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;
```

```
91          opaque            bindSecurityOption<>;
92          };
93
94       struct BindArgument {
95          Credentials       credentials;
96          nint32            retrieveRestrictionsOption;
97          opaque            bindSecurityOption<>;
98          };
99
100      struct BindResult {
101         OctetString       authAttributeSet<>;
102         ErrorReturn      *errorReturnOptionPtr;
103         nint32            sessionHandle;
104         };
105
106      /*******************************************
107       **          Unbind Operation Parameters     **
108       *******************************************/
109
110      struct UnbindArgument {
111         nint32            sessionHandle;
112         };
113
114      struct UnbindResult {
115         ErrorReturn      *errorReturnOptionPtr;
116        };
117
118      /*******************************************
119       **          Print Operation Parameters      **
120       *******************************************/
121
122      struct DocumentDescription {
123         ObjectIdentifier        transferMethod;
124         DocumentContent        *documentContentOptionPtr;
125         ObjectIdentifier        documentType;
126         AttributeSet            documentAttributes;
127         };
128
129      struct CreateJob {
130         QualifiedName           printerName;
131         bool                    jobSubmissionComplete;
132         AttributeSet            jobAttributes;
133         DocumentDescription    *firstDocumentOptionPtr;
134         CommonArguments         commonArgumentsOption;
135         };
136
137      struct AddDocument {
138         PrtContainedObjectId    existingJob;
139         bool                    jobSubmissionComplete;
140         DocumentDescription    *newDocumentPtr;
141         CommonArguments         commonArgumentsOption;
142         };
143
144      struct CloseJob {
145         PrtContainedObjectId    existingJob;
146         CommonArguments         commonArgumentsOption;
147         };
```

```
148         enum PrintArgEnum {
149            PRINT_ARG_CREATE_JOB,               /* (0) */
150            PRINT_ARG_ADD_DOCUMENT,             /* (1) */
151            PRINT_ARG_CLOSE_JOB                 /* (2) */
152           };
153
154         union PrintOperation switch(PrintArgEnum designator) {
155            case PRINT_ARG_CREATE_JOB:
156               CreateJob     createJob;
157            case PRINT_ARG_ADD_DOCUMENT:
158               AddDocument   addDocument;
159            case PRINT_ARG_CLOSE_JOB:
160               CloseJob      closeJob;
161            };
162
163         struct PrintArgument {
164            nint32            sessionHandle;
165            PrintOperation    printOperation;
166            };
167
168
169         struct PrintResult {
170            PrtContainedObjectId    jobIdentification;
171            ObjectIdentifier        serverStateOption;
172            NameOrOid              *serverMessageOptionPtr;
173            AttributeSet            documentStatusOption;
174            AttributeSet            jobStatus;
175            ErrorReturn            *errorReturnOptionPtr;
176            };
177
178         /*******************************************
179          **          Modify Operation Parameters     **
180          *******************************************/
181
182         enum ModifyOperatorEnum {
183            MODIFY_OP_NULL,                      /* (0) */
184            MODIFY_OP_REPLACE,                   /* (1) */
185            MODIFY_OP_ADD_VALUES,                /* (2) */
186            MODIFY_OP_REMOVE_VALUES,             /* (3) */
187            MODIFY_OP_SET_TO_DEFAULT,            /* (4) */
188            MODIFY_OP_REMOVE_ATTRIBUTE           /* (5) */
189            };
190
191         struct ModifyJobArgument {
192            nint32                  sessionHandle;
193            PrtContainedObjectId    jobIdentification;
194            nuint32                 documentNumberOption;
195            AttributeSet            jobAttrModificationSet;
196            AttributeSet            docAttrModificationSet;
197            NameOrOid              *modifyMessageOptionPtr;
198            CommonArguments         commonArgumentsOption;
199            };
200
201         struct ModifyJobResult {
202            AttributeSet            statusOption;
203            ErrorReturn            *errorReturnOptionPtr;
204            };
```

```
205
206         /*******************************************
207          **          Cancel Operation Parameters     **
208          *******************************************/
209
210         struct CancelJobArgument {
211            nint32                  sessionHandle;
212            PrtContainedObjectId    jobIdentifier;
213            nuint32                 documentNumberOption;
214            NameOrOid              *cancelMessageOptionPtr;
215            IntegerOption           retentionPeriodOption;
216            CommonArguments         commonArgumentsOption;
217            };
218
219         struct CancelJobResult {
220            AttributeSet            jobStatusOption;
221            ErrorReturn            *errorReturnOptionPtr;
222            };
223
224
225         /*******************************************
226          **          List Object Attributes          **
227          *******************************************/
228
229         struct Selector {
230            ObjectIdentificationSeq objectIdentificationSeqOption;
231            Filter                 *objectFilterOptionPtr;
232            nuint32                 timeLimitOption;
233            nuint32                 countLimitOption;
234            };
235
236         enum ListOperatorEnum {
237            LIST_OP_ATTRIBUTES,                      /* (0) */
238            LIST_OP_ORDERED_JOBS = 2                 /* (1) */
239            };
240
241         struct  ListSpecification {
242            ObjectIdentifier        objectClass;
243            nuint32                 scope;              /* default 0; */
244            Selector               *selectorOptionPtr;
245            ObjectIdentifierSet    *requestedAttrsOptionPtr;
246            ListOperatorEnum        listOperator;    /* default
247         DpaReturnAttributes */
248            CommonArguments         commonArgumentsOption;
249            };
250
251         struct ListContinuation {
252            OctetString             context;
253            bool                    abort;
254            CommonArguments         commonArgumentsOption;
255            };
256
257         enum ListAttrsArgEnum  {
258            LIST_ATTRIBUTES_ARG_CONTINUE,            /* (0) */
259            LIST_ATTRIBUTES_ARG_SPEC                 /* (1) */
260            };
261
```

```
262         union ListAttrsOperation switch(ListAttrsArgEnum designator) {
263            case LIST_ATTRIBUTES_ARG_CONTINUE:
264               ListContinuation     continuation;
265            case LIST_ATTRIBUTES_ARG_SPEC:
266               ListSpecification    specification;
267            };
268
269         struct ListObjectAttrsArgument {
270            nint32                  sessionHandle;
271            ListAttrsOperation      listAttrsOperation;
272            };
273
274         enum LimitEncounteredEnum {
275            LIMIT_ENCOUNTERED_TIME,                  /* (0) */
276            LIMIT_ENCOUNTERED_COUNT,                 /* (1) */
277            LIMIT_ENCOUNTERED_ERROR                  /* (2) */
278            };
279
280         struct LimitEncounteredOption {
281            nint32                  length;          /* 0 or 1 */
282            LimitEncounteredEnum    value;
283            };
284
285         struct ObjectResult {
286            ObjectIdentification    objectIdentification;
287            AttributeSet            attributes;
288            ObjectIdentifier        objectClass;
289            };
290
291         typedef ObjectResult ObjectResultSet<>;
292
293         struct ListObjectAttrsResult {
294            nuint32                 answerTime;
295            OctetString             continuationOption;
296            LimitEncounteredOption  limitEncounteredOption;
297            ObjectResultSet         resultSet;
298            ErrorReturn            *errorReturnOptionPtr;
299            };
300
301
302         /*********************************************
303          **       PromoteJob Operation Parameters    **
304          *********************************************/
305
306         struct PromoteJobArgument {
307            nint32                  sessionHandle;
308            PrtContainedObjectId    jobIdentifier;
309            NameOrOid              *messageOptionPtr;
310            CommonArguments         commonArgumentsOption;
311            };
312
313         struct PromoteJobResult {
314            AttributeSet            jobStatusOption;
315            ErrorReturn            *errorReturnOptionPtr;
316            };
317
318
```

