Appendix OID

Z39.50 Object Identifiers

(Normative)

OID.1 Object Identifier Assigned to this Standard

The following object identifier has been assigned, by ANSI, to this standard, ANSI-standard-Z39.50:

{iso (1) member-body (2) US (840) ANSI-standard-Z39.50 (10003)}

OID.2 Object Classes Assigned by this Standard

This standard assigns the following values for object classes, at the level immediately subordinate to ANSI-standard-Z39.50:

1 = application context definitions
2 = abstract syntax definition for APDUs
3 = attribute set definitions
4 = diagnostic definitions
5 = record syntax definitions
6 = transfer syntax definitions
7 = resource report format definitions
8 = access control format definitions
9 = extended services definitions
10 = user information format definitions
11 = element specification format definitions
12 = variant set definitions
13 = database schema definitions
14 = tag set definitions

The following ASN.1 module establishes shorthand notation for the Z39.50 object identifier, and for the object classes. The notation is used in appendices that follow.

```
ANSI-Z39-50-3-ObjectIdentifier  DEFINITIONS ::=
BEGIN
Z39-50  OBJECT IDENTIFIER ::=
{ iso (1) member-body (2) US (840) ANSI-standard-Z39.50 (10003)}

Z39-50-context           OBJECT IDENTIFIER ::= {Z39-50 1}    -- See Appendix CTX.
Z39-50-APDU              OBJECT IDENTIFIER ::= {Z39-50 2}    -- See 4.1 and note below.
Z39-50-attributeSet      OBJECT IDENTIFIER ::= {Z39-50 3}    -- See Appendix ATR.
Z39-50-diagnostic        OBJECT IDENTIFIER ::= {Z39-50 4}    -- See Appendix ERR.
Z39-50-recordSyntax      OBJECT IDENTIFIER ::= {Z39-50 5}    -- See Appendix REC.
Z39-50-transferSyntax    OBJECT IDENTIFIER ::= {Z39-50 6}    -- See note below.
Z39-50-resourceReport    OBJECT IDENTIFIER ::= {Z39-50 7}    -- See Appendix RSC.
```

```
Z39-50-accessControl        OBJECT IDENTIFIER ::= {Z39-50 8}     -- See Appendix ACC.
Z39-50-extendedService      OBJECT IDENTIFIER ::= {Z39-50 9}     -- See Appendix EXT.
Z39-50-userInfoFormat       OBJECT IDENTIFIER ::= {Z39-50 10}    -- See Appendix USR.
Z39-50-elementSpec          OBJECT IDENTIFIER ::= {Z39-50 11}    -- See Appendix ESP.
Z39-50-variantSet           OBJECT IDENTIFIER ::= {Z39-50 12}    -- See Appendix VAR.
Z39-50-schema               OBJECT IDENTIFIER ::= {Z39-50 13}    -- See Appendix SCH.
Z39-50-tagSet               OBJECT IDENTIFIER ::= {Z39-50 14}    -- See Appendix SCH.
END
```

Note:

No object identifier is assigned by this standard for any transfer syntax. For APDUs, as well as for any other abstract syntaxes described using ASN.1, the transfer syntax results from the application of the ASN.1 Basic Encoding Rules (ISO 8825). For the purposes of Presentation Context negotiation, this syntax is identified by a pair of object identifiers, one for the abstract-syntax (Z39.50-apdus) and one for the basic encoding rules:

{ joint-iso-ccitt asn1 (1) basic-encoding (1) }

OID.3 Object Identifiers Assigned by this Standard

All object identifiers assigned by this standard are explicitly assigned in the appendices that follow.

OID.4 Object Identifiers Used by this Standard

There are four types of Object Identifiers used by Z39.50:

(a) Object identifiers assigned by this standard. These are assigned in the appendices that follow.

(b) Standard, public object identifiers assigned by the Z39.50 Maintenance Agency. See OID.5.

(c) Locally defined public object identifiers. See OID.6.

(d) Locally defined private object identifiers. See OID.6.

OID.5 Object Identifiers Assigned by the Z39.50 Maintenance Agency

Additional Object Identifiers may be assigned by the Z39.50 Maintenance Agency (see note), of the form: {Z39-50 n m} where {z39-50 n} is an object class defined in OID.2, or is an additional object class defined by the maintenance agency.

Note: At the time of approval of this standard, the Z39.50 Maintenance Agency is the Library of Congress.

OID.6 Locally Registered Objects

Locally registered objects are of the form {Z39-50 n 1000 p m} where {z39-50 n} is as described in OID.5, and 'p' is the OID index of a registered Z39.50 Implementor (contact the Z39.50 Maintenance Agency for procedures for registration of an implementor). A locally registered object may be 'public' or 'private'. Local, public objects are those whose definitions are coordinated with and published by the Z39.50 Maintenance Agency. Local, private objects are those not published by the Z39.50 Maintenance Agency.

Appendix CTX

Application Context basic-Z39.50-ac

(Normative)

This standard defines and registers the application context basic-Z39.50-ac. The object identifier for application context basic-Z39.50-ac is:

{ Z39-50-context basic-Z39.50-ac (1) }

<u>Definition of application context basic-Z39.50-ac</u>
ANSI-standard-Z39.50 application context basic-Z39.50-ac supports an application-entity that contains only the following two application service elements (ASEs):

a) the association control service element (ACSE, ISO 8650), and
b) the Z39.50 service element.
   Z39.50 and ACSE are used according to the procedures in section 4.2.1.
   The P-Data service is used to transmit Z39.50 APDUs.
   In the event of protocol errors, the system detecting the error shall abort the association.

Appendix ATR

Attribute Sets

(Normative)

This standard registers the attribute sets listed below, and assigns the following object identifiers:

Bib-1	{Z39-50-attributeSet	**1**}	(See ATR.1)
Exp-1	{Z39-50-attributeSet	**2**}	(See ATR.2)
Ext-1	{Z39-50-attributeSet	**3**}	(See ATR.3)
CCL-1	{Z39-50-attributeSet	**4**}	
GILS	{Z39-50-attributeSet	**5**}	

ATR.1 Attribute Set bib-1

This section defines the attribute-set bib-1.

In the Z39.50 ASN.1 definitions, when the value of AttributeSetId, accompanying AttributeList, equals the object identifier for this attribute-set, then: AttributeType (within AttributeElement within AttributeList) takes values from the attribute types listed below, and AttributeValue takes values from the corresponding attribute table (1 through 6).

A bib-1 attribute list may include zero or more attributes for each attribute type. Some attribute combinations (in particular, when no attribute or more than one attribute are included for a given type) may result in unpredictable behavior by a given server.

