STRING:
         OctetString    octetString;
      case DA_DIST_NAME_STRING:
         DistinguishedNameString distinguishedNameString;
      case DA_RPC_ADDRESS:
         void;
      case DA_QUALIFIED_NAME:
```

```
         QualifiedName   qualifiedName;
      };

   /*
   // This structure supports the identification of printer
   // configuration objects. These objects describe job defaults
   // and limits.  Each name service printer object can have one
   // of each of these objects in the managed object database.
   // These objects are identified by Logical Printer
   // Name and by name service Printer Name.
   */

   struct PrtConfigObjectId {
      Text                  printerName;
      QualifiedName         qualifiedName;
      };


   enum eventTypeEnum {
       EVENT_TYPE_VALUE_CHANGE,    /* (0) */
       EVENT_TYPE_ERROR,           /* (1) */
       EVENT_TYPE_WARNING,         /* (2) */
       EVENT_TYPE_REPORT           /* (3) */
      };

   struct EventObjectId {
      Text                  name;
      ObjectIdentifier      containingClassOid;
      enum eventTypeEnum    eventType;
      };

   struct QualifiedNameMap {
      QualifiedName         primary;
      QualifiedName         secondary;
      };


   /* ------ privileges ------ */
   typedef opaque PrivAttrCertificate<>;

   struct Privileges {
      PrivAttrCertificate operationPacOption;
      PrivAttrCertificate proxyPacOption;
      };

   /* ------ default delivery addresses ------ */
   struct DeliveryAddressForMethod {
      ObjectIdentifier  method;
      DeliveryAddress   address;
      };

   /* ------ ObjectIdentification ------ */
   enum ObjectIdentificationEnum {
      OBJ_ID_NONE,                              /* (0) */
      OBJ_ID_PRT_CONTAIND_OBJ_ID,               /* (1) */
      OBJ_ID_DOCUMENT_IDENTIFIER,               /* (2) */
      OBJ_ID_OBJECT_IDENTIFIER,                 /* (3) */
```

```
        OBJ_ID_OBJECT_NAME,                     /* (4) */
        OBJ_ID_NAME_OR_OID,                     /* (5) */
        OBJ_ID_SIMPLE_NAME,                     /* (6) */
        OBJ_ID_PRT_CONFIG_OBJ_ID,               /* (7) */
        OBJ_ID_QUALIFIED_NAME,                  /* (8) */
        OBJ_ID_EVENT_OBJECT_ID                  /* (9) */
        };

    union ObjectIdentification switch(ObjectIdentificationEnum
    designator) {
        case OBJ_ID_NONE:
            void;
        case OBJ_ID_PRT_CONTAIND_OBJ_ID:
            PrtContainedObjectId    prtContainedObjectId;
        case OBJ_ID_DOCUMENT_IDENTIFIER:
            DocumentIdentifier      documentIdentifier;
        case OBJ_ID_OBJECT_IDENTIFIER:
            ObjectIdentifier        objectIdentifier;
        case OBJ_ID_OBJECT_NAME:
            DistinguishedNameString objectName;
        case OBJ_ID_NAME_OR_OID:
            NameOrOid               nameOrOid;
        case OBJ_ID_SIMPLE_NAME:
            Text                    simpleName;
        case OBJ_ID_PRT_CONFIG_OBJ_ID:
            PrtConfigObjectId       prtConfigObjectId;
        case OBJ_ID_QUALIFIED_NAME:
            QualifiedName           qualifiedName;
        case OBJ_ID_EVENT_OBJECT_ID:
            EventObjectId           eventObjectId;
        };

    typedef ObjectIdentification ObjectIdentificationSeq<>;


    struct TypedName {
        Text            name;
        nint32          level;
        nint32          interval;
        };

    enum NetAddressTypeEnum {
        NET_IPX,
        NET_IP,
        NET_SDLC,
        NET_TOKENRING_ETHERNET,
        NET_OSI,
        NET_APPLETALK,
        NET_COUNT
        };

    struct NetAddress {
        NetAddressTypeEnum   type;
        OctetString          address;
        };

    struct NameOrOidDimensionMap {
```

```
         NameOrOid            nameOrOid;
         XYCardinalDimensions value;
         };

      enum StateSeverityEnum {
         STATE_SEVERITY_OTHER = 1,               /* 1 */
         STATE_SEVERITY_WARNING,                 /* 2 */
         STATE_SEVERITY_CRITICAL                 /* 3 */
         };

      enum TrainingEnum {
         TRAINING_OTHER = 1,                     /* 1 */
         TRAINING_UNKNOWN,                       /* 2 */
         TRAINING_UNTRAINED,                     /* 3 */
         TRAINING_TRAINED,                       /* 4 */
         TRAINING_FIELD_SERVICE,                 /* 5 */
         TRAINING_MANAGEMENT                     /* 6 */
         };

      struct PrinterStateReason {
         NameOrOid            identifier;
         StateSeverityEnum    severity;
         TrainingEnum         trainingLevel;
         ObjectIdentifier     objectClassOid;
         ObjectIdentification objectIdentification;
         nuint32              time;
         Text                 messageOption;
         };

      enum PersistenceEnum {
         PERSISTENCE_PERMANENT,                  /*  0 */
         PERSISTENCE_VOLATILE                    /*  1 */
         };

      enum AddressItemTypeEnum  {
         ADDR_ITEM_USER,                 /* 0  */
         ADDR_ITEM_SERVER,               /* 1  */
         ADDR_ITEM_VOLUME,               /* 2  */
         ADDR_ITEM_ORG_UNIT,             /* 3  */
         ADDR_ITEM_ORG,                  /* 4  */
         ADDR_ITEM_GROUP,                /* 5  */
         ADDR_ITEM_DN,                   /* 6  */
         ADDR_ITEM_USR_OR_CONTAINER,     /* 7  */
         ADDR_ITEM_CASE_EXACT_STR,       /* 8  */
         ADDR_ITEM_CASE_IGNORE_STR,      /* 9  */
         ADDR_ITEM_NUMERIC_STR,          /* 10 */
         ADDR_ITEM_DOS_FILENAME,         /* 11 */
         ADDR_ITEM_PHONE_NUMBER,         /* 12 */
         ADDR_ITEM_BOOLEAN,              /* 13 */
         ADDR_ITEM_INTEGER,              /* 14 */
         ADDR_ITEM_NET_ADDRESS,          /* 15 */
         ADDR_ITEM_OCTET_STRING          /* 16 */
         };

      union AddressItem switch(AddressItemTypeEnum designator) {
         case ADDR_ITEM_USER:
            QualifiedName     userName;
```

```
         case ADDR_ITEM_SERVER:
            QualifiedName     serverName;
         case ADDR_ITEM_VOLUME:
            QualifiedName     volumeName;
         case ADDR_ITEM_ORG_UNIT:
            QualifiedName     orgUnitName;
         case ADDR_ITEM_ORG:
            QualifiedName     orgName;
         case ADDR_ITEM_GROUP:
            QualifiedName     groupName;
         case ADDR_ITEM_DN:
            QualifiedName     dN;
         case ADDR_ITEM_USR_OR_CONTAINER:
            QualifiedName     userOrContainerName;
         case ADDR_ITEM_CASE_EXACT_STR:
            Text              ceStr;
         case ADDR_ITEM_CASE_IGNORE_STR:
            Text              ciStr;
         case ADDR_ITEM_NUMERIC_STR:
            Text              numStr;
         case ADDR_ITEM_DOS_FILENAME:
            Text              filename;
         case ADDR_ITEM_PHONE_NUMBER:
            Text              phoneNumber;
         case ADDR_ITEM_BOOLEAN:
            bool              bFlag;
         case ADDR_ITEM_INTEGER:
            nint32            integer;
         case ADDR_ITEM_NET_ADDRESS:
            NetAddress        netAddress;
         case ADDR_ITEM_OCTET_STRING:
            OctetString       octetString;
         };

      typedef  AddressItem  NotifyDeliveryAddr<>;

      enum EventObjectOperation {
         EVENT_OBJ_OP_NONE,
         EVENT_OBJ_OP_ADD_FLAG,
         EVENT_OBJ_OP_DELETE_FLAG,
         EVENT_OBJ_OP_DELETE_OBJECT
         };

      enum EventObjectTypeEnum {
         EVENT_OBJ_TYPE_OBJECT,
         EVENT_OBJ_TYPE_FILTER,
         EVENT_OBJ_TYPE_DETAIL
         };

      union EventObjectItem switch (EventObjectTypeEnum designator) {
         case EVENT_OBJ_TYPE_OBJECT:
            void;
         case EVENT_OBJ_TYPE_FILTER:
            ObjectIdentifier     filterClassOid;
         case EVENT_OBJ_TYPE_DETAIL:
            ObjectIdentifierSet  eventOidSet;
         };
```