```
319        /*********************************************
320         **       Pause Operation Parameters         **
321         *********************************************/
322
323        enum PauseJobEnum  {
324           PAUSE_JOB_ID,                        /* (0) */
325           PAUSE_PRINTER_NM                     /* (1) */
326           };
327
328        union PauseJob switch(PauseJobEnum designator) {
329           case PAUSE_JOB_ID:
330                PrtContainedObjectId     jobIdentifier;
331           case PAUSE_PRINTER_NM:
332                QualifiedName           printerName;
333           };
334
335        struct PauseArgument {
336           nint32                   sessionHandle;
337           PauseJob                 pausedJob;
338           NameOrOid               *pauseMessageOptionPtr;
339           CommonArguments          commonArgumentsOption;
340           };
341
342        struct PauseResult {
343           PrtContainedObjectId pausedJobId;
344           AttributeSet             jobStatusOption;
345           ErrorReturn             *errorReturnOptionPtr;
346         };
347
348        /*********************************************
349         **       Resume Operation Parameters        **
350         *********************************************/
351
352        struct ResumeArgument {
353           nint32                   sessionHandle;
354           PrtContainedObjectId     resumedJob;
355           NameOrOid               *resumeMessageOptionPtr;
356           CommonArguments          commonArgumentsOption;
357           };
358
359        struct ResumeResult {
360           AttributeSet             jobStatusOption;
361           ErrorReturn             *errorReturnOptionPtr;
362           };
363
364
365
366        /*********************************************
367         **             ResubmitJobs parameters         **
368         *********************************************/
369
370        enum ResubmitOpEnum {
371           RESUBMIT_OP_COPY,                        /* (0) */
372           RESUBMIT_OP_MOVE                         /* (1) */
373           };
374
375        /*
```

```
376         // If documentNumber is 0, docAttrSet is applied to all documents
377         */
378
379         struct ResubmitJob {
380            PrtContainedObjectId    jobId;
381            nuint32                 documentNumber;
382            AttributeSet            jobAttrSet;
383            AttributeSet            docAttrSet;
384            };
385
386         typedef ResubmitJob ResubmitJobSet<>;
387
388         struct ResubmitJobsArgument {
389            nint32                    sessionHandle;
390            QualifiedName             destPrinterNameOption;
391            NetAddress               *destPrinterNetAddressPtr;
392            ResubmitOpEnum            operation;
393            ResubmitJobSet            resubmitJobSet;
394            NameOrOid                *resubmitMessageOptionPtr;
395            CommonArguments           commonArgumentsOption;
396           };
397
398         struct ResubmitJobResult {
399            PrtContainedObjectId    oldJobIdentifier;
400            PrtContainedObjectId    newJobIdentifier;
401            AttributeSet            jobStatusOption;
402            };
403
404         typedef ResubmitJobResult ResubmitJobResultSet<>;
405
406         struct ResubmitJobsResult {
407            ResubmitJobResultSet    resubmitJobResultSet;
408            ErrorReturn            *errorReturnOptionPtr;
409            };
410
411
412         program PRINTPROGRAM {
413           version PRINTVERS {
414
415            BindResult
416            BindPrinter ( BindPrinterArgument
417            ) = 1;
418
419            UnbindResult
420            Unbind ( UnbindArgument
421            ) = 2;
422
423            PrintResult
424            Print ( PrintArgument
425            ) = 3;
426
427            ModifyJobResult
428            ModifyJob ( ModifyJobArgument
429            ) = 4;
430
431            CancelJobResult
432            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
```

```
518      //        array item count contains the number of bytes, rather
519      //         than the number of 16-bit characters.
520      */
521
522      typedef opaque Text<>;
523
524
525      /* ------ name or OID ------ */
526      /* this definition is placed here to eliminate forward reference */
527      enum NameOrOidEnum {
528         NAME_OR_OID_GLOBAL,                  /* (0) */
529         NAME_OR_OID_LOCAL                    /* (1) */
530         };
531
532      union NameOrOid switch(NameOrOidEnum designator)  {
533         case NAME_OR_OID_GLOBAL:
534            ObjectIdentifier  globalForm;
535         case NAME_OR_OID_LOCAL:
536            Text              localForm;
537         } ;
538
539      /* ------ distinguishedNameString ------ */
540      struct DistinguishedNameString {
541         Text              name;
542         NameOrOid        *syntaxOptionPtr;
543         };
544
545      /*
546      // The QualifiedName union is intended to handle object names
547      //  with context
548      //  information.  Example:  NDS Tree Name
549      //  For now, provision is made for a distinguished name with no tree
550      name.
551      */
552
553      enum QualifiedNameEnum {
554         QUALIFIED_NAME_NONE,
555         QUALIFIED_NAME_OTHER
556         };
557
558      union QualifiedName switch (QualifiedNameEnum designator) {
559         case QUALIFIED_NAME_NONE:
560            void;
561         case QUALIFIED_NAME_OTHER:
562            Text      otherName;
563         };
564
565      typedef QualifiedName QualifiedNameSet<>;
566
567      /* ------ distinguishedNameStringSequence ------ */
568      typedef DistinguishedNameString DistinguishedNameStringSeq<>;
569
570
571      /* ------ integer sequence ------ */
572      typedef nint32 IntegerSeq<>;
573
574      /* ------ 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:
```

```
746            NameOrOid         nameOrOid;
747         };
748
749      /* ------ positiveIntegerOrOid type  ------ */
750
751      enum PositiveIntegerOrOidEnum {
752         INT_OR_OID_ID,                        /* (0) */
753         INT_OR_OID_NUMBER                     /* (1) */
754         };
755
756      union PositiveIntegerOrOid switch (
757                  PositiveIntegerOrOidEnum designator) {
758         case INT_OR_OID_ID:
759            ObjectIdentifier  oid;
760         case INT_OR_OID_NUMBER:
761            nint32            integer;
762         };
763
764      /* ------ job identifier ------ */
765      /* defined here to eliminate forward reference */
766
767      struct PrtContainedObjectId {
768          Text          printerName;
769          nuint32       localIdentifier;
770          };
771      typedef PrtContainedObjectId PrtContainedObjectIdSet<>;
772
773      /* ------ document identifier ------ */
774      struct DocumentIdentifier {
775         PrtContainedObjectId jobIdentifier;
776         nuint32              documentNumber;
777         } ;
778
779      enum DeliveryAddressEnum {
780         DA_OR_ADDR_AND_OR_DIR_NAME,           /* (0) */
781         DA_DISTINGUISHED_NAME,                /* (1) */
782         DA_TEXT,                              /* (2) */
783         DA_OCTET_STRING,                      /* (3) */
784         DA_DIST_NAME_STRING,                  /* (4) */
785         DA_RPC_ADDRESS,                       /* (5) */
786         DA_QUALIFIED_NAME                     /* (6) */
787         };
788
789      union DeliveryAddress switch(DeliveryAddressEnum designator) {
790         case DA_OR_ADDR_AND_OR_DIR_NAME:
791            Text     mhsAddress;
792         case DA_DISTINGUISHED_NAME:
793            Text     distinguishedName;
794         case DA_TEXT:
795            Text     text;
796         case DA_OCTET_STRING:
797            OctetString    octetString;
798         case DA_DIST_NAME_STRING:
799            DistinguishedNameString distinguishedNameString;
800         case DA_RPC_ADDRESS:
801            void;
802         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) */
```