Attribute Types

Attribute Type	Value	Attribute Type	Value
Use	1	Structure	4
Relation	2	Truncation	5
Position	3	Completeness	6

Table 1: Use Attributes

Use	Value	Use	Value
Personal name	1	NAL call number	18
Corporate name	2	MOS call number	19
Conference name	3	Local classification	20
Title	4	Subject heading	21
Title series	5	Subject Rameau	22
Title uniform	6	BDI index subject	23
ISBN	7	INSPEC subject	24
ISSN	8	MESH subject	25
LC card number	9	PA subject	26
BNB card no.	10	LC subject heading	27
BGF number	11	RVM subject heading	28
Local number	12	Local subject index	29
Dewey classification	13	Date	30
UDC classification	14	Date of publication	31
Bliss classification	15	Date of acquisition	32
LC call number	16	Title key	33
NLM call number	17	Title collective	34

Table 1: Use Attributes (continued)

Use	Value	Use	Value
Title parallel	35	Author	1003
Title cover	36	Author-name personal	1004
Title added title page	37	Author-name corporate	1005
Title caption	38	Author-name conference	1006
Title running	39	Identifier -- standard	1007
Title spine	40	Subject -- LC children's	1008
Title other variant	41	Subject name -- personal	1009
Title former	42	Body of text	1010
Title abbreviated	43	Date/time added to database	1011
Title expanded	44	Date/time last modified	1012
Subject precis	45	Authority/format identifier	1013
Subject rswk	46	Concept-text	1014
Subject subdivision	47	Concept-reference	1015
Number national bibliography	48	Any	1016
Number legal deposit	49	Server-choice	1017
Number govt publication	50	Publisher	1018
Number publisher for music	51	Record-source	1019
Number db	52	Editor	1020
Number local call	53	Bib-level	1021
Code -- language	54	Geographic-class	1022
Code -- geographic area	55	Indexed-by	1023
Code -- institution	56	Map-scale	1024
Name and title	57	Music-key	1025
Name geographic	58	Related-periodical	1026
Place publication	59	Report-number	1027
CODEN	60	Stock-number	1028
Microform generation	61	Thematic-number	1030
Abstract	62	Material-type	1031
Note	63	Doc-id	1032
Author-title	1000	Host-item	1033
Record type	1001	Content-type	1034
Name	1002		

Table 2: Relation Attributes

Relation	Value	Relation	Value
less than	1	greater than	5
less than or equal	2	not equal	6
equal	3	phonetic	100
greater or equal	4	stem	101
		relevance	102

Notes:

(1) "term" is the right operand of the relation. For example, if 'Use' is 'date', 'relation' is 'less than', and 'term' is '1800', then the relationship is "date less than 1800".

(2) "stem" refers to a linguistic match; "phonetic" refers to a soundex or some similar type of match based on aural similarity. For example stemming might match 'women' and 'woman', but not 'night' and 'knight'; 'woman' and 'women' would not match phonetically but 'knight' and 'night' would.

Table 3: Position Attributes

Position	Value	Position	Value
first in field	1	any position in field	3
first in subfield	2		

Table 4: Structure Attribute

Structure	**Value**	**Structure**	**Value**
Phrase	1	name (normalized)	101
word	2	name (un-normalized)	102
key	3	structure	103
year	4	urx	104
date (normalized)	5	free-form-text	105
word list	6	document-text	106
date (un-normalized)	100	local number	107

Table 5: Truncation Attributes

Truncation	**Value**	**Truncation**	**Value**
Right Truncation	1	Do not truncate	100
Left truncation	2	Process # in the search term	101
Left and right	3	Glob	102
		Regexp	103

Notes:

(1) A Glob regular expression is a string of zero or more literal or special characters. Scanning the term from left to right, all characters are interpreted as literal except when one of the following is encountered:

- * Matches any string of zero or more characters.
- ? Matches any single character.
- [When '[' is encountered, followed by a sequence of zero or more characters, followed by ']', the entire string (left and right square bracket and the enclosed sequence of characters) matches any single character from the sequence; unless the sequence begins with '^', in which case it matches any single character not from the sequence. If two characters in the sequence are separated by '-', this is shorthand for the full list of ASCII characters between them, inclusive (for example '[0-9]' matches any decimal digit). ']' is interpreted as a literal in the sequence if it is the first character (following a possible '^'). '-' is interpreted as a literal if it is the first or last character.
- { When '{' is encountered, followed by a sequence of zero or more regular expressions separated by commas, followed by '}', the entire string (left and right curly bracket and the enclosed sequence of regular expressions) matches anything that matches one of the expressions from the sequence. For example, "a{bc,de}f matches "abcf" or "adef" (as opposed to "a[bc,de]f" which matched "abf", "acf", "a,f", "adf", or "aef").

(2) A Regexp regular expression is zero or more *branches*, separated by '|'. It matches anything that matches one of the branches. A branch is one or more *pieces*, concatenated. It matches a match for the first followed by a match for the second, etc. A piece is an *atom* possibly followed by one of the following:

- * an atom followed by '*' matches a sequence of 0 or more matches of the atom
- + an atom followed by '+' matches a sequence of 1 or more matches of the atom
- ? an atom followed by '?' matches a match of the atom, or the null string.

An atom is one of the following:

Atom	Matches
regular expression enclosed in parenthesis	the regular expression
sequence enclosed in square brackets	as described for Glob
'.'	any single character
'^'	null at beginning
'$'	null at end
'\' followed by a single character	the single character
single character	the character

Table 6: Completeness Attributes

Completeness	**Value**	**Completeness**	**Value**
incomplete subfield	1	complete field	3
complete subfield	2		

ATR.2 Attribute Set exp-1

This section defines the attribute-set exp-1, for use with an Explain database.

Attribute Types

Attribute Type	Value
Use	1

Table 7: Use Attributes

Use	**Value**	**Use**	**Value**
ExplainCategory	1	HumanStringLanguage	9
HumanStringFormat	2	Producer	10
Database-Name	3	Supplier	11
Alternate-DB-names	4	Element-name	12
Target-name	5	Element-set-name	13
Alternate-target-name	6	Display-name	14
Attribute-Set-Name	7	Table-name	15
Record-Syntax	8	Term-list	16
		Schema	17

Notes:

(1) Terms for ExplainCategory are listed in table 8.

(2) Terms for HumanStringFormat are listed in table 9.

(3) Term for HumanStringLanguage is the three-character language code from "USMARC Code List for Languages"

Table 8: Search terms associated with use attribute "ExplainCategory"

General-Target-Info	Record-Syntax-Info
List-of-Databases	Schema-info
Target-IR-Parameters	Record-Element-Info
General-Database-Info	Element-Details
Database-IR-Parameters	Display-Language
Database-Attributes-Info	Term-List
Attribute-details	Category-list

Table 9: Search terms associated with use attribute "HumanString-Format"

PlainText	ODIF
PostScript	SGML

ATR.3 Attribute Set ext-1

This section defines the attribute-set ext-1, for use with an Extended Services database.

Attribute Type	Value
Use	1
Permissions	2

Table 10: Use Attributes

Use	**Value**	**Use**	**Value**
UserId	1	TaskStatus	4
PackageName	2	PackageType	5
CreationDatetime	3	RetentionTime	6

Table 11: Permission Attributes

The Permission attribute is for use only when the value of the Use attribute is Userid, in which case the purpose is to search for task packages for which the specified user has the specified permission.

Permission	**Value**	**Use**	**Value**
Delete	1	Present	4
Modify	2	Invoke	5
ModifyPermissions	3	Any	6

Additional attributes may be defined within a specific Extended Service definition. The attribute set id to be used to identify those attributes should be the ObjectIdentifier which identifies the specific Extended Service.

Appendix ERR

Error Diagnostics

(Normative)

This standard defines and registers the diagnostic set bib-1 and the diagnostic format diag-1. The following object identifiers are assigned:

bib-1	{Z39-50-diagnostic 1}	(See ERR.1)
diag-1	{Z39-50-diagnostic 2}	(See ERR.2)

When version 2 is in effect, a diagnostic record must conform to the following format:

```
DefaultDiagFormat ::= SEQUENCE{
        diagnosticSetId         OBJECT IDENTIFIER,
        condition               INTEGER,
        addinfo                 VisibleString}
```

The condition described by the diagnostic record is given by an integer, qualified by an OBJECT IDENTIFIER, the "diagnostic set id".