```
struct EventObject {
   nuint32                 eventType;
   ObjectIdentifier        containingClassOid;
   ObjectIdentification    containingObjectId;
   EventObjectOperation    opCode;
   EventObjectItem         item;
   };

typedef EventObject EventObjectSet<>;

struct EventHandlingProfile {
   nuint32                 profileId;
   PersistenceEnum         persistence;
   QualifiedName           consumerName;
   OctetString             supplierId;
   nint32                  languageId;
   NameOrOid               methodId;
   NotifyDeliveryAddr      deliveryAddress;
   EventObjectSet          eventObjectSet;
   QualifiedName           accountOption;
   };

struct ModifyEventProfileArgument {
   nuint32                 sessionHandle;
   nuint32                 profileId;
   bool                    supplierFlag;
   OctetString             supplierIdOption;
   bool                    languageFlag;
   nint32                  languageId;             /* Optional */
   bool                    methodFlag;
   NameOrOid              *methodIdOptionPtr;      /* Optional */
   bool                    deliveryAddrFlag;
   NotifyDeliveryAddr     *deliveryAddrOptionPtr;  /* Optional */
   EventObjectSet          eventObjectSet;         /* Optional */
   };

enum ListProfilesChoiceEnum {
   LIST_CHOICE_ID,                            /* (0) */
   LIST_CHOICE_FILTER                         /* (1) */
   };

enum ListProfilesResultEnum {
   LIST_RESULT_COMPLETE,
   LIST_RESULT_NO_EVENT_OBJS,
   LIST_RESULT_PROFILE_IDS
   };

struct ListProfilesFilter {
   QualifiedName           consumerNameOption;  /* Optional */
   NameOrOid              *methodIdOptionPtr;   /* Optional */
   IntegerOption           languageId;          /* Optional */
   };

union ListProfilesChoice switch(ListProfilesChoiceEnum designator) {
   case LIST_CHOICE_ID:
      CardinalSeq          profileIdSeq;
   case LIST_CHOICE_FILTER:
```

```
         ListProfilesFilter   filter;
      };

   enum ListProfilesTypeEnum {
      LIST_EVENT_PROFILES_SPEC,              /* (0) */
      LIST_EVENT_PROFILES_CONTINUE           /* (1) */
      };

   struct ListEventProfilesSpec {
      QualifiedName            supplierAlias;
      ListProfilesChoice       choice;
      ListProfilesResultEnum   detail;
      IntegerOption            countOption;
      };

   struct ListEventProfilesCont {
      OctetString              context;
      bool                     abort;
      };

   union ListProfilesType switch(ListProfilesTypeEnum designator) {
      case LIST_EVENT_PROFILES_SPEC:
         ListEventProfilesSpec   specification;
      case LIST_EVENT_PROFILES_CONTINUE:
         ListEventProfilesCont   continuation;
      };

   struct ListEventProfilesArgument {
      nint32                      sessionHandle;
      ListProfilesType            listProfilesType;
      };

   typedef EventHandlingProfile ProfileResultSet<>;


   /* -------- results profile ------- */

   struct ResultsProfile {
      ObjectIdentifier      deliveryMethodOption;
      NameOrOid            *resultsSetCommentOptionPtr;
      DeliveryAddress      *deliveryAddressOptionPtr;
      nint32                jobCopies;
      NameOrOid            *outputBinOptionPtr;
      };


   union CriterionThreshold switch(AVTEnum designator) {
      case AVT_DELTA_TIME:
         nint32   deltaTime;
      case AVT_TIME:
         nuint32  time;
      case AVT_INTEGER:
         nint32   integer;
      case AVT_CARDINAL:
         nuint32   cardinal;
      case AVT_POSITIVE_INTEGER:
         nint32   positiveInteger;
```

```
        case AVT_PERCENT:
           nint32   percent;
        };

     struct Criteria {
        ObjectIdentifier     type;             /* AttributeId */
        CriterionThreshold   threshold;
        };


     /* ------ job security attributes ------ */

     struct JobLevel {
        ObjectIdentifier     organizationId;
        nuint32              level;
        };

     struct JobCategories {
        ObjectIdentifier     organizationId;
        CardinalSeq          categories;
        };


     /* ------ job ignored attributes ------ */
     /* NOTE:  Should really be :  AttributeValueSet values;
     **  We can not return values becuase of the
     **  recursive definition!
     */

     union IgnoredAttributeValue switch(AVTEnum designator) {
        case AVT_NULL:
           void ;                  /* Pass nothing */
        case AVT_OBJECT_IDENTIFIER:
           ObjectIdentifier     identifier;
        case AVT_NAME_OR_OID:
           NameOrOid nameOrOid;
        };

     typedef IgnoredAttributeValue IgnoredAttributeValueSet<>;

     struct IgnoredAttribute {
        nuint32              documentNumberOption;
        ObjectIdentifier     attributeId;
        IgnoredAttributeValueSet    values;
        };

     /* ------ default resources  ------ */
     enum ResourceEnum
     {
        RESOURCE_NAME_OR_OID,                  /* (0) */
        RESOURCE_TEXT                          /* (1) */
     };

     union Resource switch(ResourceEnum designator)
     {
        case RESOURCE_NAME_OR_OID:
           NameOrOid  nameOrOid;
```

```
      case RESOURCE_TEXT:
         Text       name;
   };

   /* ------ medium substitution  ------ */

   struct MediumSubstitution {
      NameOrOid  original;
      NameOrOid  substitution;
      };

   /* ------ font substitution  ------ */

   struct FontSubstitution {
      Text             original;
      Text             substitution;
      };

   /* ------ resource context  ------ */
   enum ResourceContextEnum
   {
      RESOURCE_CONTXT_NAM_OR_OID,            /* (0) */
      RESOURCE_CONTEXT_TEXT                  /* (1) */
   };

   union ResourceContext switch(ResourceContextEnum designator)
   {
      case RESOURCE_CONTXT_NAM_OR_OID:
         NameOrOid  objectName;
      case RESOURCE_CONTEXT_TEXT:
         Text       pathName;
   };

   typedef ResourceContext ResourceContextSeq<>;

   /* ------ page select  ------ */

   enum PageIdentifierEnum  {
      PAGE_ID_NOMINAL_NUMBER,                /* (0) */
      PAGE_ID_ALPHANUMERIC,                  /* (1) */
      PAGE_ID_TAG                            /* (2) */
      };

   union PageIdentifier switch(PageIdentifierEnum designator) {
      case PAGE_ID_NOMINAL_NUMBER:
         nint32   nominalPageNumber;
      case PAGE_ID_ALPHANUMERIC:
         Text    alphanumericPageNumber;
      case PAGE_ID_TAG:
         NameOrOid pageTag;
      };

   struct PageIdentifierOption  {
      nint32           flag;
      PageIdentifier   value;
      };
```

```
struct PageSelect {
   PageIdentifierOption beginningPage;
   PageIdentifierOption endingPage;
   };

typedef PageSelect PageSelectSequence<>;

/* ------ document format ------ */
/*
// Use prtInterpeterLangFamily from the IETF Printer MIB
*/