```
860         OBJ_ID_OBJECT_NAME,                   /* (4) */
861         OBJ_ID_NAME_OR_OID,                   /* (5) */
862         OBJ_ID_SIMPLE_NAME,                   /* (6) */
863         OBJ_ID_PRT_CONFIG_OBJ_ID,             /* (7) */
864         OBJ_ID_QUALIFIED_NAME,                /* (8) */
865         OBJ_ID_EVENT_OBJECT_ID                /* (9) */
866         };
867
868      union ObjectIdentification switch(ObjectIdentificationEnum
869      designator) {
870         case OBJ_ID_NONE:
871            void;
872         case OBJ_ID_PRT_CONTAIND_OBJ_ID:
873            PrtContainedObjectId    prtContainedObjectId;
874         case OBJ_ID_DOCUMENT_IDENTIFIER:
875            DocumentIdentifier      documentIdentifier;
876         case OBJ_ID_OBJECT_IDENTIFIER:
877            ObjectIdentifier        objectIdentifier;
878         case OBJ_ID_OBJECT_NAME:
879            DistinguishedNameString objectName;
880         case OBJ_ID_NAME_OR_OID:
881            NameOrOid               nameOrOid;
882         case OBJ_ID_SIMPLE_NAME:
883            Text                    simpleName;
884         case OBJ_ID_PRT_CONFIG_OBJ_ID:
885            PrtConfigObjectId       prtConfigObjectId;
886         case OBJ_ID_QUALIFIED_NAME:
887            QualifiedName           qualifiedName;
888         case OBJ_ID_EVENT_OBJECT_ID:
889            EventObjectId           eventObjectId;
890         };
891
892      typedef ObjectIdentification ObjectIdentificationSeq<>;
893
894
895      struct TypedName {
896         Text              name;
897         nint32            level;
898         nint32            interval;
899         };
900
901      enum NetAddressTypeEnum {
902         NET_IPX,
903         NET_IP,
904         NET_SDLC,
905         NET_TOKENRING_ETHERNET,
906         NET_OSI,
907         NET_APPLETALK,
908         NET_COUNT
909         };
910
911      struct NetAddress {
912         NetAddressTypeEnum   type;
913         OctetString          address;
914         };
915
916      struct NameOrOidDimensionMap {
```

```
917         NameOrOid            nameOrOid;
918         XYCardinalDimensions value;
919         };
920
921      enum StateSeverityEnum {
922         STATE_SEVERITY_OTHER = 1,              /* 1 */
923         STATE_SEVERITY_WARNING,                /* 2 */
924         STATE_SEVERITY_CRITICAL                /* 3 */
925         };
926
927      enum TrainingEnum {
928         TRAINING_OTHER = 1,                    /* 1 */
929         TRAINING_UNKNOWN,                      /* 2 */
930         TRAINING_UNTRAINED,                    /* 3 */
931         TRAINING_TRAINED,                      /* 4 */
932         TRAINING_FIELD_SERVICE,                /* 5 */
933         TRAINING_MANAGEMENT                    /* 6 */
934         };
935
936      struct PrinterStateReason {
937         NameOrOid            identifier;
938         StateSeverityEnum    severity;
939         TrainingEnum         trainingLevel;
940         ObjectIdentifier     objectClassOid;
941         ObjectIdentification objectIdentification;
942         nuint32              time;
943         Text                 messageOption;
944         };
945
946      enum PersistenceEnum {
947         PERSISTENCE_PERMANENT,                 /*  0 */
948         PERSISTENCE_VOLATILE                   /*  1 */
949         };
950
951      enum AddressItemTypeEnum  {
952         ADDR_ITEM_USER,                /* 0  */
953         ADDR_ITEM_SERVER,              /* 1  */
954         ADDR_ITEM_VOLUME,              /* 2  */
955         ADDR_ITEM_ORG_UNIT,            /* 3  */
956         ADDR_ITEM_ORG,                 /* 4  */
957         ADDR_ITEM_GROUP,               /* 5  */
958         ADDR_ITEM_DN,                  /* 6  */
959         ADDR_ITEM_USR_OR_CONTAINER,    /* 7  */
960         ADDR_ITEM_CASE_EXACT_STR,      /* 8  */
961         ADDR_ITEM_CASE_IGNORE_STR,     /* 9  */
962         ADDR_ITEM_NUMERIC_STR,         /* 10 */
963         ADDR_ITEM_DOS_FILENAME,        /* 11 */
964         ADDR_ITEM_PHONE_NUMBER,        /* 12 */
965         ADDR_ITEM_BOOLEAN,             /* 13 */
966         ADDR_ITEM_INTEGER,             /* 14 */
967         ADDR_ITEM_NET_ADDRESS,         /* 15 */
968         ADDR_ITEM_OCTET_STRING         /* 16 */
969         };
970
971      union AddressItem switch(AddressItemTypeEnum designator) {
972         case ADDR_ITEM_USER:
973            QualifiedName     userName;
```

```
974          case ADDR_ITEM_SERVER:
975             QualifiedName     serverName;
976          case ADDR_ITEM_VOLUME:
977             QualifiedName     volumeName;
978          case ADDR_ITEM_ORG_UNIT:
979             QualifiedName     orgUnitName;
980          case ADDR_ITEM_ORG:
981             QualifiedName     orgName;
982          case ADDR_ITEM_GROUP:
983             QualifiedName     groupName;
984          case ADDR_ITEM_DN:
985             QualifiedName     dN;
986          case ADDR_ITEM_USR_OR_CONTAINER:
987             QualifiedName     userOrContainerName;
988          case ADDR_ITEM_CASE_EXACT_STR:
989             Text              ceStr;
990          case ADDR_ITEM_CASE_IGNORE_STR:
991             Text              ciStr;
992          case ADDR_ITEM_NUMERIC_STR:
993             Text              numStr;
994          case ADDR_ITEM_DOS_FILENAME:
995             Text              filename;
996          case ADDR_ITEM_PHONE_NUMBER:
997             Text              phoneNumber;
998          case ADDR_ITEM_BOOLEAN:
999             bool              bFlag;
1000         case ADDR_ITEM_INTEGER:
1001            nint32            integer;
1002         case ADDR_ITEM_NET_ADDRESS:
1003            NetAddress        netAddress;
1004         case ADDR_ITEM_OCTET_STRING:
1005            OctetString       octetString;
1006         };
1007
1008      typedef  AddressItem  NotifyDeliveryAddr<>;
1009
1010      enum EventObjectOperation {
1011         EVENT_OBJ_OP_NONE,
1012         EVENT_OBJ_OP_ADD_FLAG,
1013         EVENT_OBJ_OP_DELETE_FLAG,
1014         EVENT_OBJ_OP_DELETE_OBJECT
1015         };
1016
1017      enum EventObjectTypeEnum {
1018         EVENT_OBJ_TYPE_OBJECT,
1019         EVENT_OBJ_TYPE_FILTER,
1020         EVENT_OBJ_TYPE_DETAIL
1021         };
1022
1023      union EventObjectItem switch (EventObjectTypeEnum designator) {
1024         case EVENT_OBJ_TYPE_OBJECT:
1025            void;
1026         case EVENT_OBJ_TYPE_FILTER:
1027            ObjectIdentifier     filterClassOid;
1028         case EVENT_OBJ_TYPE_DETAIL:
1029            ObjectIdentifierSet  eventOidSet;
1030         };
```