In version 3 a diagnostic record may assume the form above, or alternately, may take an EXTERNALly defined form, identified by an OBJECT IDENTIFIER (which identifies the diagnostic format, as opposed to the diagnostic set).

Bib-1 is a diagnostic set. It was defined and registered in Z39.50-1992. The conditions listed in Z39.50-1994 for bib-1 include all those which were listed in bib-1 in Z39.50-1992, as well as several additional diagnostics which have been added. (In particular, several of the conditions described by diag-1 that can be expressed by the above format have been added to bib-1.)

Diag-1 is a diagnostic format. It includes several structures for diagnostic information, each tailored to the error information being described. It also includes a single structure through which a diagnostic from a diagnostic set (e.g. bib-1) can be referenced.

Diag-1 allows several diagnostic conditions within a single diagnostic record, to describe multiple errors pertaining to the same record or operation. In particular, diagnostics from different diagnostic sets may be included within the same diag-1 diagnostic record.

ERR.1 Diagnostic Set Bib-1

This section defines the diagnostic set bib-1.

The table below is for use when DiagnosticSetId (within DefaultDiagFormat) equals the object identifier for this diagnostic set, in which case, Condition (within OldDiagRec) takes values from the "Code" column below.

This table may also be used by diagnostic format diag-1 when v2DiagRec is selected for "diagnostic", and DiagnosticSetId equals the object identifier for this diagnostic set. In that case, the values of "code" and "addinfo" would be taken from this table.

code	Meaning	Addinfo	Type
1	permanent system error	(unspecified)	1
2	temporary system error	(unspecified)	1
3	unsupported search	(unspecified)	2
4	Terms only exclusion (stop) words	(unspecified)	2
5	Too many argument words	(unspecified)	2

6	Too many boolean operators	(unspecified)	2
7	Too many truncated words	(unspecified)	2
8	Too many incomplete subfields	(unspecified)	2
9	Truncated words too short	(unspecified)	2
10	Invalid format for rec. number (search term)	(unspecified)	2
11	Too many characters in search statement	(unspecified)	2
12	Too many records retrieved	(unspecified)	2
13	Present request out-of-range	(unspecified)	3
14	System error in presenting records	(unspecified)	4
15	Record not authorized to be sent intersystem	(unspecified)	4
16	Record exceeds Preferred-message-size	(unspecified)	4
17	Record exceeds Maximum-record-size	(unspecified)	4
18	Result set not supported as a search term	(unspecified)	2
19	Only <u>single</u> result set as search term supported	(unspecified)	2
20	Only <u>AND</u>ing of a <u>single</u> result set as search term	(unspecified)	2
21	Result set exists and replace indicator off	(unspecified)	2
22	Result set naming not supported	(unspecified)	2
23	Specified combination of databasess not supported	(unspecified)	2
24	Element set names not supported	(unspecified)	1
25	Specified element set name not valid for specified database	(unspecified)	1
26	Only generic form of element set name supported	(unspecified)	1
27	Result set no longer exists - unilaterally deleted by target	(unspecified)	1
28	Result set is in use	(unspecified)	1
29	One of the specified databases is locked	(unspecified)	1
30	Specified result set does not exist	(unspecified)	1
31	Resources exhausted - no results available	(unspecified)	2
32	Resources exhausted - unpred. partial results available	(unspecified)	2
33	Resources exhausted - valid subset of results available	(unspecified)	2
100	(unspecified) error	(unspecified)	1
101	Access-control failure	(unspecified)	1
102	Challenge required, could not be issued - op. terminated	(unspecified)	1
103	Challenge required, could not be issued - rec. not included	(unspecified)	4
104	Challenge failed - record not included	(unspecified)	4
105	Terminated at origin request	(unspecified)	1
106	No abstract syntaxes agreed to for this record	(unspecified)	4
107	Query type not supported	(unspecified)	2
108	Malformed query	(unspecified)	2
109	Database unavailable	database name	2
110	Operator unsupported	operator	2
111	Too many databases specified	maximum	2
112	Too many result sets created	maximum	2
113	Unsupported attribute type	type	2
114	Unsupported Use attribute	value	2
115	Unsupported value for Use attribute	term	2
116	Use attribute required but not supplied	(unspecified)	2
117	Unsupported Relation attribute	value	2
118	Unsupported Structure attribute	value	2
119	Unsupported Position attribute	value	2
120	Unsupported Truncation attribute	value	2

121	Unsupported Attribute Set	oid	2
122	Unsupported Completeness attribute	value	2
123	Unsupported attribute combination	(unspecified)	2
124	Unsupported coded value for term	value	2
125	Malformed search term	(unspecified)	2
126	Illegal term value for attribute	term	2
127	Unparsable format for un-normalized value	value	2
128	Illegal result set name	name	1
129	Proximity search of sets not supported	(unspecified)	2
130	Illegal result set in proximity search	result set name	2
131	Unsupported proximity relation	value	2
132	Unsupported proximity unit code	value	2
201	Proximity not supported with this attribute combination	attribute list	2
202	Unsupported distance for proximity	distance	2
203	Ordered flag not supported for proximity	(unspecified)	2
204	Exclusion flag not supported for proximity	(unspecified)	2
205	Only zero step size supported for Scan	(unspecified)	5
206	Specified step size not supported for Scan	step size	5
207	Cannot sort according to sequence	sequence	5
208	No result set name supplied on Sort	(unspecified)	5
209	Generic sort not supported (db specific only)	(unspecified)	5
210	Db specific sort not supported	(unspecified)	5
211	Too many sort keys	number	5
212	Duplicate sort keys	key	5
213	Unuspported missing data action	value	5
214	Illegal sort relation	relation	5
215	Illegal case value	value	5
216	Illegal missing data action	value	5
217	Cannot guarantee records will fit in specified segments	(unspecified)	3
218	ES: Package name already in use	name	5
219	ES: no such package, on modify/delete	name	5
220	ES: quota exceeded	(unspecified)	5
221	ES: extended service type not supported	type	5
222	ES: permission denied on ES - id not authorized	(unspecified)	5
223	ES: permission denied on ES - cannot modify or delete	(unspecified)	5
224	ES: immediate execution failed	(unspecified)	5
225	ES: immediate execution not supported for this service	(unspecified)	5
226	ES: immediate execution not supported for these parameters	(unspecified)	5
227	No data available in requested record syntax	(unspecified)	4
228	Scan: malformed scan	(unspecified)	5
229	Term type not supported	type	1
230	Sort: too many input results	max	5
231	Sort: incompatible record formats	(unspecified)	5

"Type" is as follows:

(1) May occur when search-status or present-status = "failure".
(2) May occur only when search-status = "failure".
(3) May occur only when present-status = "failure".
(4) May occur only as a surrogate for a database record.
(5) Applies to a service other than Search or Present, and may occur only as a non-surrogate diagnostic.