/* ------ document content  ------ */

enum DocumentContentEnum  {
   DOC_CONTENT_INCLUDED,           /* (0) */
   DOC_CONTENT_REFERENCED          /* (1) */
   };

union DocumentContent switch(DocumentContentEnum designator) {
   case DOC_CONTENT_INCLUDED:
     OctetString        includedDocument;    /* External */
   case DOC_CONTENT_REFERENCED:
      DistinguishedNameString referencedDocument;
   };

/* ------- intended page size --------- */

enum PageSizeEnum {
   PAGE_SIZE_ID,                        /* (0) */
   PAGE_SIZE_DIMENSIONS                 /* (1) */
   };

union PageSize switch(PageSizeEnum designator) {
   case PAGE_SIZE_ID:
      ObjectIdentifier  pageSizeId;
   case PAGE_SIZE_DIMENSIONS:
      XYRealDimensions pageDimensions;
   };

/* ------ presentation direction ------ */

enum PresentationDirectionEnum {
   DIR_TO_RIGHT_TO_BOTTOM,                /* (0) */
   DIR_TO_LEFT_TO_BOTTOM,                 /* (1) */
   DIR_BIDIRECT_TO_BOTTOM,                /* (2) */
   DIR_TO_RIGHT_TO_TOP,                   /* (3) */
   DIR_TO_LEFT_TO_TOP,                    /* (4) */
   DIR_BIDIRECT_TO_TOP,                   /* (5) */
   DIR_TO_BOTTOM_TO_RIGHT,                /* (6) */
   DIR_TO_BOTTOM_TO_LEFT,                 /* (7) */
   DIR_TO_TOP_TO_LEFT,                    /* (8) */
   DIR_TO_TOP_TO_RIGHT                    /* (9) */
   };

/* ------ page order received  ------ */
```

```
   enum PageOrderTypeEnum {
      PAGE_ORDER_UNKNOWN,                    /* (0) */
      PAGE_ORDER_FIRST_TO_LAST,              /* (1) */
      PAGE_ORDER_LAST_TO_FIRST               /* (2) */
      };


   /* ------ page select supported ------ */

   enum PageIdTypeEnum {
      PAGE_ID_TYPE_NUMERIC,                  /* (0) */
      PAGE_ID_TYPE_ALPHANUMERIC,             /* (1) */
      PAGE_ID_TYPE_TAG                       /* (2) */
      };

   /* ------ page media select ------ */
   enum MediaSelectEnum {
      MEDIA_SELECT_ALL_PAGES,              /* (0) */
      MEDIA_SELECT_SELECTD_PAGES          /* (1) */
      };

   struct MediaSelect {
      PageSelect    pageRange;
      NameOrOid     mediumId;
      };

   typedef MediaSelect MediaSelectSeq<>;

   union PageMediaSelect switch (MediaSelectEnum designator) {
      case MEDIA_SELECT_ALL_PAGES:
         NameOrOid   mediumForAllPages;
      case MEDIA_SELECT_SELECTD_PAGES:
         MediaSelectSeq mediumForSelectedPagesSeq;
      };

   /* ------ medium source size  ------ */
   enum MediumSizeEnum {
      MEDIUM_SIZE_DISCRETE,                  /* (0) */
      MEDIUM_SIZE_CONTINUOUS                 /* (1) */
      };

   struct MediumDiscreteSizes {
      PageSize          pageSize;
      bool              longEdgeFeeds;             /* default true */
      Area              assuredReproductionArea;
      };

   struct MediumContinuousSizes {
      RealRange         sizeRangeAcrossFeedDirection;
      nreal64           sizeIncrementAcrossFeedDir;
      RealRange         sizeRangeInFeedDirection;
      nreal64           sizeIncrementInFeedDir;
      bool              longEdgeFeeds;             /* default true */
      Area              GenericAssuredReproductionArea;
      };

   union MediumSize switch (MediumSizeEnum designator) {
```

```
         case MEDIUM_SIZE_DISCRETE:
            MediumDiscreteSizes     discreteSizes;
         case MEDIUM_SIZE_CONTINUOUS:
            MediumContinuousSizes   continuousSizes;
         };

      struct MediumSourceSize {
         NameOrOid     *inputTrayOptionPtr;
         MediumSize     mediumSize;
         };

      /* ------ input trays medium  ------ */
      struct InputTrayMedium {
         NameOrOid      inputTray;
         NameOrOid      medium;
         };

      /* ------ output bins characteristics ------ */
      enum PageOrientationEnum {
         PAGE_FACE_UNKNOWN,                        /* (0) */
         PAGE_FACE_UP,                             /* (1) */
         PAGE_FACE_DOWN                            /* (2) */
         };

      struct OutputBinChar {
         PageOrderTypeEnum    stackingOrder;
         PageOrientationEnum  orientation;
         };

      typedef OutputBinChar OutBinsCharacteristics<>;

      /* ------ printer confidentiality level range ------ */

      struct LevelRange {
         ObjectIdentifier  organizationId;
         nint32            minimum;
         nint32            maximum;
         };

      /* ------ printer confidentiality category sets ------ */

      struct CategorySet {
         ObjectIdentifier  organizationId;
         CardinalSeq       categoryMinimum;
         CardinalSeq       categoryMaximum;
         };

      /* numbers up supported ------ */
      enum NumbersUpEnum {
         NUMBERS_UP_CARDINAL,
         NUMBERS_UP_NAME_OR_OID,
         NUMBERS_UP_CARDINAL_RANGE
         };

      union NumbersUpSupported switch(NumbersUpEnum designator) {
         case NUMBERS_UP_CARDINAL:
            nuint32        cardinal;
```

```
      case NUMBERS_UP_NAME_OR_OID:
         NameOrOid      nameOrOid;
      case NUMBERS_UP_CARDINAL_RANGE:
         CardinalRange  cardinalRange;
      };


   /* -------------- finishing  ---------------- */

   struct CommonParameters {
      XYDimensions *referenceSizeOptionPtr;
      EdgeOption    referenceEdgeOption;
      EdgeOption    jogEdgeOption;
      };

   enum StitchingEnum {
       STITCHING_NAMED,                     /* (0) */
       STITCHING_PARAMETERS                 /* (1) */
       };

   struct StitchParameters {
      CommonParameters  common;
      Dimension        *processOffsetOptionPtr;
      Locations        *headLocationsOptionPtr;
      NameOrOid         stitchType;
      };

   union StitchingSpec switch (StitchingEnum designator) {
      case STITCHING_NAMED:
         NameOrOid namedStitching;
      case STITCHING_PARAMETERS:
         StitchParameters parameters;
      };

   enum BindingEnum {
      BINDING_NAMED,                        /* (0) */
      BINDING_PARAMETERS                    /* (1) */
      };

   struct BindParameters {
      CommonParameters  common;
      NameOrOid         bindType;
      NameOrOid         bindColor;
      };

   union BindingSpec switch (BindingEnum designator) {
      case BINDING_NAMED:
         NameOrOid        namedBinding;
      case BINDING_PARAMETERS:
         BindParameters   parameters;
      };

   enum TrimmingEnum {
      TRIMMING_NAMED,                       /* (0) */
      TRIMMING_PARAMETERS                   /* (1) */
      };
```