```
1031       struct EventObject {
1032          nuint32                eventType;
1033          ObjectIdentifier       containingClassOid;
1034          ObjectIdentification   containingObjectId;
1035          EventObjectOperation   opCode;
1036          EventObjectItem        item;
1037          };
1038
1039       typedef EventObject EventObjectSet<>;
1040
1041       struct EventHandlingProfile {
1042          nuint32                profileId;
1043          PersistenceEnum        persistence;
1044          QualifiedName          consumerName;
1045          OctetString            supplierId;
1046          nint32                 languageId;
1047          NameOrOid              methodId;
1048          NotifyDeliveryAddr     deliveryAddress;
1049          EventObjectSet         eventObjectSet;
1050          QualifiedName          accountOption;
1051          };
1052
1053       struct ModifyEventProfileArgument {
1054          nuint32                sessionHandle;
1055          nuint32                profileId;
1056          bool                   supplierFlag;
1057          OctetString            supplierIdOption;
1058          bool                   languageFlag;
1059          nint32                 languageId;            /* Optional */
1060          bool                   methodFlag;
1061          NameOrOid             *methodIdOptionPtr;     /* Optional */
1062          bool                   deliveryAddrFlag;
1063          NotifyDeliveryAddr    *deliveryAddrOptionPtr;  /* Optional */
1064          EventObjectSet         eventObjectSet;        /* Optional */
1065          };
1066
1067       enum ListProfilesChoiceEnum {
1068          LIST_CHOICE_ID,                       /* (0) */
1069          LIST_CHOICE_FILTER                    /* (1) */
1070          };
1071
1072       enum ListProfilesResultEnum {
1073          LIST_RESULT_COMPLETE,
1074          LIST_RESULT_NO_EVENT_OBJS,
1075          LIST_RESULT_PROFILE_IDS
1076          };
1077
1078       struct ListProfilesFilter {
1079          QualifiedName          consumerNameOption;  /* Optional */
1080          NameOrOid             *methodIdOptionPtr;   /* Optional */
1081          IntegerOption          languageId;          /* Optional */
1082          };
1083
1084       union ListProfilesChoice switch(ListProfilesChoiceEnum designator) {
1085          case LIST_CHOICE_ID:
1086             CardinalSeq         profileIdSeq;
1087          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;
```

```
1145        case AVT_PERCENT:
1146           nint32   percent;
1147        };
1148
1149     struct Criteria {
1150        ObjectIdentifier     type;             /* AttributeId */
1151        CriterionThreshold   threshold;
1152        };
1153
1154
1155     /* ------ job security attributes ------ */
1156
1157     struct JobLevel {
1158        ObjectIdentifier     organizationId;
1159        nuint32              level;
1160        };
1161
1162     struct JobCategories {
1163        ObjectIdentifier     organizationId;
1164        CardinalSeq          categories;
1165        };
1166
1167
1168     /* ------ job ignored attributes ------ */
1169     /* NOTE:  Should really be :  AttributeValueSet values;
1170     **  We can not return values becuase of the
1171     **  recursive definition!
1172     */
1173
1174     union IgnoredAttributeValue switch(AVTEnum designator) {
1175        case AVT_NULL:
1176           void ;                  /* Pass nothing */
1177        case AVT_OBJECT_IDENTIFIER:
1178           ObjectIdentifier     identifier;
1179        case AVT_NAME_OR_OID:
1180           NameOrOid nameOrOid;
1181        };
1182
1183     typedef IgnoredAttributeValue IgnoredAttributeValueSet<>;
1184
1185     struct IgnoredAttribute {
1186        nuint32              documentNumberOption;
1187        ObjectIdentifier     attributeId;
1188        IgnoredAttributeValueSet    values;
1189        };
1190
1191     /* ------ default resources  ------ */
1192     enum ResourceEnum
1193     {
1194        RESOURCE_NAME_OR_OID,                  /* (0) */
1195        RESOURCE_TEXT                          /* (1) */
1196     };
1197
1198     union Resource switch(ResourceEnum designator)
1199     {
1200        case RESOURCE_NAME_OR_OID:
1201           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  ------ */
```

```
1316       enum PageOrderTypeEnum {
1317          PAGE_ORDER_UNKNOWN,                    /* (0) */
1318          PAGE_ORDER_FIRST_TO_LAST,              /* (1) */
1319          PAGE_ORDER_LAST_TO_FIRST               /* (2) */
1320          };
1321
1322
1323       /* ------ page select supported ------ */
1324
1325       enum PageIdTypeEnum {
1326          PAGE_ID_TYPE_NUMERIC,                  /* (0) */
1327          PAGE_ID_TYPE_ALPHANUMERIC,             /* (1) */
1328          PAGE_ID_TYPE_TAG                       /* (2) */
1329          };
1330
1331       /* ------ page media select ------ */
1332       enum MediaSelectEnum {
1333          MEDIA_SELECT_ALL_PAGES,              /* (0) */
1334          MEDIA_SELECT_SELECTD_PAGES          /* (1) */
1335          };
1336
1337       struct MediaSelect {
1338          PageSelect    pageRange;
1339          NameOrOid     mediumId;
1340          };
1341
1342       typedef MediaSelect MediaSelectSeq<>;
1343
1344       union PageMediaSelect switch (MediaSelectEnum designator) {
1345          case MEDIA_SELECT_ALL_PAGES:
1346             NameOrOid   mediumForAllPages;
1347          case MEDIA_SELECT_SELECTD_PAGES:
1348             MediaSelectSeq mediumForSelectedPagesSeq;
1349          };
1350
1351       /* ------ medium source size  ------ */
1352       enum MediumSizeEnum {
1353          MEDIUM_SIZE_DISCRETE,                  /* (0) */
1354          MEDIUM_SIZE_CONTINUOUS                 /* (1) */
1355          };
1356
1357       struct MediumDiscreteSizes {
1358          PageSize          pageSize;
1359          bool              longEdgeFeeds;             /* default true */
1360          Area              assuredReproductionArea;
1361          };
1362
1363       struct MediumContinuousSizes {
1364          RealRange         sizeRangeAcrossFeedDirection;
1365          nreal64           sizeIncrementAcrossFeedDir;
1366          RealRange         sizeRangeInFeedDirection;
1367          nreal64           sizeIncrementInFeedDir;
1368          bool              longEdgeFeeds;             /* default true */
1369          Area              GenericAssuredReproductionArea;
1370          };
1371
1372       union MediumSize switch (MediumSizeEnum designator) {
```

```
1373          case MEDIUM_SIZE_DISCRETE:
1374             MediumDiscreteSizes     discreteSizes;
1375          case MEDIUM_SIZE_CONTINUOUS:
1376             MediumContinuousSizes   continuousSizes;
1377          };
1378
1379       struct MediumSourceSize {
1380          NameOrOid     *inputTrayOptionPtr;
1381          MediumSize     mediumSize;
1382          };
1383
1384       /* ------ input trays medium  ------ */
1385       struct InputTrayMedium {
1386          NameOrOid      inputTray;
1387          NameOrOid      medium;
1388          };
1389
1390       /* ------ output bins characteristics ------ */
1391       enum PageOrientationEnum {
1392          PAGE_FACE_UNKNOWN,                    /* (0) */
1393          PAGE_FACE_UP,                         /* (1) */
1394          PAGE_FACE_DOWN                        /* (2) */
1395          };
1396
1397       struct OutputBinChar {
1398          PageOrderTypeEnum    stackingOrder;
1399          PageOrientationEnum  orientation;
1400          };
1401
1402       typedef OutputBinChar OutBinsCharacteristics<>;
1403
1404       /* ------ printer confidentiality level range ------ */
1405
1406       struct LevelRange {
1407          ObjectIdentifier  organizationId;
1408          nint32            minimum;
1409          nint32            maximum;
1410          };
1411
1412       /* ------ printer confidentiality category sets ------ */
1413
1414       struct CategorySet {
1415          ObjectIdentifier  organizationId;
1416          CardinalSeq       categoryMinimum;
1417          CardinalSeq       categoryMaximum;
1418          };
1419
1420       /* numbers up supported ------ */
1421       enum NumbersUpEnum {
1422          NUMBERS_UP_CARDINAL,
1423          NUMBERS_UP_NAME_OR_OID,
1424          NUMBERS_UP_CARDINAL_RANGE
1425          };
1426
1427       union NumbersUpSupported switch(NumbersUpEnum designator) {
1428          case NUMBERS_UP_CARDINAL:
1429             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) */
      };
```