ERR.2 Diagnostic Format Diag-1

This section defines the diagnostic format diag-1.

```
DiagnosticFormatDiag1
{Z39-50-diagnosticFormat diag-1 (2)} DEFINITIONS ::=
BEGIN
IMPORTS Term, Specification, AttributeList, SortElement, DatabaseName, SortElement
FROM Z39-50-APDU;
DiagnosticFormat ::=  SEQUENCE OF SEQUENCE{
   diagnostic  [1] CHOICE{
                  v2DiagRec             [1] IMPLICIT SEQUENCE{
                                             diagnosticSetId   OBJECT IDENTIFIER,
                                                                   -- A diagnostic set. Could be bib-1.
                                             condition         INTEGER, -- A code from the set.
                                             addInfo           VisibleString},
                  explicitDiagnostic    [2] DiagFormat},
   addMsg     [2] IMPLICIT GeneralString OPTIONAL}

DiagFormat ::= CHOICE{
tooMany    [1000]   IMPLICIT SEQUENCE{
                        tooManyWhat    [1]   IMPLICIT INTEGER{
                                                argumentWords          (1),
                                                truncatedWords         (2),
                                                booleanOperators       (3),
                                                incompleteSubfields    (4),
                                                characters             (5),
                                                recordsRetrieved       (6),
                                                dataBasesSpecified     (7),
                                                resultSetsCreated      (8),
                                                indexTermsProcessed    (9)},
                        max            [2]   IMPLICIT INTEGER OPTIONAL},
--
badSpec    [1001]   IMPLICIT SEQUENCE{  -- element set name or specification
                        spec       [1] IMPLICIT Specification, -- esn or element spec not supported
                        db         [2] IMPLICIT DatabaseName OPTIONAL,
                                      -- if db specified, above spec not supported for db; otherwise,
                                      -- spec not supported period.
                        goodOnes   [3] IMPLICIT SEQUENCE OF Specification OPTIONAL
                                      -- target supplies ones that are supported
                                      },
dbUnavail  [1002]   IMPLICIT SEQUENCE{ -- database unavailable
                        db    [1] IMPLICIT DatabaseName,
                        why   [2] IMPLICIT SEQUENCE{
                                 reasonCode    [1] IMPLICIT INTEGER{
                                                      doesNotExist         (0),
                                                      existsButUnavail     (1),
                                                      locked               (2),
                                                      accessDenied         (3)} OPTIONAL,
                                 message       [2] IMPLICIT VisibleString OPTIONAL}},
```

```
unSupOp   [1003]  IMPLICIT INTEGER{  -- unsupported operator
                      and        (0),
                      or         (1),
                      and-not    (2),
                      prox       (3)},
--
attribute  [1004]  IMPLICIT SEQUENCE{
                         -- applies for unsupported attribute set, attribute type,
                         -- attribute value, or term (for a given attribute type or value).
                      id        [1]   IMPLICIT OBJECT IDENTIFIER,
                         -- if only "id" occurs, then attribute set is not supported.
                      type      [2]   IMPLICIT INTEGER OPTIONAL,
                         -- must occur if value occurs.
                      value     [3]   IMPLICIT INTEGER OPTIONAL,
                         -- if omitted, and Type occurs, then Type is what is unsupported.
                      term      [4]   Term
                         -- if occurs, term is illegal or not supported, for attribute value,
                         -- if value occurs; otherwise, for type.
                                              },
--
attCombo  [1005]  IMPLICIT AttributeList,  -- attribute combination not supported
--
term      [1006]  IMPLICIT SEQUENCE{
                      problem    [1] IMPLICIT INTEGER{
                                      codedValue    (1),
                                      unparsable    (2),
                                      tooShort      (3),
                                      type          (4)},
                      term       [2] Term},
--
proximity  [1007] CHOICE{                         -- proximity diagnostics:
                   resultSets  [1]   IMPLICIT NULL,           -- proximity between sets not supported
                   badSet      [2]   IMPLICIT VisibleString,  -- bad result set specified
                   relation    [3]   IMPLICIT INTEGER,        -- 1 to 6 ; relation not supported
                   unit        [4]   IMPLICIT INTEGER,        -- unsupported unit code
                   distance    [5]   IMPLICIT INTEGER,        -- unsupported distance
                   attributes  [6]   AttributeList,           -- proximity not supported with specified
                                                              --      attribute combination
                   ordered     [7]   IMPLICIT NULL,           -- ordered flag not supported
                   exclusion   [8]   IMPLICIT NULL            -- exclusion flag not supported
                          },
--
scan          [1008] CHOICE{                              -- scan diagnostics
                      nonZeroStepSize    [0]  IMPLICIT NULL,      -- only zero step size supported
                      specifiedStepSize  [1]  IMPLICIT NULL},     -- Specified step size not supported
--
```

```
  sort           [1009] CHOICE{
                        sequence      [0]    IMPLICIT NULL,    -- Cannot sort according to sequence.
                        noRsName      [1]    IMPLICIT NULL,      -- no result set name supplied
|                       tooMany       [2]    IMPLICIT INTEGER,   --   too many input result sets.
|                                                                --  Maximum supplied
|                       incompatble   [3]    IMPLICIT NULL,      -- records with different formats
|                                                                -- not compatible for sorting
                        generic       [4]    IMPLICIT NULL,      -- generic sort not supported
                                                                 -- (db specific only)
                        dbSpecific    [5]    IMPLICIT NULL,      -- db specific sort not supported
                        sortElement   [6]    SortElement,
                        key           [7]    IMPLICIT INTEGER{
                                               tooMany    (1),   -- to many sort keys
                                               duplicate  (2)},  -- duplicate sort keys
                        action        [8]    IMPLICIT NULL,    -- unuspported missing data action
                        illegal       [9]    IMPLICIT INTEGER{
                                               relation (1),     -- illegal sort relation
                                               case     (2),     -- illegal case value
                                               action   (3),     -- illegal missing data action
                                               sort     (4)},    -- illegal sort
|                 inputTooLarge       [10]   IMPLICIT SEQUENCE OF VisibleString,
                                                             --   one or more of the input result sets
                                                             --   too large to sort
|                aggregateTooLarge    [11]   IMPLICIT NULL   --  aggregate result set too large
                                      },
  segmentation   [1010]  CHOICE{
                        segments      [0]    IMPLICIT NULL   -- Cannot guarantee records will fit
                                                             --  within specified segments.
                                      },
  extServices    [1011] CHOICE{
                        req           [1] IMPLICIT INTEGER{ -- bad request
                                            nameInUse    (1),   -- Package name already in use
                                            noSuchName   (2),   -- no such package, on modify/delete
                                            quota        (3),   -- quota exceeded
                                            type         (4)},  -- extended service type not supported
                        permission    [2] IMPLICIT INTEGER{ -- permission denied on ES:
                                            id           (1),   --  id not authorized
                                            modifyDelete (2)},  -- cannot modify or delete
                        immediate     [3] IMPLICIT INTEGER{ -- immediate execution:
                                            failed       (1),   -- failed
                                            service      (2),   -- not supported for this service, or
                                            parameters   (3)    -- for these parameters
                                                 }}}
  END
```

Appendix REC

Record Syntaxes

(Normative)

This standard registers the following object identifiers for record syntaxes:

Object identifiers assigned for bibliographic syntaxes, not described in ASN.1.

Unimarc	{Z39-50-recordSyntax	**1**	}
Intermarc	{Z39-50-recordSyntax	**2**	}
CCF	{Z39-50-recordSyntax	**3**	}
USmarc	{Z39-50-recordSyntax	**10**	}
UKmarc	{Z39-50-recordSyntax	**11**	}
Normarc	{Z39-50-recordSyntax	**12**	}
Librismarc	{Z39-50-recordSyntax	**13**	}
Danmarc	{Z39-50-recordSyntax	**14**	}
Finmarc	{Z39-50-recordSyntax	**15**	}
MAB	{Z39-50-recordSyntax	**16**	}
Canmarc	{Z39-50-recordSyntax	**17**	}
SBN	{Z39-50-recordSyntax	**18**	}
Picamarc	{Z39-50-recordSyntax	**19**	}
Ausmarc	{Z39-50-recordSyntax	**20**	}

Note: The following transfer syntax (defined in a different standard) may be used in conjunction with the above bibliographic definitions:

ISO2709 {iso standard 2709 transfer-syntax (1) character-encoding (1)}

When presentation context negotiation is used, the above syntaxes may be paired with the transfer syntax identified by the object identifier ISO2709 for the transfer-syntax for bibliographic records defined in ISO 2709. When presentation context negotiation is not used, the above record syntaxes are assumed to be paired with ISO2709.