```
   struct TrimParameters {
      CommonParameters common;
      Dimension        trimOffset;
      XYDimensions     trimDimensions;
      };

   union TrimmingSpec switch (TrimmingEnum designator) {
      case TRIMMING_NAMED:
         NameOrOid        namedTrimming;
      case TRIMMING_PARAMETERS:
         TrimParameters   parameters;
      };

   enum DieCuttingEnum {
      DIE_CUTTING_NAMED,                        /* (0) */
      DIE_CUTTING_PARAMETERS                    /* (1) */
      };

   struct DieCuttingParameters {
      CommonParameters common;
      XYDimensions     dieCutLocation;
      NameOrOid        dieCutName;
      };

   union DieCuttingSpec switch (DieCuttingEnum designator) {
      case DIE_CUTTING_NAMED:
         NameOrOid              namedDieCutting;
      case DIE_CUTTING_PARAMETERS:
         DieCuttingParameters   parameters;
      };

   enum PunchingEnum {
      PUNCHING_NAMED,                           /* (0) */
      PUNCHING_PARAMETERS                       /* (1) */
      };

   struct PunchParameters {
      CommonParameters    common;
      Dimension          *processOffsetOptionPtr; /* dimension not area
   */
      Locations           headLocations;
      Dimension          *punchDiameterPtr;
      };

   union PunchingSpec switch (PunchingEnum designator) {
      case PUNCHING_NAMED:
         NameOrOid namedPunching;
      case PUNCHING_PARAMETERS:
         PunchParameters parameters;
      };

   enum PerforatingEnum {
      PERFORATING_NAMED,                        /* (0) */
      PERFORATING_PARAMETERS                    /* (1) */
      };

   struct PerfParameters {
```

```
         CommonParameters common;
         Locations        headLocations;
         NameOrOid        perfType;
         };

    union PerforatingSpec switch (PerforatingEnum designator) {
       case PERFORATING_NAMED:
          NameOrOid        namedPerforating;
       case PERFORATING_PARAMETERS:
          PerfParameters   parameters;
       };

    enum SlittingEnum {
       SLITTING_NAMED,                       /* (0) */
       SLITTING_PARAMETERS                   /* (1) */
       };

    struct SlitParameters {
       CommonParameters common;
       Locations        headLocations;
       };

    union SlittingSpec switch (SlittingEnum designator) {
       case SLITTING_NAMED:
          NameOrOid namedSlitting;
       case SLITTING_PARAMETERS:
          SlitParameters parameters;
       };

    enum InsertIdEnum {
       INSERT_NAME,                           /* (0) */
       INSERT_BIN                              /* (1) */
       };

    union InsertId switch (InsertIdEnum designator) {
       case INSERT_NAME:
          NameOrOid   insertName;
       case INSERT_BIN:
          nuint32     insertBin;
       };

    enum InsertTopSurfaceEnum {
       INSERT_TOP_SURFACE_TOP,               /* (0) */
       INSERT_TOP_SURFACE_BOTTOM             /* (1) */
       };

    struct InsertSheet {
       InsertId          insertId;
       XYDimensions     *insertSizeOptionPtr;
       EdgeOption        insertEdgeOption;
       InsertTopSurfaceEnum insertTopSurface;
       nuint32           insertAfter;
       NameOrOid        *insertMessageOptionPtr;
       };

    typedef InsertSheet InsertSheetSeq<>;
```

```
   struct InsertParameters {
      EdgeOption     referenceEdgeOption;
      EdgeOption     jogEdgeOption;
      InsertSheetSeq insertSheetList;
      };

   enum InsertSpecEnum {
      INSERT_SPEC_NAMED,                    /* (0) */
      INSERT_SPEC_PARAMETERS                /* (1) */
      };

   union InsertSpec switch (InsertSpecEnum designator) {
      case INSERT_SPEC_NAMED:
         NameOrOid         namedInserting;
      case INSERT_SPEC_PARAMETERS:
         InsertParameters  parameters;
      };

   enum CoverSpecEnum {
      COVER_SPEC_NAMED,                     /* (0) */
      COVER_SPEC_PARAMETERS                 /* (1) */
      };

   struct CoverParameters {
      CommonParameters common;
      NameOrOid   *frontCoverOptionPtr;
      NameOrOid   *backCoverOptionPtr;
      };

   union CoverSpec switch (CoverSpecEnum designator) {
      case COVER_SPEC_NAMED:
         NameOrOid        namedCovers;
      case COVER_SPEC_PARAMETERS:
         CoverParameters  parameters;
      };

   enum FoldingSpecEnum {
      FOLDING_SPEC_NAMED,                  /* (0) */
      FOLDING_SPEC_PARAMETERS              /* (1) */
      };

   struct FoldingParameters {
      CommonParameters common;
      Locations        headLocations;
      };

   union FoldingSpec switch (FoldingSpecEnum designator) {
      case FOLDING_SPEC_NAMED:
         NameOrOid           namedFolding;
      case FOLDING_SPEC_PARAMETERS:
         FoldingParameters   parameters;
      };

   struct OtherParameters {
      CommonParameters  common;
      Dimension        *processOffsetOptionPtr;
      Locations        *headLocationsOptionPtr;
```

```
      opaque            otherParams < >;
      };

   struct OtherFinishingSpec {
      NameOrOid        finishingOpType;
      NameOrOid       *namedSpecOptionPtr;
      OtherParameters  parameters;
      };


   enum FinishingProcEnum {
      FINISHING_PROC_STITCHING,            /* (0) */
      FINISHING_PROC_BINDING,              /* (1) */
      FINISHING_PROC_TRIMMING,             /* (2) */
      FINISHING_PROC_DIE_CUTTING,          /* (3) */
      FINISHING_PROC_PUNCHING,             /* (4) */
      FINISHING_PROC_PERFORATING,          /* (5) */
      FINISHING_PROC_SLITTING,             /* (6) */
      FINISHING_PROC_INSERT,               /* (7) */
      FINISHING_PROC_COVER,                /* (8) */
      FINISHING_PROC_FOLDING,              /* (9) */
      FINISHING_PROC_OTHER                 /* (10) */
      };

   union FinishingProcSpec switch (FinishingProcEnum designator) {
      case FINISHING_PROC_STITCHING:
         StitchingSpec       stitchingSpec;
      case FINISHING_PROC_BINDING:
         BindingSpec         bindingSpec;
      case FINISHING_PROC_TRIMMING:
         TrimmingSpec        trimmingSpec;
      case FINISHING_PROC_DIE_CUTTING:
         DieCuttingSpec      dieCuttingSpec;
      case FINISHING_PROC_PUNCHING:
         PunchingSpec        punchingSpec;
      case FINISHING_PROC_PERFORATING:
         PerforatingSpec     perforatingSpec;
      case FINISHING_PROC_SLITTING:
         SlittingSpec        slittingSpec;
      case FINISHING_PROC_INSERT:
         InsertSpec          insertSpec;
      case FINISHING_PROC_COVER:
         CoverSpec           coverSpec;
      case FINISHING_PROC_FOLDING:
         FoldingSpec         foldingSpec;
      case FINISHING_PROC_OTHER:
         OtherFinishingSpec  otherFinishingSpec;
      };

   typedef FinishingProcSpec FinishingProcSpecSeq<>;

   enum FinishingSpecEnum {
      FINISHING_SPEC_NAMED,                  /* (0) */
      FINISHING_SPEC_LIST                    /* (1) */
      };

   union FinishingSpec switch (FinishingSpecEnum designator) {
```

```
        case FINISHING_SPEC_NAMED:
           NameOrOid         named;
        case FINISHING_SPEC_LIST:
           FinishingProcSpecSeq finishingSpecList;
        };

   struct Finishing {
      NameOrOid        *messageOptionPtr;
      FinishingSpec     spec;
      };


   /* ------ imposition ------- */

   enum ImpositionAttrEnum {
      IMP_ATTR_PLEX,
      IMP_ATTR_NUMBER_UP,
      IMP_ATTR_X_IMAGE_SHIFT,
      IMP_ATTR_Y_IMAGE_SHIFT,
      IMP_ATTR_LOG_PAGE_ORIGIN,
      IMP_ATTR_LOG_PAGE_X_OFFSET,
      IMP_ATTR_LOG_PAGE_Y_OFFSET,
      IMP_ATTR_LOG_PAGE_SCALING
      };

   union ImpositionAttr switch (ImpositionAttrEnum designator) {
      case IMP_ATTR_PLEX:
         opaque      plex<>;
      case IMP_ATTR_NUMBER_UP:
         nint32   numberUp;
      case IMP_ATTR_X_IMAGE_SHIFT:
         nreal64  xImageShift;
      case IMP_ATTR_Y_IMAGE_SHIFT:
         nreal64  yImageShift;
      case IMP_ATTR_LOG_PAGE_ORIGIN:
         opaque      logPageOrigin<>;
      case IMP_ATTR_LOG_PAGE_X_OFFSET:
         nreal64  logPageXOffset;
      case IMP_ATTR_LOG_PAGE_Y_OFFSET:
         nreal64  logPageYOffset;
      case IMP_ATTR_LOG_PAGE_SCALING:
         nreal64  logPageScaling;
      };

   enum ImpositionEnum {
      IMPOSITION_OBJECT,                     /* (0) */
      IMPOSITION_ATTR                        /* (1) */
      };

   union Imposition switch (ImpositionEnum designator) {
      case IMPOSITION_OBJECT:
         NameOrOid  impositionObject;
      case IMPOSITION_ATTR:
         ImpositionAttr    impositionAttr;
      };
```