```
1487       struct TrimParameters {
1488          CommonParameters common;
1489          Dimension        trimOffset;
1490          XYDimensions     trimDimensions;
1491          };
1492
1493       union TrimmingSpec switch (TrimmingEnum designator) {
1494          case TRIMMING_NAMED:
1495             NameOrOid        namedTrimming;
1496          case TRIMMING_PARAMETERS:
1497             TrimParameters   parameters;
1498          };
1499
1500       enum DieCuttingEnum {
1501          DIE_CUTTING_NAMED,                  /* (0) */
1502          DIE_CUTTING_PARAMETERS              /* (1) */
1503          };
1504
1505       struct DieCuttingParameters {
1506          CommonParameters common;
1507          XYDimensions     dieCutLocation;
1508          NameOrOid        dieCutName;
1509          };
1510
1511       union DieCuttingSpec switch (DieCuttingEnum designator) {
1512          case DIE_CUTTING_NAMED:
1513             NameOrOid              namedDieCutting;
1514          case DIE_CUTTING_PARAMETERS:
1515             DieCuttingParameters   parameters;
1516          };
1517
1518       enum PunchingEnum {
1519          PUNCHING_NAMED,                     /* (0) */
1520          PUNCHING_PARAMETERS                 /* (1) */
1521          };
1522
1523       struct PunchParameters {
1524          CommonParameters    common;
1525          Dimension          *processOffsetOptionPtr; /* dimension not area
1526       */
1527          Locations           headLocations;
1528          Dimension          *punchDiameterPtr;
1529          };
1530
1531       union PunchingSpec switch (PunchingEnum designator) {
1532          case PUNCHING_NAMED:
1533             NameOrOid namedPunching;
1534          case PUNCHING_PARAMETERS:
1535             PunchParameters parameters;
1536          };
1537
1538       enum PerforatingEnum {
1539          PERFORATING_NAMED,                  /* (0) */
1540          PERFORATING_PARAMETERS              /* (1) */
1541          };
1542
1543       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<>;
```

```
1601       struct InsertParameters {
1602          EdgeOption     referenceEdgeOption;
1603          EdgeOption     jogEdgeOption;
1604          InsertSheetSeq insertSheetList;
1605          };
1606
1607       enum InsertSpecEnum {
1608          INSERT_SPEC_NAMED,                  /* (0) */
1609          INSERT_SPEC_PARAMETERS              /* (1) */
1610          };
1611
1612       union InsertSpec switch (InsertSpecEnum designator) {
1613          case INSERT_SPEC_NAMED:
1614             NameOrOid         namedInserting;
1615          case INSERT_SPEC_PARAMETERS:
1616             InsertParameters  parameters;
1617          };
1618
1619       enum CoverSpecEnum {
1620          COVER_SPEC_NAMED,                   /* (0) */
1621          COVER_SPEC_PARAMETERS               /* (1) */
1622          };
1623
1624       struct CoverParameters {
1625          CommonParameters common;
1626          NameOrOid   *frontCoverOptionPtr;
1627          NameOrOid   *backCoverOptionPtr;
1628          };
1629
1630       union CoverSpec switch (CoverSpecEnum designator) {
1631          case COVER_SPEC_NAMED:
1632             NameOrOid        namedCovers;
1633          case COVER_SPEC_PARAMETERS:
1634             CoverParameters  parameters;
1635          };
1636
1637       enum FoldingSpecEnum {
1638          FOLDING_SPEC_NAMED,                 /* (0) */
1639          FOLDING_SPEC_PARAMETERS             /* (1) */
1640          };
1641
1642       struct FoldingParameters {
1643          CommonParameters common;
1644          Locations        headLocations;
1645          };
1646
1647       union FoldingSpec switch (FoldingSpecEnum designator) {
1648          case FOLDING_SPEC_NAMED:
1649             NameOrOid           namedFolding;
1650          case FOLDING_SPEC_PARAMETERS:
1651             FoldingParameters   parameters;
1652          };
1653
1654       struct OtherParameters {
1655          CommonParameters  common;
1656          Dimension        *processOffsetOptionPtr;
1657          Locations        *headLocationsOptionPtr;
```

```
1658        opaque            otherParams < >;
1659        };
1660
1661     struct OtherFinishingSpec {
1662        NameOrOid        finishingOpType;
1663        NameOrOid       *namedSpecOptionPtr;
1664        OtherParameters  parameters;
1665        };
1666
1667
1668     enum FinishingProcEnum {
1669        FINISHING_PROC_STITCHING,            /* (0) */
1670        FINISHING_PROC_BINDING,              /* (1) */
1671        FINISHING_PROC_TRIMMING,             /* (2) */
1672        FINISHING_PROC_DIE_CUTTING,          /* (3) */
1673        FINISHING_PROC_PUNCHING,             /* (4) */
1674        FINISHING_PROC_PERFORATING,          /* (5) */
1675        FINISHING_PROC_SLITTING,             /* (6) */
1676        FINISHING_PROC_INSERT,               /* (7) */
1677        FINISHING_PROC_COVER,                /* (8) */
1678        FINISHING_PROC_FOLDING,              /* (9) */
1679        FINISHING_PROC_OTHER                 /* (10) */
1680        };
1681
1682     union FinishingProcSpec switch (FinishingProcEnum designator) {
1683        case FINISHING_PROC_STITCHING:
1684           StitchingSpec       stitchingSpec;
1685        case FINISHING_PROC_BINDING:
1686           BindingSpec         bindingSpec;
1687        case FINISHING_PROC_TRIMMING:
1688           TrimmingSpec        trimmingSpec;
1689        case FINISHING_PROC_DIE_CUTTING:
1690           DieCuttingSpec      dieCuttingSpec;
1691        case FINISHING_PROC_PUNCHING:
1692           PunchingSpec        punchingSpec;
1693        case FINISHING_PROC_PERFORATING:
1694           PerforatingSpec     perforatingSpec;
1695        case FINISHING_PROC_SLITTING:
1696           SlittingSpec        slittingSpec;
1697        case FINISHING_PROC_INSERT:
1698           InsertSpec          insertSpec;
1699        case FINISHING_PROC_COVER:
1700           CoverSpec           coverSpec;
1701        case FINISHING_PROC_FOLDING:
1702           FoldingSpec         foldingSpec;
1703        case FINISHING_PROC_OTHER:
1704           OtherFinishingSpec  otherFinishingSpec;
1705        };
1706
1707     typedef FinishingProcSpec FinishingProcSpecSeq<>;
1708
1709     enum FinishingSpecEnum {
1710        FINISHING_SPEC_NAMED,                /* (0) */
1711        FINISHING_SPEC_LIST                  /* (1) */
1712        };
1713
1714     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;
      };
```