Object identifiers assigned for syntaxes which are described using ASN.1.

Explain	{Z39-50-recordSyntax **100**}	(See REC.1)
SUTRS	{Z39-50-recordSyntax **101**}	(See REC.2)
OPAC	{Z39-50-recordSyntax **102**}	(See REC.3)
Summary	{Z39-50-recordSyntax **103**}	(See REC.4)
GRS-0	{Z39-50-recordSyntax **104**}	
GRS-1	{Z39-50-recordSyntax **105**}	(See REC.5)

Note: The following transfer syntax (defined in a different standard) may be used in conjunction with these definitions:

ISO8825 ::= OBJECT IDENTIFIER {joint-iso-ccitt (2) basic-encoding (1)}

When presentation context negotiation is used, these syntaxes may be paired with the transfer syntax identified by the object identifier ISO8825 for the transfer-syntax defined in ISO 8825. When presentation context negotiation is not used, the above record syntaxes are assumed to be paired with ISO8825.

REC.1 Explain Record Syntax

```
RecordSyntax-explain
{Z39-50-recordSyntax explain (100)} DEFINITIONS ::=
BEGIN
| IMPORTS AttributeList, AttributeElement, AttributeSetId, OtherInformation, DatabaseName,
| ElementSetName, IntUnit, Unit, StringOrNumeric FROM Z39-50-APDU;
Explain-Record ::= CHOICE {
    generalTargetInfo       [0]     IMPLICIT GeneralTargetInfo,
    listOfDatabases         [1]     IMPLICIT ListOfDatabases,
    targetIRparameters      [2]     IMPLICIT TargetIRparameters ,
    generalDatabaseInfo     [3]     IMPLICIT GeneralDatabaseInfo,
    dbIRparameters          [4]     IMPLICIT DatabaseIRparameters,
    dbAttributesInfo        [5]     IMPLICIT DatabaseAttributesInfo,
    attributeDetails        [6]     IMPLICIT AttributeDetails,
    recordSyntaxInfo        [7]     IMPLICIT RecordSyntaxInfo,
|   schemaInfo              [8]     IMPLICIT SchemaInfo,
    recordElementInfo       [9]     IMPLICIT RecordElementInfo,
    elementDetails          [10]    IMPLICIT ElementDetails,
    displayLanguage         [11]    IMPLICIT DisplayLanguage,
    termList                [12]    IMPLICIT TermList,
|   categoryList            [100]   IMPLICIT CategoryList}

GeneralTargetInfo ::= SEQUENCE {
    commonInfo              [0]     IMPLICIT CommonInfo OPTIONAL,
    targetname              [1]     IMPLICIT VisibleString OPTIONAL,
    nicknames               [2]     IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
    description             [3]     IMPLICIT HumanString OPTIONAL,
    usage-restrictions      [4]     IMPLICIT HumanString OPTIONAL,
    contactInfo             [5]     IMPLICIT HumanString OPTIONAL,
    paymentAddress          [6]     IMPLICIT HumanString OPTIONAL,
    recent-news             [7]     IMPLICIT HumanString OPTIONAL,
    hours                   [8]     IMPLICIT HumanString OPTIONAL,
    icon                    [9]     IMPLICIT IconObject OPTIONAL,
    otherTargetInfo                 OtherInformation}

ListOfDatabases ::= SEQUENCE {
    commonInfo              [0]     IMPLICIT CommonInfo OPTIONAL,
    dbDescriptions          [1]     IMPLICIT SEQUENCE OF DatabaseSummary,
    dbCombinations          [2]     IMPLICIT SEQUENCE OF DbCombo,
    otherListDBInfo                 OtherInformation OPTIONAL}

TargetIRparameters ::= SEQUENCE {
    commonInfo              [0]  IMPLICIT CommonInfo OPTIONAL,
    namedResultSets         [1]  IMPLICIT BOOLEAN,
    multipleDBsearch        [2]  IMPLICIT BOOLEAN,
    maxResultSets           [3]  IMPLICIT INTEGER OPTIONAL,
    maxResultSize           [4]  IMPLICIT INTEGER OPTIONAL,
    maxTerms                [5]  IMPLICIT INTEGER OPTIONAL,
```

```
    timeoutInterval          [6]   IMPLICIT INTEGER OPTIONAL,
    queryTypesSupported      [7]   IMPLICIT SEQUENCE OF INTEGER OPTIONAL,
    servicesSupported        [8]   IMPLICIT SEQUENCE OF INTEGER OPTIONAL,
    structureDescription     [9]   IMPLICIT VisibleString,
    pageDescription          [10]  IMPLICIT VisibleString,
    proximitySupport         [11]  IMPLICIT ProximitySupport,
    diagnosticsSupported     [12]  IMPLICIT SEQUENCE OF OBJECT IDENTIFIER,
    resourcesSupported       [13]  IMPLICIT SEQUENCE OF OBJECT IDENTIFIER,
    accessSupported          [14]  IMPLICIT SEQUENCE OF OBJECT IDENTIFIER,
    otherIRInfo                    OtherInformation OPTIONAL}

GeneralDatabaseInfo ::= SEQUENCE {
    commonInfo               [0]   IMPLICIT CommonInfo OPTIONAL,
    dbSummary                [1]   IMPLICIT DatabaseSummary,
    dbCombinations           [2]   IMPLICIT SEQUENCE OF DbCombo OPTIONAL,
    disclaimers              [4]   IMPLICIT HumanString OPTIONAL,
    news                     [5]   IMPLICIT HumanString OPTIONAL,
    recordCount              [6]   CHOICE {
                                      actualNumber   [0]   IMPLICIT INTEGER,
                                      approxNumber   [1]   IMPLICIT INTEGER} OPTIONAL,
    defaultOrder             [7]   IMPLICIT HumanString OPTIONAL,
    avRecordSize             [8]   IMPLICIT INTEGER OPTIONAL,
    maxRecordSize            [9]   IMPLICIT INTEGER OPTIONAL,
    hours                    [10]  IMPLICIT HumanString OPTIONAL,
    bestTime                 [11]  IMPLICIT HumanString OPTIONAL,
    lastUpdate               [12]  IMPLICIT GeneralizedTime OPTIONAL,
    updateInterval           [13]  IMPLICIT SEQUENCE {
                                      unit     [0]   IMPLICIT INTEGER{
                                                        seconds  (1),
                                                        hours    (2),
                                                        days     (3),
                                                        weeks    (4),
                                                        months   (5) },
                                      value    [1]   IMPLICIT INTEGER } OPTIONAL,
    coverageDates            [14]  IMPLICIT HumanString OPTIONAL,
    proprietary              [15]  IMPLICIT BOOLEAN,
    copyrightText            [16]  IMPLICIT HumanString OPTIONAL,
    copyrightNotice          [17]  IMPLICIT HumanString OPTIONAL,