```
/*
** NOTE:
**  In the following discriminant union, all values fit in 16 bytes
**  or are pointed to, whether they are conformant or fixed length.
**  Thus the storage for all attribute-values is 20 bytes, including
the
**  designator.
*/

union AttributeValue switch(AVTEnum designator) {
   case AVT_NULL:
      void ;
   case AVT_TEXT:
      Text                  text;
   case AVT_DESCRIPTIVE_NAME:
      Text                  descriptiveName;
   case AVT_DESCRIPTOR:
      Text                  descriptor;
   case AVT_MESSAGE:
      NameOrOid             message;
   case AVT_ERROR_MESSAGE:
      NameOrOid             errorMessage;
   case AVT_SIMPLE_NAME:
      Text                  simpleName;
   case AVT_DISTINGUISHED_NAME_STR:
      DistinguishedNameString distinguishedNameString;
   case AVT_DIST_NAME_STR_SEQ:
      DistinguishedNameStringSeq distinguishedNameStringSeq;
   case AVT_DELTA_TIME:
      nint32                deltaTime;
   case AVT_TIME:
      nuint32               time;
   case AVT_INTEGER:
      nint32                integer;
   case AVT_INTEGER_SEQ:
      IntegerSeq            integerSeq;
   case AVT_CARDINAL:
      nuint32               cardinal;
   case AVT_CARDINAL_SEQ:
      CardinalSeq           cardinalSeq;
   case AVT_POSITIVE_INTEGER:
      nint32                positiveInteger;
   case AVT_INTEGER_RANGE:
      IntegerRange          integerRange;
   case AVT_CARDINAL_RANGE:
      CardinalRange         cardinalRange;
   case AVT_MAXIMUM_INTEGER:
      nint32                maximumInteger;
   case AVT_MINIMUM_INTEGER:
      nint32                minimumInteger;
   case AVT_INTEGER_64:
      void;
   case AVT_INTEGER_64_SEQ:
      void;
   case AVT_CARDINAL_64:
      nuint64               cardinal64;
```

```
        case AVT_CARDINAL_64_SEQ:
           void;
        case AVT_POSITIVE_INTEGER_64:
           void;
        case AVT_INTEGER_64_RANGE:
           void;
        case AVT_CARDINAL_64_RANGE:
           void;
        case AVT_MAXIMUM_INTEGER_64:
           void;
        case AVT_MINIMUM_INTEGER_64:
           void;
        case AVT_REAL:
           nreal64              real;
        case AVT_REAL_SEQ:
           RealSet              realSeq;
        case AVT_NON_NEGATIVE_REAL:
           nreal64              nonNegativeReal;
        case AVT_REAL_RANGE:
           RealRange            realRange;
        case AVT_NON_NEGATIV_REAL_RANGE:
           RealRange            nonNegativeRealRange;
        case AVT_BOOLEAN:
           bool                 boolean;
        case AVT_PERCENT:
           nint32               percent;
        case AVT_OBJECT_IDENTIFIER:
           ObjectIdentifier     identifier;
        case AVT_OBJECT_IDENTIFIER_SEQ:
           ObjectIdentifierSet  objectIdentifierSeq;
        case AVT_NAME_OR_OID:
           NameOrOid            nameOrOid;
        case AVT_NAME_OR_OID_SEQ:
           NameOrOidSet         nameOrOidSeq;
        case AVT_DISTINGUISHED_NAME:
           Text                 distinguishedName;
        case AVT_RDN_SEQUENCE:
           RDNSequence          rdnSequence;
        case AVT_REALIZATION:
           RealizationEnum      realization;
        case AVT_MEDIUM_DIMENSIONS:
           XYRealDimensions     mediumDimensions;
        case AVT_DIMENSION:
           Dimension           *dimensionPtr;
        case AVT_XY_DIMENSIONS:
           XYDimensions        *xyDimensionsPtr;
        case AVT_LOCATIONS:
           Locations           *locationsPtr;
        case AVT_AREA:
           Area                *areaPtr;
        case AVT_AREA_SEQ:
           AreaSeq              areaSeq;
        case AVT_EDGE:
           Edge                 edge;
        case AVT_FONT_REFERENCE:
           Text                 fontReference;
        case AVT_CARDINAL_OR_OID:
```

```
        CardinalOrOid        cardinalOrOid;
     case AVT_OID_CARDINAL_MAP:
        OidCardinalMap       oidCardinalMap;
     case AVT_CARDINAL_OR_NAM_OR_OID:
        CardinalOrNameOrOid  cardinalOrNameOrOid;
     case AVT_POSITIVE_INT_OR_OID:
        PositiveIntegerOrOid positiveIntegerOrOid;
     case AVT_EVENT_HANDLING_PROFILE:
        EventHandlingProfile *eventHandlingProfilePtr;
     case AVT_OCTET_STRING:
        OctetString          octetString;
     case AVT_PRIORITY:
        nint32               jobPriority;
     case AVT_LOCALE:
        Text                 locale;
     case AVT_METHOD_DELIVERY_ADDR:
        DeliveryAddressForMethod *deliveryMethodAddressPtr;
     case AVT_OBJECT_IDENTIFICATION:
        ObjectIdentification *objectIdentificationPtr;
     case AVT_RESULTS_PROFILE:
        ResultsProfile      *resultsProfilePtr;
     case AVT_CRITERIA:
        Criteria             criteria;
     case AVT_JOB_PASSWORD:
        OctetString          jobPassword;
     case AVT_JOB_LEVEL:
        JobLevel             jobLevel;
     case AVT_JOB_CATEGORIES:
        JobCategories        jobCategories;
     case AVT_PRINT_CHECKPOINT:
        OctetString          printCheckpoint;
     case AVT_IGNORED_ATTRIBUTE:
        IgnoredAttribute    *ignoredAttributePtr;
     case AVT_RESOURCE:
        Resource             resource;
     case AVT_MEDIUM_SUBSTITUTION:
        MediumSubstitution  *mediumSubstitutionPtr;
     case AVT_FONT_SUBSTITUTION:
        FontSubstitution     fontSubstitution;
     case AVT_RESOURCE_CONTEXT_SEQ:
        ResourceContextSeq   resourceContextSeq;
     case AVT_SIDES:
        nint32               sides;
     case AVT_PAGE_SELECT_SEQ:
        PageSelectSequence   pageSelectSeq;
     case AVT_PAGE_MEDIA_SELECT:
        PageMediaSelect      pageMediaSelect;
     case AVT_DOCUMENT_CONTENT:
        DocumentContent     *documentContentPtr;
     case AVT_PAGE_SIZE:
        PageSize            *pageSizePtr;
     case AVT_PRESENTATION_DIRECTION:
        PresentationDirectionEnum presentationDirection;
     case AVT_PAGE_ORDER:
        PageOrderTypeEnum    pageOrderType;
     case AVT_FILE_REFERENCE:
        DistinguishedNameString fileReference;
```