```
1772
1773       /*
1774       ** NOTE:
1775       **  In the following discriminant union, all values fit in 16 bytes
1776       **  or are pointed to, whether they are conformant or fixed length.
1777       **  Thus the storage for all attribute-values is 20 bytes, including
1778       the
1779       **  designator.
1780       */
1781
1782       union AttributeValue switch(AVTEnum designator) {
1783          case AVT_NULL:
1784             void ;
1785          case AVT_TEXT:
1786             Text                 text;
1787          case AVT_DESCRIPTIVE_NAME:
1788             Text                 descriptiveName;
1789          case AVT_DESCRIPTOR:
1790             Text                 descriptor;
1791          case AVT_MESSAGE:
1792             NameOrOid            message;
1793          case AVT_ERROR_MESSAGE:
1794             NameOrOid            errorMessage;
1795          case AVT_SIMPLE_NAME:
1796             Text                 simpleName;
1797          case AVT_DISTINGUISHED_NAME_STR:
1798             DistinguishedNameString distinguishedNameString;
1799          case AVT_DIST_NAME_STR_SEQ:
1800             DistinguishedNameStringSeq distinguishedNameStringSeq;
1801          case AVT_DELTA_TIME:
1802             nint32               deltaTime;
1803          case AVT_TIME:
1804             nuint32              time;
1805          case AVT_INTEGER:
1806             nint32               integer;
1807          case AVT_INTEGER_SEQ:
1808             IntegerSeq           integerSeq;
1809          case AVT_CARDINAL:
1810             nuint32              cardinal;
1811          case AVT_CARDINAL_SEQ:
1812             CardinalSeq          cardinalSeq;
1813          case AVT_POSITIVE_INTEGER:
1814             nint32               positiveInteger;
1815          case AVT_INTEGER_RANGE:
1816             IntegerRange         integerRange;
1817          case AVT_CARDINAL_RANGE:
1818             CardinalRange        cardinalRange;
1819          case AVT_MAXIMUM_INTEGER:
1820             nint32               maximumInteger;
1821          case AVT_MINIMUM_INTEGER:
1822             nint32               minimumInteger;
1823          case AVT_INTEGER_64:
1824             void;
1825          case AVT_INTEGER_64_SEQ:
1826             void;
1827          case AVT_CARDINAL_64:
1828             nuint64              cardinal64;
```

```
1829          case AVT_CARDINAL_64_SEQ:
1830             void;
1831          case AVT_POSITIVE_INTEGER_64:
1832             void;
1833          case AVT_INTEGER_64_RANGE:
1834             void;
1835          case AVT_CARDINAL_64_RANGE:
1836             void;
1837          case AVT_MAXIMUM_INTEGER_64:
1838             void;
1839          case AVT_MINIMUM_INTEGER_64:
1840             void;
1841          case AVT_REAL:
1842             nreal64              real;
1843          case AVT_REAL_SEQ:
1844             RealSet              realSeq;
1845          case AVT_NON_NEGATIVE_REAL:
1846             nreal64              nonNegativeReal;
1847          case AVT_REAL_RANGE:
1848             RealRange            realRange;
1849          case AVT_NON_NEGATIV_REAL_RANGE:
1850             RealRange            nonNegativeRealRange;
1851          case AVT_BOOLEAN:
1852             bool                 boolean;
1853          case AVT_PERCENT:
1854             nint32               percent;
1855          case AVT_OBJECT_IDENTIFIER:
1856             ObjectIdentifier     identifier;
1857          case AVT_OBJECT_IDENTIFIER_SEQ:
1858             ObjectIdentifierSet  objectIdentifierSeq;
1859          case AVT_NAME_OR_OID:
1860             NameOrOid            nameOrOid;
1861          case AVT_NAME_OR_OID_SEQ:
1862             NameOrOidSet         nameOrOidSeq;
1863          case AVT_DISTINGUISHED_NAME:
1864             Text                 distinguishedName;
1865          case AVT_RDN_SEQUENCE:
1866             RDNSequence          rdnSequence;
1867          case AVT_REALIZATION:
1868             RealizationEnum      realization;
1869          case AVT_MEDIUM_DIMENSIONS:
1870             XYRealDimensions     mediumDimensions;
1871          case AVT_DIMENSION:
1872             Dimension           *dimensionPtr;
1873          case AVT_XY_DIMENSIONS:
1874             XYDimensions        *xyDimensionsPtr;
1875          case AVT_LOCATIONS:
1876             Locations           *locationsPtr;
1877          case AVT_AREA:
1878             Area                *areaPtr;
1879          case AVT_AREA_SEQ:
1880             AreaSeq              areaSeq;
1881          case AVT_EDGE:
1882             Edge                 edge;
1883          case AVT_FONT_REFERENCE:
1884             Text                 fontReference;
1885          case AVT_CARDINAL_OR_OID:
```

```
1886            CardinalOrOid        cardinalOrOid;
1887         case AVT_OID_CARDINAL_MAP:
1888            OidCardinalMap       oidCardinalMap;
1889         case AVT_CARDINAL_OR_NAM_OR_OID:
1890            CardinalOrNameOrOid  cardinalOrNameOrOid;
1891         case AVT_POSITIVE_INT_OR_OID:
1892            PositiveIntegerOrOid positiveIntegerOrOid;
1893         case AVT_EVENT_HANDLING_PROFILE:
1894            EventHandlingProfile *eventHandlingProfilePtr;
1895         case AVT_OCTET_STRING:
1896            OctetString          octetString;
1897         case AVT_PRIORITY:
1898            nint32               jobPriority;
1899         case AVT_LOCALE:
1900            Text                 locale;
1901         case AVT_METHOD_DELIVERY_ADDR:
1902            DeliveryAddressForMethod *deliveryMethodAddressPtr;
1903         case AVT_OBJECT_IDENTIFICATION:
1904            ObjectIdentification *objectIdentificationPtr;
1905         case AVT_RESULTS_PROFILE:
1906            ResultsProfile      *resultsProfilePtr;
1907         case AVT_CRITERIA:
1908            Criteria             criteria;
1909         case AVT_JOB_PASSWORD:
1910            OctetString          jobPassword;
1911         case AVT_JOB_LEVEL:
1912            JobLevel             jobLevel;
1913         case AVT_JOB_CATEGORIES:
1914            JobCategories        jobCategories;
1915         case AVT_PRINT_CHECKPOINT:
1916            OctetString          printCheckpoint;
1917         case AVT_IGNORED_ATTRIBUTE:
1918            IgnoredAttribute    *ignoredAttributePtr;
1919         case AVT_RESOURCE:
1920            Resource             resource;
1921         case AVT_MEDIUM_SUBSTITUTION:
1922            MediumSubstitution  *mediumSubstitutionPtr;
1923         case AVT_FONT_SUBSTITUTION:
1924            FontSubstitution     fontSubstitution;
1925         case AVT_RESOURCE_CONTEXT_SEQ:
1926            ResourceContextSeq   resourceContextSeq;
1927         case AVT_SIDES:
1928            nint32               sides;
1929         case AVT_PAGE_SELECT_SEQ:
1930            PageSelectSequence   pageSelectSeq;
1931         case AVT_PAGE_MEDIA_SELECT:
1932            PageMediaSelect      pageMediaSelect;
1933         case AVT_DOCUMENT_CONTENT:
1934            DocumentContent     *documentContentPtr;
1935         case AVT_PAGE_SIZE:
1936            PageSize            *pageSizePtr;
1937         case AVT_PRESENTATION_DIRECTION:
1938            PresentationDirectionEnum presentationDirection;
1939         case AVT_PAGE_ORDER:
1940            PageOrderTypeEnum    pageOrderType;
1941         case AVT_FILE_REFERENCE:
1942            DistinguishedNameString fileReference;
```