    restrictedAccess         [18]  IMPLICIT BOOLEAN,
    accessText               [19]  IMPLICIT HumanString OPTIONAL,
    costInfo                 [20]  IMPLICIT SEQUENCE {
                                      connectCharge   [0]   IMPLICIT Charge OPTIONAL,
                                      displayCharge   [1]   IMPLICIT Charge OPTIONAL,
                                      searchCharge    [2]   IMPLICIT Charge OPTIONAL} OPTIONAL,
    producerContactInfo      [21]  IMPLICIT ContactInfo OPTIONAL,
    supplierContactInfo      [22]  IMPLICIT ContactInfo OPTIONAL,
    submisionContactInfo     [23]  IMPLICIT ContactInfo OPTIONAL,
    otherGeneralDBInfo             OtherInformation OPTIONAL}
```

```
DatabaseIRparameters ::= SEQUENCE {
   commonInfo           [0]   IMPLICIT CommonInfo OPTIONAL,
   databaseName         [1]   IMPLICIT DatabaseName,
   attributeSetIds      [2]   IMPLICIT SEQUENCE OF AttributeSetId OPTIONAL,
   schemas              [3]   IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
   recordSyntaxes       [4]   IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
   elementSetNames      [5]   IMPLICIT SEQUENCE OF SEQUENCE {
                                 name           [0]   IMPLICIT VisibleString,
                                 description    [1]   IMPLICIT HumanString OPTIONAL,
                                 contents       [2]   IMPLICIT SEQUENCE OF VisibleString
                                                          OPTIONAL} OPTIONAL,
   elementSetRelations  [6]   IMPLICIT SEQUENCE OF SEQUENCE {
                                 subSet      [0]    IMPLICIT VisibleString,
                                 superSet    [1]    IMPLICIT VisibleString} OPTIONAL,
   diagnosticSets       [7]   IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
   resourceReports      [8]   IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
   accessChallenges     [9]   IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
   structureDescription [10]  IMPLICIT VisibleString,
   pageDescription      [11]  IMPLICIT VisibleString,
   proximitySupport     [12]  IMPLICIT ProximitySupport,
   associatedDbs        [13]  IMPLICIT DbCombo,
   subDbs               [14]  IMPLICIT DbCombo,
   termLists            [15]  IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
   queryTypesSupported  [16]  IMPLICIT SEQUENCE OF INTEGER OPTIONAL,
   otherDatabaseInfo          OtherInformation OPTIONAL}


DatabaseAttributesInfo ::= SEQUENCE {
   commonInfo           [0]   IMPLICIT CommonInfo,
   databaseName         [1]   IMPLICIT DatabaseName,
   attributeSetId       [2]   IMPLICIT AttributeSetId,
   useAttributeInfo     [3]   IMPLICIT SEQUENCE OF UseAttributeInfo,
   nonUseInfo           [4]   IMPLICIT SEQUENCE OF SEQUENCE{
                                 type           [1]   IMPLICIT INTEGER,
                                 description    [2]   IMPLICIT HumanString} OPTIONAL,
   otherAttributesInfo        OtherInformation OPTIONAL}


AttributeDetails ::= SEQUENCE {
   commonInfo                [0]   IMPLICIT CommonInfo,
   databaseNames             [1]   IMPLICIT DbCombo,
   attributeSetId            [2]   IMPLICIT AttributeSetId,
   useAttributeInfo          [3]   IMPLICIT UseAttributeInfo,
   otherAttributeDetailInfo        OtherInformation OPTIONAL}
```

```
RecordSyntaxInfo ::= SEQUENCE {
   commonInfo            [0]   IMPLICIT CommonInfo OPTIONAL,
   recordSyntax          [1]   IMPLICIT OBJECT IDENTIFIER,
   nicknames             [2]   IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
   transferSyntaxes      [3]   IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
   description           [4]   IMPLICIT HumanString OPTIONAL,
   databaseNames         [5]   IMPLICIT DbCombo OPTIONAL,
   asn1Module            [6]   IMPLICIT VisibleString OPTIONAL,
   elementSetNames       [7]   IMPLICIT SEQUENCE OF ElementSetName OPTIONAL,
   otherSyntaxInfo             OtherInformation OPTIONAL}


SchemaInfo ::= SEQUENCE {
   commonInfo            [0]   IMPLICIT CommonInfo OPTIONAL,
   schema                [1]   IMPLICIT OBJECT IDENTIFIER,
   names                 [2]   IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
   description           [3]   IMPLICIT HumanString OPTIONAL,
   databaseNames         [4]   IMPLICIT DbCombo OPTIONAL,
   elements              [5]   IMPLICIT SEQUENCE OF SEQUENCE{
                                 elementname    [1] IMPLICIT GeneralString,
                                 elementTag     [2] StringOrNumeric,
                                 description    [3] IMPLICIT HumanString OPTIONAL} OPTIONAL,
   schemasImported       [6]   IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
   recordSyntaxes        [7]   IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
   otherSchemaInfo             OtherInformation OPTIONAL}


RecordElementInfo ::= SEQUENCE {
   commonInfo              [0]   IMPLICIT CommonInfo OPTIONAL,
   databaseName            [1]   IMPLICIT DbCombo,
   schema                  [2]   IMPLICIT OBJECT IDENTIFIER OPTIONAL,
   description             [3]   IMPLICIT HumanString OPTIONAL,
   elementSets             [4]   IMPLICIT SEQUENCE{
                                 name                ElementSetName,
                                 description   [1]   IMPLICIT HumanString OPTIONAL},
   attributeSets           [5]   IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
   numberElementDetails    [6]   IMPLICIT INTEGER,
                                 -- purely an implementation aide, the number of elements below.
   detailsPerElement       [7]   IMPLICIT SEQUENCE OF PerElementDetails,
   otherElementInfo              OtherInformation OPTIONAL}

ElementDetails ::= SEQUENCE {
   commonInfo                [0]   IMPLICIT CommonInfo OPTIONAL,
   databaseName              [1]   IMPLICIT DatabaseName,
   schema                    [2]   IMPLICIT OBJECT IDENTIFIER,