```
        case AVT_MEDIUM_SOURCE_SIZE:
           MediumSourceSize    *mediumSourceSizePtr;
        case AVT_INPUT_TRAY_MEDIUM:
           InputTrayMedium     *inputTrayMediumPtr;
        case AVT_OUTPUT_BINS_CHARS:
           OutBinsCharacteristics outBinsCharacteristics;
        case AVT_PAGE_ID_TYPE:
           PageIdTypeEnum      pageIdType;
        case AVT_LEVEL_RANGE:
           LevelRange          levelRange;
        case AVT_CATEGORY_SET:
           CategorySet        *categorySetPtr;
        case AVT_NUMBERS_UP_SUPPORTED:
           NumbersUpSupported   numbersUpSupported;
        case AVT_FINISHING:
           Finishing           *finishingPtr;
        case AVT_PRT_CONTAINED_OBJ_ID:
           PrtContainedObjectId prtContainedObjectId;
        case AVT_PRT_CONFIG_OBJECT_ID:
           PrtConfigObjectId   *prtConfigObjectIdPtr;
        case AVT_TYPED_NAME:
           TypedName            typedName;
        case AVT_NET_ADDRESS:
           NetAddress           netAddress;
        case AVT_XY_DIMENSIONS_VALUE:
           XYDimensionsValue   *xyDimensionsValuePtr;
        case AVT_NAME_OR_OID_DIM_MAP:
           NameOrOidDimensionMap  *nameOrOidDimensionMapPtr;
        case AVT_PRINTER_STATE_REASON:
           PrinterStateReason  *printerStateReasonPtr;
        case AVT_ENUMERATION:
           nint32               enumeration;
         case AVT_QUALIFIED_NAME:
           QualifiedName        qualifiedName;
         case AVT_QUALIFIED_NAME_SET:
           QualifiedNameSet     qualifiedNameSet;
        case AVT_COLORANT_SET:
           ColorantSet         *colorantSetPtr;
        case AVT_RESOURCE_PRINTER_ID:
           ResourcePrinterId   *resourcePrinterIdPtr;
        case AVT_EVENT_OBJECT_ID:
           EventObjectId       *eventObjectIdPtr;
        case AVT_QUALIFIED_NAME_MAP:
           QualifiedNameMap    *qualifiedNameMapPtr;
        case AVT_FILE_PATH:
           Text                 filePath;
        };

     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<>;

   /* ------ filters  ---------------- */

   enum SubstringMatchCriteriaEnum {
      MATCH_EXACT,                             /* (0) */
      MATCH_CASE_INSENSITIVE,                  /* (1) */
      MATCH_SAME_LETTER,                       /* (2) */
      MATCH_APPROXIMATE                        /* (3) */
      };

   struct SubStrings {
      ObjectIdentifier     attributeId;
      SubstringMatchCriteriaEnum matchCriteria;
      AttributeValue       initialValueOption;
      AttributeValueSet    anyValueSetOption;
      AttributeValue       finalValueOption;
      };

   enum FilterItemEnum {
      FILTER_EQUALITY,                         /* (0) */
      FILTER_SUBSTRINGS,                       /* (1) */
      FILTER_GREATER_OR_EQUAL,                 /* (2) */
      FILTER_LESS_OR_EQUAL,                    /* (3) */
      FILTER_PRESENT,                          /* (4) */
      FILTER_SUBSET_OF,                        /* (5) */
      FILTER_SUPERSET_OF,                      /* (6) */
      FILTER_NON_NULL_SET_INTERS               /* (7) */
      };

   union FilterItem switch(FilterItemEnum designator) {
      case FILTER_EQUALITY:
         Attribute   equality;
      case FILTER_SUBSTRINGS:
         SubStrings  substrings;
      case FILTER_GREATER_OR_EQUAL:
         Attribute   greaterOrEqual;
      case FILTER_LESS_OR_EQUAL:
         Attribute   lessOrEqual;
      case FILTER_PRESENT:
         ObjectIdentifier  present;  /* AttributeId */
      case FILTER_SUBSET_OF:
         Attribute   subsetOf;
      case FILTER_SUPERSET_OF:
         Attribute   supersetOf;
      case FILTER_NON_NULL_SET_INTERS:
         Attribute   nonNullSetIntersect;
```

```
   };

enum FilterEnum {
   FILTER_ITEM,                          /* (0) */
   FILTER_AND,                           /* (1) */
   FILTER_OR,                            /* (2) */
   FILTER_NOT                            /* (5) */
   };

union Filter switch(FilterEnum designator) {
   case FILTER_ITEM:
      FilterItem  filterItem;
   case FILTER_AND:
      struct Filter   filterAnd<>;
   case FILTER_OR:
      struct Filter   filterOr<>;
   case FILTER_NOT:
      struct Filter  *filterNotPtr;
   };

enum DpAccessRightsEnum {
   DP_ACCESS_RIGHTS_USER,                /* (0) */
   DP_ACCESS_RIGHTS_ADMIN                /* (1) */
   };

struct DpAccessListElement {
   Text                accessId;
   DpAccessRightsEnum  accessRights;
   };

struct AccessAndAccounting {
   Text              userName;
   OctetString       accountingInformation;
   };

/*
** NOTE:
**   Define all error information to be returned in a
**   single structure
**   with a primary error code (error type)
**
**   Error messages are optional.
*/
enum ProblemTypeEnum {
   PROBLEM_TYPE_STANDARD,                  /* (0) */
   PROBLEM_TYPE_EXTENDED                   /* (1) */
   };

/***********************************************
 **          Access Error / Problem          **
 ***********************************************/

enum AccessProblemEnum {
   ACCESS_WRONG_OBJECT_CLASS,              /* (0) */
   ACCESS_LACK_ACCESS_RIGHTS,              /* (1) */
   ACCESS_CANT_INTERRUPT_JOB,              /* (2) */
   ACCESS_WRONG_OBJECT_STATE,              /* (3) */
```

```
         ACCESS_CLIENT_NOT_BOUND,                /* (4) */
         ACCESS_NOT_AVAILABLE,                   /* (5) */
         ACCESS_NOTIF_SVC_NOT_BOUND              /* (6) */
         };

      union AccessProblem switch(ProblemTypeEnum designator)
      {
         case PROBLEM_TYPE_STANDARD:
            AccessProblemEnum standardProblem;
         case PROBLEM_TYPE_EXTENDED:
            ObjectIdentifier  extendedProblem;
      };

      struct AccessError
      {
         AccessProblem           problem;
         ObjectIdentification    objectIdentification;
         NameOrOid              *messageOptionPtr;
      };

      /**********************************************
       **          Attribute Error / Problem        **
       **********************************************/

      enum AttributeProblemEnum {
         ATTR_INVALID_SYNTAX,                    /* (0)  */
         ATTR_UNDEFINED_TYPE,                    /* (1)  */
         ATTR_WRONG_MATCHING,                    /* (2)  */
         ATTR_CONSTRAINT_VIOLATED,               /* (3)  */
         ATTR_UNSUPPORTED_TYPE,                  /* (4)  */
         ATTR_ILLEGAL_MODIFICATION,              /* (5)  */
         ATTR_CONSIST_W_OTHER_ATTR,              /* (6)  */
         ATTR_UNDEFINED_ATTR_VALUE,              /* (7)  */
         ATTR_UNSUPPORTED_VALUE,                 /* (8) */
         ATTR_INVALID_NONCOMPUL_MOD,             /* (9) */
         ATTR_PER_JOB_INADMISSIBLE,              /* (10) */
         ATTR_NOT_MULTI_VALUED,                  /* (11) */
         ATTR_MANDATORY_OMITTED,                 /* (12) */
         ATTR_ILLEGAL_FOR_CLASS                  /* (13) */
         };

      union AttributeProblem switch(ProblemTypeEnum designator)
      {
         case PROBLEM_TYPE_STANDARD:
            AttributeProblemEnum standardProblem;
         case PROBLEM_TYPE_EXTENDED:
            ObjectIdentifier     extendedProblem;
      };

      struct AttributeProblemItem
      {
         AttributeProblem  problem;
         Attribute         attribute;
         NameOrOid        *messageOptionPtr;
      };

      struct AttributeError
```

```
    {
       AttributeProblemItem    problems<>;
       ObjectIdentification    objectIdentification;
    };