```
1943          case AVT_MEDIUM_SOURCE_SIZE:
1944             MediumSourceSize    *mediumSourceSizePtr;
1945          case AVT_INPUT_TRAY_MEDIUM:
1946             InputTrayMedium     *inputTrayMediumPtr;
1947          case AVT_OUTPUT_BINS_CHARS:
1948             OutBinsCharacteristics outBinsCharacteristics;
1949          case AVT_PAGE_ID_TYPE:
1950             PageIdTypeEnum      pageIdType;
1951          case AVT_LEVEL_RANGE:
1952             LevelRange          levelRange;
1953          case AVT_CATEGORY_SET:
1954             CategorySet        *categorySetPtr;
1955          case AVT_NUMBERS_UP_SUPPORTED:
1956             NumbersUpSupported   numbersUpSupported;
1957          case AVT_FINISHING:
1958             Finishing           *finishingPtr;
1959          case AVT_PRT_CONTAINED_OBJ_ID:
1960             PrtContainedObjectId prtContainedObjectId;
1961          case AVT_PRT_CONFIG_OBJECT_ID:
1962             PrtConfigObjectId   *prtConfigObjectIdPtr;
1963          case AVT_TYPED_NAME:
1964             TypedName            typedName;
1965          case AVT_NET_ADDRESS:
1966             NetAddress           netAddress;
1967          case AVT_XY_DIMENSIONS_VALUE:
1968             XYDimensionsValue   *xyDimensionsValuePtr;
1969          case AVT_NAME_OR_OID_DIM_MAP:
1970             NameOrOidDimensionMap  *nameOrOidDimensionMapPtr;
1971          case AVT_PRINTER_STATE_REASON:
1972             PrinterStateReason  *printerStateReasonPtr;
1973          case AVT_ENUMERATION:
1974             nint32               enumeration;
1975           case AVT_QUALIFIED_NAME:
1976             QualifiedName        qualifiedName;
1977           case AVT_QUALIFIED_NAME_SET:
1978             QualifiedNameSet     qualifiedNameSet;
1979          case AVT_COLORANT_SET:
1980             ColorantSet         *colorantSetPtr;
1981          case AVT_RESOURCE_PRINTER_ID:
1982             ResourcePrinterId   *resourcePrinterIdPtr;
1983          case AVT_EVENT_OBJECT_ID:
1984             EventObjectId       *eventObjectIdPtr;
1985          case AVT_QUALIFIED_NAME_MAP:
1986             QualifiedNameMap    *qualifiedNameMapPtr;
1987          case AVT_FILE_PATH:
1988             Text                 filePath;
1989          };
1990
1991       typedef AttributeValue AttributeValueSet<>;
1992
1993       /*
1994       ** NOTE:
1995       **      Sending an empty sequence for values allows an attribute
1996       **      to be set as if it was not specified. This is primarily
1997       **      for use in the modify function.
1998       */
1999
```

```
   /* ------ 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;
```

```
2057        };
2058
2059     enum FilterEnum {
2060        FILTER_ITEM,                        /* (0) */
2061        FILTER_AND,                         /* (1) */
2062        FILTER_OR,                          /* (2) */
2063        FILTER_NOT                          /* (5) */
2064        };
2065
2066     union Filter switch(FilterEnum designator) {
2067        case FILTER_ITEM:
2068           FilterItem  filterItem;
2069        case FILTER_AND:
2070           struct Filter   filterAnd<>;
2071        case FILTER_OR:
2072           struct Filter   filterOr<>;
2073        case FILTER_NOT:
2074           struct Filter  *filterNotPtr;
2075        };
2076
2077     enum DpAccessRightsEnum {
2078        DP_ACCESS_RIGHTS_USER,              /* (0) */
2079        DP_ACCESS_RIGHTS_ADMIN              /* (1) */
2080        };
2081
2082     struct DpAccessListElement {
2083        Text                accessId;
2084        DpAccessRightsEnum  accessRights;
2085        };
2086
2087     struct AccessAndAccounting {
2088        Text              userName;
2089        OctetString       accountingInformation;
2090        };
2091
2092     /*
2093     ** NOTE:
2094     **   Define all error information to be returned in a
2095     **   single structure
2096     **   with a primary error code (error type)
2097     **
2098     **   Error messages are optional.
2099     */
2100     enum ProblemTypeEnum {
2101        PROBLEM_TYPE_STANDARD,                /* (0) */
2102        PROBLEM_TYPE_EXTENDED                 /* (1) */
2103        };
2104
2105     /*********************************************
2106      **          Access Error / Problem          **
2107      *********************************************/
2108
2109     enum AccessProblemEnum {
2110        ACCESS_WRONG_OBJECT_CLASS,            /* (0) */
2111        ACCESS_LACK_ACCESS_RIGHTS,            /* (1) */
2112        ACCESS_CANT_INTERRUPT_JOB,            /* (2) */
2113        ACCESS_WRONG_OBJECT_STATE,            /* (3) */
```

```
2114          ACCESS_CLIENT_NOT_BOUND,              /* (4) */
2115          ACCESS_NOT_AVAILABLE,                 /* (5) */
2116          ACCESS_NOTIF_SVC_NOT_BOUND            /* (6) */
2117          };
2118
2119       union AccessProblem switch(ProblemTypeEnum designator)
2120       {
2121          case PROBLEM_TYPE_STANDARD:
2122             AccessProblemEnum standardProblem;
2123          case PROBLEM_TYPE_EXTENDED:
2124             ObjectIdentifier  extendedProblem;
2125       };
2126
2127       struct AccessError
2128       {
2129          AccessProblem           problem;
2130          ObjectIdentification    objectIdentification;
2131          NameOrOid              *messageOptionPtr;
2132       };
2133
2134       /*********************************************
2135        **          Attribute Error / Problem        **
2136        *********************************************/
2137
2138       enum AttributeProblemEnum {
2139          ATTR_INVALID_SYNTAX,                  /* (0)  */
2140          ATTR_UNDEFINED_TYPE,                  /* (1)  */
2141          ATTR_WRONG_MATCHING,                  /* (2)  */
2142          ATTR_CONSTRAINT_VIOLATED,             /* (3)  */
2143          ATTR_UNSUPPORTED_TYPE,                /* (4)  */
2144          ATTR_ILLEGAL_MODIFICATION,            /* (5)  */
2145          ATTR_CONSIST_W_OTHER_ATTR,            /* (6)  */
2146          ATTR_UNDEFINED_ATTR_VALUE,            /* (7)  */
2147          ATTR_UNSUPPORTED_VALUE,               /* (8) */
2148          ATTR_INVALID_NONCOMPUL_MOD,           /* (9) */
2149          ATTR_PER_JOB_INADMISSIBLE,            /* (10) */
2150          ATTR_NOT_MULTI_VALUED,                /* (11) */
2151          ATTR_MANDATORY_OMITTED,               /* (12) */
2152          ATTR_ILLEGAL_FOR_CLASS                /* (13) */
2153          };
2154
2155       union AttributeProblem switch(ProblemTypeEnum designator)
2156       {
2157          case PROBLEM_TYPE_STANDARD:
2158             AttributeProblemEnum standardProblem;
2159          case PROBLEM_TYPE_EXTENDED:
2160             ObjectIdentifier     extendedProblem;
2161       };
2162
2163       struct AttributeProblemItem
2164       {
2165          AttributeProblem  problem;
2166          Attribute         attribute;
2167          NameOrOid        *messageOptionPtr;
2168       };
2169
2170       struct AttributeError
```

```
2171       {
2172          AttributeProblemItem    problems<>;
2173          ObjectIdentification    objectIdentification;
2174       };
2175
2176       /*********************************************
2177        **          Document Access Error / Problem  **
2178        *********************************************/
2179
2180       enum DocAccessProblemEnum {
2181          DOC_ACCESS_NOT_AVAILABLE,              /* (0) */
2182          DOC_ACCESS_FIDELI_TIME_EXP,            /* (1) */
2183          DOC_ACCESS_DENIED,                     /* (2) */
2184          DOC_ACCESS_UNKNOWN_DOC,                /* (3) */
2185          DOC_ACCESS_NO_DOCS_IN_JOB              /* (4) */
2186          };
2187
2188       union DocAccessProblem switch(ProblemTypeEnum designator)
2189       {
2190          case PROBLEM_TYPE_STANDARD:
2191             DocAccessProblemEnum standardProblem;
2192          case PROBLEM_TYPE_EXTENDED:
2193             ObjectIdentifier     extendedProblem;
2194       };
2195
2196       struct DocAccessError
2197       {
2198          DocAccessProblem        problem;
2199          ObjectIdentification    objectIdentification;
2200          NameOrOid              *messageOptionPtr;
2201       };
2202
2203       /*********************************************
2204        **          Printer Error / Problem         **
2205        *********************************************/
2206
2207       enum PrinterProblemEnum {
2208          PRINTER_ERROR,                         /* (0) */
2209          PRINTER_NEEDS_ATTENTION,               /* (1) */
2210          PRINTER_NEEDS_KEY_OPERATOR             /* (2) */
2211          };
2212
2213       union PrinterProblem switch(ProblemTypeEnum designator)
2214       {
2215          case PROBLEM_TYPE_STANDARD:
2216             PrinterProblemEnum   standardProblem;
2217          case PROBLEM_TYPE_EXTENDED:
2218             ObjectIdentifier     extendedProblem;
2219       };
2220
2221       struct PrinterError
2222       {
2223          PrinterProblem          problem;
2224          ObjectIdentification    objectIdentification;
2225          NameOrOid              *messageOptionPtr;
2226       };
2227
```