   details                   [3]   IMPLICIT PerElementDetails,
   otherElementDetailInfo          OtherInformation OPTIONAL}
```

```
DisplayLanguage ::= SEQUENCE{
   commonInfo            [0]   IMPLICIT CommonInfo OPTIONAL,
   databaseName          [1]   IMPLICIT DatabaseName,
   displays              [2]   IMPLICIT SEQUENCE OF SEQUENCE{
                                  compName     [0]   IMPLICIT OBJECT IDENTIFIER,
                                  description  [1]   IMPLICIT HumanString,
                                  display      [2]   IMPLICIT VisibleString},
   tables                [3]   IMPLICIT SEQUENCE OF SEQUENCE{
                                  name         [0]   IMPLICIT HumanString,
                                  table        [1]   IMPLICIT VisibleString},
   otherDisplayInfo            OtherInformation OPTIONAL}


TermList ::= SEQUENCE{
   commonInfo            [0]   IMPLICIT CommonInfo OPTIONAL,
   termListName          [1]   IMPLICIT VisibleString OPTIONAL,
   description           [2]   IMPLICIT HumanString OPTIONAL,
   databases             [3]   IMPLICIT DbCombo,
   alternativeDbs        [4]   IMPLICIT DbCombo OPTIONAL,
   attributeList         [5]   IMPLICIT AttributeList,
   alternativeLists      [6]   IMPLICIT SEQUENCE OF AttributeList OPTIONAL,
   maxStepSize           [7]   IMPLICIT INTEGER,
   collatingSequence     [8]   IMPLICIT HumanString OPTIONAL,
   increasing            [9]   IMPLICIT BOOLEAN OPTIONAL,
                                  -- No default.  If absent, target not telling.
   estNumberTerms        [10]  IMPLICIT INTEGER OPTIONAL,
   sampleTerms           [11]  IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
   otherTermInfo               OtherInformation}


CategoryList ::= SEQUENCE{
   commonInfo            [0]   IMPLICIT CommonInfo OPTIONAL,
   categories            [1]   IMPLICIT SEQUENCE OF SEQUENCE{
               category              [1]   IMPLICIT GeneralString,
               originalCategory      [2]   IMPLICIT GeneralString OPTIONAL,
               description           [3]   IMPLICIT HumanString OPTIONAL,
               asn1Module            [4]   IMPLICIT VisibleString OPTIONAL}}

CommonInfo ::= SEQUENCE {
   dateAdded                [0]   IMPLICIT GeneralizedTime OPTIONAL,
   dateChanged              [1]   IMPLICIT GeneralizedTime OPTIONAL,
   expiry                   [2]   IMPLICIT GeneralizedTime OPTIONAL,
   humanString-Format       [3]   IMPLICIT OBJECT IDENTIFIER OPTIONAL,
   humanString-Language     [4]   IMPLICIT LanguageCode}

HumanString ::= SEQUENCE OF SEQUENCE {
   language       [0]      IMPLICIT LanguageCode OPTIONAL,
   format         [1]     IMPLICIT OBJECT IDENTIFIER OPTIONAL,
   text           [2]      IMPLICIT OCTET STRING}
```

```
IconObject ::= SEQUENCE {
   bitsPerPixel           [0]    IMPLICIT INTEGER,
   horizontalPixels       [1]    IMPLICIT INTEGER,
   verticalPixels         [2]    IMPLICIT INTEGER,
   icon                   [3]    IMPLICIT OCTET STRING -- encoded by row
                                 }
LanguageCode ::= VisibleString  -- 3-character codes as defined for use with US MARC

ContactInfo ::= SEQUENCE{
   name                   [0]   IMPLICIT HumanString OPTIONAL,
   description            [1]   IMPLICIT HumanString OPTIONAL,
   address                [2]   IMPLICIT HumanString OPTIONAL,
   email                  [3]   IMPLICIT VisibleString OPTIONAL,
   phone                  [4]   IMPLICIT VisibleString OPTIONAL}

ProximitySupport ::= SEQUENCE {
   anySupport             [0]   IMPLICIT BOOLEAN,
   unitsSupported         [1]   IMPLICIT SEQUENCE OF INTEGER OPTIONAL}

UseAttributeInfo ::= SEQUENCE {
   name                   [1]   IMPLICIT VisibleString,
   number                 [2]   IMPLICIT INTEGER,
   description            [3]   IMPLICIT HumanString,
   subAttributes          [4]   IMPLICIT SEQUENCE OF INTEGER OPTIONAL,
   superAttributes        [5]   IMPLICIT SEQUENCE OF INTEGER OPTIONAL,
   nonUseAttributes       [6]   IMPLICIT SEQUENCE OF AttributeElement}

PerElementDetails ::= SEQUENCE {
   elementName            [1]   IMPLICIT VisibleString OPTIONAL,
   elementNumber          [2]   IMPLICIT INTEGER OPTIONAL,
   maxSize                [3]   IMPLICIT INTEGER OPTIONAL,
   minSize                [4]   IMPLICIT INTEGER OPTIONAL,
   avgSize                [5]   IMPLICIT INTEGER OPTIONAL,
   fixedSize              [6]   IMPLICIT INTEGER OPTIONAL,
   repeatable             [8]   IMPLICIT BOOLEAN,
   required               [9]   IMPLICIT BOOLEAN,
   searchAttributes       [10]  IMPLICIT AttributeList OPTIONAL,
   elementSets            [11]  IMPLICIT SEQUENCE OF ElementSetName OPTIONAL,
   description            [12]  IMPLICIT HumanString OPTIONAL,
   dataType               [13]  IMPLICIT PrintableString OPTIONAL,
   billingInfo            [14]  IMPLICIT HumanString OPTIONAL,
   restrictions           [15]  IMPLICIT HumanString OPTIONAL,
   alternateNames         [16]  IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
   genericNames           [17]  IMPLICIT SEQUENCE OF VisibleString OPTIONAL}
```

```
DatabaseSummary ::= SEQUENCE {
   databaseName          [0]   IMPLICIT DatabaseName,
   nicknames             [1]   IMPLICIT SEQUENCE OF DatabaseName OPTIONAL,
   description           [3]   IMPLICIT HumanString OPTIONAL,
   icon                  [4]   IMPLICIT IconObject OPTIONAL,
   user-fee              [5]   IMPLICIT BOOLEAN,
   available             [6]   IMPLICIT BOOLEAN,
   titleString           [7]   IMPLICIT HumanString OPTIONAL
                 -- short descriptive phrase describing content
                }

| Charge ::= SEQUENCE{
|    cost          [1] IMPLICIT IntUnit,
|    perWhat       [2] IMPLICIT Unit OPTIONAL,
|                          -- e.g. "seconds", "minutes"...
|    text          [3]     HumanString OPTIONAL}

DbCombo       ::=     SEQUENCE OF DatabaseName
END
```