    /*********************************************
     **          Document Access Error / Problem  **
     *********************************************/

    enum DocAccessProblemEnum {
       DOC_ACCESS_NOT_AVAILABLE,              /* (0) */
       DOC_ACCESS_FIDELI_TIME_EXP,            /* (1) */
       DOC_ACCESS_DENIED,                     /* (2) */
       DOC_ACCESS_UNKNOWN_DOC,                /* (3) */
       DOC_ACCESS_NO_DOCS_IN_JOB              /* (4) */
       };

    union DocAccessProblem switch(ProblemTypeEnum designator)
    {
       case PROBLEM_TYPE_STANDARD:
          DocAccessProblemEnum standardProblem;
       case PROBLEM_TYPE_EXTENDED:
          ObjectIdentifier     extendedProblem;
    };

    struct DocAccessError
    {
       DocAccessProblem        problem;
       ObjectIdentification    objectIdentification;
       NameOrOid              *messageOptionPtr;
    };

    /*********************************************
     **          Printer Error / Problem         **
     *********************************************/

    enum PrinterProblemEnum {
       PRINTER_ERROR,                         /* (0) */
       PRINTER_NEEDS_ATTENTION,               /* (1) */
       PRINTER_NEEDS_KEY_OPERATOR             /* (2) */
       };

    union PrinterProblem switch(ProblemTypeEnum designator)
    {
       case PROBLEM_TYPE_STANDARD:
          PrinterProblemEnum   standardProblem;
       case PROBLEM_TYPE_EXTENDED:
          ObjectIdentifier     extendedProblem;
    };

    struct PrinterError
    {
       PrinterProblem          problem;
       ObjectIdentification    objectIdentification;
       NameOrOid              *messageOptionPtr;
    };
```

```
/*********************************************
 **      Security Access Error / Problem     **
 *********************************************/

enum SecurityProblemEnum {
   SECURITY_AUTHENTICATION,              /* (0) */
   SECURITY_CREDENTIALS,                 /* (1) */
   SECURITY_RIGHTS,                      /* (2) */
   SECURITY_INVALID_PAC                  /* (3) */
   };

union SecurityProblem switch(ProblemTypeEnum designator)
{
   case PROBLEM_TYPE_STANDARD:
      SecurityProblemEnum  standardProblem;
   case PROBLEM_TYPE_EXTENDED:
      ObjectIdentifier     extendedProblem;
};

struct SecurityError {
   SecurityProblem      problem;
   NameOrOid           *messageOptionPtr;
   };

/*********************************************
 **          Selection Error / Problem       **
 *********************************************/

enum SelectionProblemEnum {
   SELECTION_INVALID_ID,                 /* (0) */
   SELECTION_UNKNOWN_ID,                 /* (1) */
   SELECTION_OBJECT_EXISTS               /* (2) */
   };

union SelectionProblem switch(ProblemTypeEnum designator)
{
   case PROBLEM_TYPE_STANDARD:
      SelectionProblemEnum standardProblem;
   case PROBLEM_TYPE_EXTENDED:
      ObjectIdentifier     extendedProblem;
};

struct SelectionError
{
   SelectionProblem        problem;
   ObjectIdentification    objectIdentification;
   Attribute              *attributeOptionPtr;
   NameOrOid              *messageOptionPtr;
};

/*********************************************
 **          Service Error / Problem         **
 *********************************************/

enum ServiceProblemEnum {
   SERVICE_SERVER_BUSY,                  /* (0)  */
   SERVICE_SERVER_UNAVAILABLE,           /* (1)  */
```

```
      SERVICE_OPERATION_COMPLEX,            /* (2)  */
      SERVICE_RESOURCE_LIMIT,               /* (3)  */
      SERVICE_UNCLASS_SERVER_ERR,           /* (4)  */
      SERVICE_TOO_MANY_ITEM_LIST,           /* (5)  */
      SERVICE_RESOURCE_NOT_AVAIL,           /* (6)  */
      SERVICE_CANCEL_DOC_SUPPORT,           /* (7)  */
      SERVICE_MODIFY_DOC_SUPPORT,           /* (8)  */
      SERVICE_MULTI_DOC_SUPPORT,            /* (9)  */
      SERVICE_PARM_VAL_SUPPORT,             /* (10) */
      SERVICE_INVALID_CHECKPOINT,           /* (11) */
      SERVICE_CONTINUATN_CONTEXT,           /* (12) */
      SERVICE_PAUSE_LIMIT_EXCEED,           /* (13) */
      SERVICE_UNSUPPORTED_OP,               /* (14) */
      SERVICE_NOTIF_SERVICE_ERR             /* (15) */
      };

   union ServiceProblem switch(ProblemTypeEnum designator)
   {
      case PROBLEM_TYPE_STANDARD:
         ServiceProblemEnum   standardProblem;
      case PROBLEM_TYPE_EXTENDED:
         ObjectIdentifier     extendedProblem;
   };

   struct ServiceError
   {
      ServiceProblem          problem;
      ObjectIdentification    objectIdentification;
      Attribute              *attributeOptionPtr;
      nuint32                 libError;
      nuint32                 otherError;
      nuint32                 otherError2;
      NameOrOid              *messageOptionPtr;
   };

   /**********************************************
    **           Update Error / Problem          **
    **********************************************/

   enum UpdateProblemEnum {
      UPDATE_NO_MODS_ALLOWED,               /* (0) */
      UPDATE_INSUFFICIENT_RIGHTS,           /* (1) */
      UPDATE_PREV_OP_INCOMPLETE,            /* (2) */
      UPDATE_CANCEL_NOT_POSSIBLE            /* (3) */
      };

   union UpdateProblem switch(ProblemTypeEnum designator)
   {
      case PROBLEM_TYPE_STANDARD:
         UpdateProblemEnum standardProblem;
      case PROBLEM_TYPE_EXTENDED:
         ObjectIdentifier  extendedProblem;
   };

   struct UpdateError
   {
      UpdateProblem           problem;
```

```
      ObjectIdentification    objectIdentification;
      NameOrOid              *messageOptionPtr;
   };

   /**********************************************
    **          Update Error / Problem          **
    **********************************************/

   /* NOTE:
   **   This enumeration is ordered according to the error precedence
   **   specified in DPA.
   */
   enum ErrorTypeEnum {
      ERROR_TYPE_SECURITY,                       /* (0) */
      ERROR_TYPE_SERVICE,                        /* (1) */
      ERROR_TYPE_ACCESS,                         /* (2) */
      ERROR_TYPE_PRINTER,                        /* (3) */
      ERROR_TYPE_SELECTION,                      /* (4) */
      ERROR_TYPE_DOCUMENT_ACCESS,                /* (5) */
      ERROR_TYPE_ATTRIBUTE,                      /* (6) */
      ERROR_TYPE_UPDATE                          /* (7) */
      };

   union ErrorReturn switch(ErrorTypeEnum designator)
   {
      case ERROR_TYPE_SECURITY:
         SecurityError     securityError;
      case ERROR_TYPE_SERVICE:
         ServiceError     serviceErrorSeq<>;
      case ERROR_TYPE_ACCESS:
         AccessError      accessErrorSeq<>;
      case ERROR_TYPE_PRINTER:
         PrinterError     printerError;
      case ERROR_TYPE_SELECTION:
         SelectionError   selectionErrorSeq<>;
      case ERROR_TYPE_DOCUMENT_ACCESS:
         DocAccessError   docAccessError;
      case ERROR_TYPE_ATTRIBUTE:
         AttributeError   attributeError;
      case ERROR_TYPE_UPDATE:
         UpdateError      updateError;
   };

   /* *********************** end of LdpaTY.x
   ***************************** */
```

3. ldpaav.x

```
/*
//*****************************************************
//* Source module name: idpav.x                         *
//*****************************************************
*/

/***********************************************************
 * This enumeration is required in AttributeValue in ldpaty.x.
 ***********************************************************/