```
2228       /*******************************************
2229        **      Security Access Error / Problem     **
2230        *******************************************/
2231
2232       enum SecurityProblemEnum {
2233          SECURITY_AUTHENTICATION,              /* (0) */
2234          SECURITY_CREDENTIALS,                 /* (1) */
2235          SECURITY_RIGHTS,                      /* (2) */
2236          SECURITY_INVALID_PAC                  /* (3) */
2237          };
2238
2239       union SecurityProblem switch(ProblemTypeEnum designator)
2240       {
2241          case PROBLEM_TYPE_STANDARD:
2242             SecurityProblemEnum  standardProblem;
2243          case PROBLEM_TYPE_EXTENDED:
2244             ObjectIdentifier     extendedProblem;
2245       };
2246
2247       struct SecurityError {
2248          SecurityProblem      problem;
2249          NameOrOid           *messageOptionPtr;
2250          };
2251
2252       /*******************************************
2253        **          Selection Error / Problem       **
2254        *******************************************/
2255
2256       enum SelectionProblemEnum {
2257          SELECTION_INVALID_ID,                 /* (0) */
2258          SELECTION_UNKNOWN_ID,                 /* (1) */
2259          SELECTION_OBJECT_EXISTS               /* (2) */
2260          };
2261
2262       union SelectionProblem switch(ProblemTypeEnum designator)
2263       {
2264          case PROBLEM_TYPE_STANDARD:
2265             SelectionProblemEnum standardProblem;
2266          case PROBLEM_TYPE_EXTENDED:
2267             ObjectIdentifier     extendedProblem;
2268       };
2269
2270       struct SelectionError
2271       {
2272          SelectionProblem        problem;
2273          ObjectIdentification    objectIdentification;
2274          Attribute              *attributeOptionPtr;
2275          NameOrOid              *messageOptionPtr;
2276       };
2277
2278       /*******************************************
2279        **          Service Error / Problem         **
2280        *******************************************/
2281
2282       enum ServiceProblemEnum {
2283          SERVICE_SERVER_BUSY,                  /* (0)  */
2284          SERVICE_SERVER_UNAVAILABLE,           /* (1)  */
```

```
2285          SERVICE_OPERATION_COMPLEX,             /* (2)  */
2286          SERVICE_RESOURCE_LIMIT,                /* (3)  */
2287          SERVICE_UNCLASS_SERVER_ERR,            /* (4)  */
2288          SERVICE_TOO_MANY_ITEM_LIST,            /* (5)  */
2289          SERVICE_RESOURCE_NOT_AVAIL,            /* (6)  */
2290          SERVICE_CANCEL_DOC_SUPPORT,            /* (7)  */
2291          SERVICE_MODIFY_DOC_SUPPORT,            /* (8)  */
2292          SERVICE_MULTI_DOC_SUPPORT,             /* (9)  */
2293          SERVICE_PARM_VAL_SUPPORT,              /* (10) */
2294          SERVICE_INVALID_CHECKPOINT,            /* (11) */
2295          SERVICE_CONTINUATN_CONTEXT,            /* (12) */
2296          SERVICE_PAUSE_LIMIT_EXCEED,            /* (13) */
2297          SERVICE_UNSUPPORTED_OP,                /* (14) */
2298          SERVICE_NOTIF_SERVICE_ERR              /* (15) */
2299          };
2300
2301       union ServiceProblem switch(ProblemTypeEnum designator)
2302       {
2303          case PROBLEM_TYPE_STANDARD:
2304             ServiceProblemEnum   standardProblem;
2305          case PROBLEM_TYPE_EXTENDED:
2306             ObjectIdentifier     extendedProblem;
2307       };
2308
2309       struct ServiceError
2310       {
2311          ServiceProblem          problem;
2312          ObjectIdentification    objectIdentification;
2313          Attribute              *attributeOptionPtr;
2314          nuint32                 libError;
2315          nuint32                 otherError;
2316          nuint32                 otherError2;
2317          NameOrOid              *messageOptionPtr;
2318       };
2319
2320       /*******************************************
2321        **          Update Error / Problem         **
2322        *******************************************/
2323
2324       enum UpdateProblemEnum {
2325          UPDATE_NO_MODS_ALLOWED,                /* (0) */
2326          UPDATE_INSUFFICIENT_RIGHTS,            /* (1) */
2327          UPDATE_PREV_OP_INCOMPLETE,             /* (2) */
2328          UPDATE_CANCEL_NOT_POSSIBLE             /* (3) */
2329          };
2330
2331       union UpdateProblem switch(ProblemTypeEnum designator)
2332       {
2333          case PROBLEM_TYPE_STANDARD:
2334             UpdateProblemEnum standardProblem;
2335          case PROBLEM_TYPE_EXTENDED:
2336             ObjectIdentifier  extendedProblem;
2337       };
2338
2339       struct UpdateError
2340       {
2341          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                                 */
```

```
2504          AVT_MEDIUM_SOURCE_SIZE,/* 80 struct              reference   */
2505          AVT_INPUT_TRAY_MEDIUM, /* 81 struct              reference   */
2506          AVT_OUTPUT_BINS_CHARS, /* 82 struct              value       */
2507          AVT_PAGE_ID_TYPE,      /* 83 integer             value       */
2508          AVT_LEVEL_RANGE,       /* 84 struct              reference   */
2509          AVT_CATEGORY_SET,      /* 85 struct              reference   */
2510          AVT_NUMBERS_UP_SUPPORTED,
2511                                 /* 86 struct              reference   */
2512          AVT_FINISHING,         /* 87 struct              reference   */
2513          AVT_PRT_CONTAINED_OBJ_ID,
2514                                 /* 88 struct              reference   */
2515          AVT_PRT_CONFIG_OBJECT_ID,
2516                                 /* 89 struct              reference   */
2517          AVT_TYPED_NAME,        /* 90 struct              reference   */
2518          AVT_NET_ADDRESS,       /* 91 struct              reference   */
2519          AVT_XY_DIMENSIONS_VALUE,
2520                                 /* 92 struct              reference   */
2521          AVT_NAME_OR_OID_DIM_MAP,
2522                                 /* 93 struct              reference   */
2523          AVT_PRINTER_STATE_REASON,
2524                                 /* 94 struct              value       */
2525          AVT_ENUMERATION,       /* 95 integer             value       */
2526          AVT_QUALIFIED_NAME,    /* 96 struct              reference   */
2527          AVT_QUALIFIED_NAME_SET,/* 97 struct              reference   */
2528          AVT_COLORANT_SET,      /* 98 struct              reference   */
2529          AVT_RESOURCE_PRINTER_ID,
2530                                 /* 99 struct              value       */
2531          AVT_EVENT_OBJECT_ID,   /* 100 struct             reference   */
2532          AVT_QUALIFIED_NAME_MAP,/* 101 struct             reference   */
2533          AVT_FILE_PATH          /* 102 struct             value       */
2534          };
2535
2536       /* ***************** end of ldpaav.x  ******************** */
2537
2538
```