/*
 * symbol             value : basic type        protocol passes-by
 */


enum AVTEnum {
   AVT_NULL,               /*  0 null                    placeholder */
   AVT_TEXT,               /*  1 text                    reference   */
   AVT_DESCRIPTIVE_NAME,   /*  2 text                    reference   */
   AVT_DESCRIPTOR,         /*  3 text                    reference   */
   AVT_MESSAGE,            /*  4 text                    reference   */
   AVT_ERROR_MESSAGE,      /*  5 text                    reference   */
   AVT_SIMPLE_NAME,        /*  6 text                    reference   */
   AVT_DISTINGUISHED_NAME_STR,
                           /*  7 text, syntax oid        reference   */
   AVT_DIST_NAME_STR_SEQ,  /*  8 text, syntax oid        reference   */
   AVT_DELTA_TIME,         /*  9 integer                 value       */
   AVT_TIME,               /* 10 cardinal                value       */
   AVT_INTEGER,            /* 11 integer                 value       */
   AVT_INTEGER_SEQ,        /* 12 integer seq             value       */
   AVT_CARDINAL,           /* 13 unsigned integer        value       */
   AVT_CARDINAL_SEQ,       /* 14 unsigned integer seq reference      */
   AVT_POSITIVE_INTEGER,   /* 15 integer>0               value       */
   AVT_INTEGER_RANGE,      /* 16 struct                  value       */
   AVT_CARDINAL_RANGE,     /* 17 struct                  value       */
   AVT_MAXIMUM_INTEGER,    /* 18 integer                 value       */
   AVT_MINIMUM_INTEGER,    /* 19 integer                 value       */
   AVT_INTEGER_64,         /* 20 integer 64              value       */
   AVT_INTEGER_64_SEQ,     /* 21 struct                  reference   */
   AVT_CARDINAL_64,        /* 22 integer 64              reference   */
   AVT_CARDINAL_64_SEQ,    /* 23 struct                  reference   */
   AVT_POSITIVE_INTEGER_64,
                           /* 24 integer 64 >0           value       */
   AVT_INTEGER_64_RANGE,   /* 25 struct                  value       */
   AVT_CARDINAL_64_RANGE,  /* 26 struct                  value       */
   AVT_MAXIMUM_INTEGER_64,
                           /* 27 integer 64              value       */
   AVT_MINIMUM_INTEGER_64,
                           /* 28 integer 64              value       */
   AVT_REAL,               /* 29 real                    value       */
   AVT_REAL_SEQ,           /* 30 real seq                reference   */
   AVT_NON_NEGATIVE_REAL,  /* 31 real                    value       */
   AVT_REAL_RANGE,         /* 32 struct                  reference   */
   AVT_NON_NEGATIV_REAL_RANGE,
```

```
                          /* 33 struct              value       */
        AVT_BOOLEAN,          /* 34 bool                value       */
        AVT_PERCENT,          /* 35 Integer(0..100)     value       */
        AVT_OBJECT_IDENTIFIER, /* 36 struct             reference   */
        AVT_OBJECT_IDENTIFIER_SEQ,
                              /* 37 struct              reference   */
        AVT_NAME_OR_OID,      /* 38 struct              reference   */
        AVT_NAME_OR_OID_SEQ,  /* 39 struct              reference   */
        AVT_DISTINGUISHED_NAME,
                              /* 40 text                reference   */
        AVT_RDN_SEQUENCE,     /* 41 struct              reference   */
        AVT_REALIZATION,      /* 42 integer enum        value       */
        AVT_MEDIUM_DIMENSIONS, /* 43 real struct        value       */
        AVT_DIMENSION,        /* 44 struct              reference   */
        AVT_XY_DIMENSIONS,    /* 45 struct              reference   */
        AVT_LOCATIONS,        /* 46 struct              reference   */
        AVT_AREA,             /* 47 struct              reference   */
        AVT_AREA_SEQ,         /* 48 struct              reference   */
        AVT_EDGE,             /* 49 integer enum        value       */
        AVT_FONT_REFERENCE,   /* 50 struct              reference   */
        AVT_CARDINAL_OR_OID,  /* 51 struct              reference   */
        AVT_OID_CARDINAL_MAP, /* 52 struct              reference   */
        AVT_CARDINAL_OR_NAM_OR_OID,
                              /* 53 struct              reference   */
        AVT_POSITIVE_INT_OR_OID,
                              /* 54 struct              reference   */
        AVT_EVENT_HANDLING_PROFILE,
                              /* 55 struct              reference   */
        AVT_OCTET_STRING,     /* 56 struct              reference   */
        AVT_PRIORITY,         /* 57 integer             value       */
        AVT_LOCALE,           /* 58 text                            */
        AVT_METHOD_DELIVERY_ADDR,
                              /* 59 struct              reference   */
        AVT_OBJECT_IDENTIFICATION,
                              /* 60 struct              reference   */
        AVT_RESULTS_PROFILE,  /* 61 struct              reference   */
        AVT_CRITERIA,         /* 62 struct              reference   */
        AVT_JOB_PASSWORD,     /* 63 struct              reference   */
        AVT_JOB_LEVEL,        /* 64 struct              reference   */
        AVT_JOB_CATEGORIES,   /* 65 struct              reference   */
        AVT_PRINT_CHECKPOINT, /* 66 struct              reference   */
        AVT_IGNORED_ATTRIBUTE, /* 67 struct             reference   */
        AVT_RESOURCE,         /* 68                                 */
        AVT_MEDIUM_SUBSTITUTION,
                              /* 69 struct              reference   */
        AVT_FONT_SUBSTITUTION, /* 70 struct             reference   */
        AVT_RESOURCE_CONTEXT_SEQ,
                              /* 71                                 */
        AVT_SIDES,            /* 72 integer             value       */
        AVT_PAGE_SELECT_SEQ,  /* 73 struct              reference   */
        AVT_PAGE_MEDIA_SELECT, /* 74 struct             reference   */
        AVT_DOCUMENT_CONTENT, /* 75 struct              reference   */
        AVT_PAGE_SIZE,        /* 76                                 */
        AVT_PRESENTATION_DIRECTION,
                              /* 77 integer             value       */
        AVT_PAGE_ORDER,       /* 78 enum                value       */
        AVT_FILE_REFERENCE,   /* 79                                 */
```

```
        AVT_MEDIUM_SOURCE_SIZE,/* 80 struct               reference   */
        AVT_INPUT_TRAY_MEDIUM, /* 81 struct               reference   */
        AVT_OUTPUT_BINS_CHARS, /* 82 struct               value       */
        AVT_PAGE_ID_TYPE,      /* 83 integer              value       */
        AVT_LEVEL_RANGE,       /* 84 struct               reference   */
        AVT_CATEGORY_SET,      /* 85 struct               reference   */
        AVT_NUMBERS_UP_SUPPORTED,
                               /* 86 struct               reference   */
        AVT_FINISHING,         /* 87 struct               reference   */
        AVT_PRT_CONTAINED_OBJ_ID,
                               /* 88 struct               reference   */
        AVT_PRT_CONFIG_OBJECT_ID,
                               /* 89 struct               reference   */
        AVT_TYPED_NAME,        /* 90 struct               reference   */
        AVT_NET_ADDRESS,       /* 91 struct               reference   */
        AVT_XY_DIMENSIONS_VALUE,
                               /* 92 struct               reference   */
        AVT_NAME_OR_OID_DIM_MAP,
                               /* 93 struct               reference   */
        AVT_PRINTER_STATE_REASON,
                               /* 94 struct               value       */
        AVT_ENUMERATION,       /* 95 integer              value       */
        AVT_QUALIFIED_NAME,    /* 96 struct               reference   */
        AVT_QUALIFIED_NAME_SET,/* 97 struct               reference   */
        AVT_COLORANT_SET,      /* 98 struct               reference   */
        AVT_RESOURCE_PRINTER_ID,
                               /* 99 struct               value       */
        AVT_EVENT_OBJECT_ID,   /* 100 struct              reference   */
        AVT_QUALIFIED_NAME_MAP,/* 101 struct              reference   */
        AVT_FILE_PATH          /* 102 struct              value       */
        };

    /* ***************** end of ldpaav.x  ******************** */
```