REC.2 Simple Unstructured Text Record Syntax

The Simple Unstructured Text Record Syntax (SUTRS) is intended to be used for textual data in a Search or Present response, to allow the origin to display retrieved textual data with litle or no analysis and manipulation. It may be used as a record syntax.

| A SUTRS record is unstructured; the text of a SUTRS record might represent individual elements, but the elements are not explicitly identified by the syntax.

The convention prescribed by the SUTRS definition is to use a delimiter within the text to indicate the | end of a line of text. The prescribed line terminator is ASCII LF (X'0A'). Thus a SUTRS record consists simply of a string of textual data.

A recomended maximum line length of 72 characters is prescribed by this definition. This is not an absolute maximum, but it is recomended that targets make a best effort to limit lines to this length.

```
RecordSyntax-SUTRS
{Z39-50-recordSyntax SUTRS (101)} DEFINITIONS ::=
BEGIN
      SutrsRecord::=  GeneralString
-- Line terminator is ASCII LF (X'0A').
-- Recommended maximum line length is 72 characters.
END
```

REC.3 OPAC Record Syntax

```
RecordSyntax-opac
{Z39-50-recordSyntax opac (102)} DEFINITIONS ::=
BEGIN

OPACRecord ::= SEQUENCE {
|       bibliographicRecord     [1]     IMPLICIT EXTERNAL OPTIONAL,
        holdingsData            [2]     IMPLICIT SEQUENCE OF HoldingsRecord OPTIONAL}

HoldingsRecord ::=  CHOICE {
|       marcHoldingsRecord      [1]     IMPLICIT EXTERNAL,
        holdingsAndCirc         [2]     IMPLICIT HoldingsAndCircData}


HoldingsAndCircData ::=  SEQUENCE {
-- the following elements are required to display holdings in conformance with NISO standards.
        typeOfRecord            [1]     IMPLICIT VisibleString OPTIONAL, -- LDR 06
        encodingLevel           [2]     IMPLICIT VisibleString OPTIONAL, -- LDR 017
        format                  [3]     IMPLICIT VisibleString OPTIONAL, -- 007 00-01
        receiptAcqStatus        [4]     IMPLICIT VisibleString OPTIONAL, -- 008 06
        generalRetention        [5]     IMPLICIT VisibleString OPTIONAL, -- 008 12
        completeness            [6]     IMPLICIT VisibleString OPTIONAL, -- 008 16
        dateOfReport            [7]     IMPLICIT VisibleString OPTIONAL, -- 008 26-31
        nucCode                 [8]     IMPLICIT VisibleString OPTIONAL, -- 852 $a
        localLocation           [9]     IMPLICIT VisibleString OPTIONAL, -- 852 $b
        shelvingLocation        [10]    IMPLICIT VisibleString OPTIONAL, -- 852 $c
        callNumber              [11]    IMPLICIT VisibleString OPTIONAL, -- 852 $h and $i
        shelvingData            [12]    IMPLICIT VisibleString OPTIONAL, -- 852 $j thru $m
        copyNumber              [13]    IMPLICIT VisibleString OPTIONAL, -- 852 $t
        publicNote              [14]    IMPLICIT VisibleString OPTIONAL, -- 852 $z
        reproductionNote        [15]    IMPLICIT VisibleString OPTIONAL, -- 843
        termsUseRepro           [16]    IMPLICIT VisibleString OPTIONAL, -- 845
        enumAndChron            [17]    IMPLICIT VisibleString OPTIONAL, -- all 85x, 86x
        volumes                 [18]    IMPLICIT SEQUENCE OF Volume OPTIONAL,
                                                -- repeats for each volume held
        circulationData [19]    IMPLICIT SEQUENCE OF CircRecord OPTIONAL
                                                -- repeats for each circulating item.
                                                }
Volume ::=  SEQUENCE {
        enumeration             [1]     IMPLICIT VisibleString OPTIONAL,
        chronology              [2]     IMPLICIT VisibleString OPTIONAL,
        enumAndChron            [3]     IMPLICIT VisibleString OPTIONAL }
```

```
CircRecord ::=  SEQUENCE {
        availableNow          [1]    IMPLICIT BOOLEAN,
        availablityDate       [2]    IMPLICIT VisibleString OPTIONAL,
        availableThru         [3]    IMPLICIT VisibleString OPTIONAL,
        restrictions          [4]    IMPLICIT VisibleString OPTIONAL,
        itemId                [5]    IMPLICIT VisibleString OPTIONAL,
        renewable             [6]    IMPLICIT BOOLEAN,
        onHold                [7]    IMPLICIT BOOLEAN,
        enumAndChron          [8]    IMPLICIT VisibleString OPTIONAL,
        midspine              [9]    IMPLICIT VisibleString OPTIONAL,
        temporaryLocation     [10]   IMPLICIT VisibleString OPTIONAL}
END
```

REC.4 Summary Record Syntax

RecordSyntax-summary

```
{Z39-50-recordSyntax summary (103)} DEFINITIONS ::=
BEGIN
IMPORTS OtherInformation FROM Z39-50-APDU
BriefBib ::= SEQUENCE {
        title                 [1]    IMPLICIT VisibleString,
        author                [2]    IMPLICIT VisibleString OPTIONAL,
        callNumber            [3]    IMPLICIT VisibleString OPTIONAL,
        recordType            [4]    IMPLICIT VisibleString OPTIONAL,
        bibliographicLevel    [5]    IMPLICIT VisibleString OPTIONAL,
        format                [6]    IMPLICIT SEQUENCE OF FormatSpec OPTIONAL,
        publicationPlace      [7]    IMPLICIT VisibleString OPTIONAL,
        publicationDate       [8]    IMPLICIT VisibleString OPTIONAL,
        targetSystemKey       [9]    IMPLICIT VisibleString OPTIONAL,
        satisfyingElement     [10]   IMPLICIT VisibleString OPTIONAL,
        rank                  [11]   IMPLICIT INTEGER OPTIONAL,
        documentId            [12]   IMPLICIT VisibleString OPTIONAL,
        abstract              [13]   IMPLICIT VisibleString OPTIONAL,
        otherInfo                    OtherInformation OPTIONAL}

FormatSpec     ::=  SEQUENCE {
        type          [1]    IMPLICIT VisibleString,
        size          [2]    IMPLICIT INTEGER OPTIONAL,
        bestPosn      [3]    IMPLICIT INTEGER OPTIONAL}
END
```

REC.5 Generic Record Syntax 1

```
RecordSyntax-generic
{Z39-50-recordSyntax GRS-1 (105)} DEFINITIONS ::=
BEGIN
EXPORTS Variant;
IMPORTS IntUnit, Unit, CharacterString, StringOrNumeric FROM Z39-50-APDU;
--
GenericRecord ::= SEQUENCE OF TaggedElement

TaggedElement ::= SEQUENCE {
    tagType         [1] IMPLICIT INTEGER OPTIONAL,
                        -- if omitted, default should be supplied by tagSet-1.
    tagValue        [2] StringOrNumeric,
    tagOccurrence   [3] IMPLICIT INTEGER OPTIONAL,
                        -- occurrence within the database record, and relative to the parent.  No
                        -- default; if omitted, target not telling or it is irrelevant.
    data            [4] ElementData,
                        -- if CHOICE is 'subtree', 'elementNotThere', 'elementEmpty', or'diagnostic'
                        -- then next two (metaData and variantUsed) should not occur in this sequence.
                        -- If choice is noDataRequested then appliedVariant should not occur.
    metaData        [5]  IMPLICIT ElementMetaData OPTIONAL,

    appliedVariant  [6]  IMPLICIT Variant OPTIONAL}


ElementData ::= CHOICE{
    os              OCTET STRING,
    numeric         INTEGER,
    date            GeneralizedTime,
    ext             EXTERNAL,
    plainString     GeneralString,
    fancyString     CharacterString,
    intUnit                 [1] IMPLICIT IntUnit,
    elementNotThere         [2] IMPLICIT NULL,  -- element requested but not there
    elementEmpty            [3] IMPLICIT NULL,  -- element there, but empty
    noDataRequested         [4] IMPLICIT NULL,  -- variant request said 'no data'
    diagnostic              [5] IMPLICIT EXTERNAL,
    subtree                 [6] SEQUENCE OF TaggedElement -- Recursive, for nested tags.
                            }

ElementMetaData ::= SEQUENCE{
    seriesOrder             [1] IMPLICIT Order OPTIONAL,
    usageRight              [2] IMPLICIT Usage OPTIONAL,
    hits                    [3] IMPLICIT SEQUENCE OF HitVector OPTIONAL,
    displayName             [4]  IMPLICIT GeneralString OPTIONAL,
                                 -- name for element that origin can use for dispay
    availableVariants       [5] IMPLICIT SEQUENCE OF Variant OPTIONAL}
--
```

```
Order ::= SEQUENCE{
    ascending       [1] IMPLICIT BOOLEAN,
                            -- "true" means monotonically increasing (i.e. non-decreasing);
                            -- "false" means monotonically decreasing (i.e. non-increasing).
    order           [2] IMPLICIT INTEGER
                            --  order defined by schema
                    }


Usage ::= SEQUENCE{
                -- target telling origin to display this statement
    type            [1] IMPLICIT INTEGER OPTIONAL,
                            -- defined by schema.
    statement       [2] IMPLICIT GeneralString}


HitVector ::= SEQUENCE{
    satisfier               [1] IMPLICIT OCTET STRING OPTIONAL,
                                -- sourceword, etc.
    offsetIntoElement       [2] IMPLICIT IntUnit OPTIONAL,
    length                  [3] IMPLICIT IntUnit OPTIONAL,
    hitRank                 [4] IMPLICIT INTEGER OPTIONAL,
    targetToken             [5] IMPLICIT OCTET STRING OPTIONAL}


Variant ::= SEQUENCE{
    globalVariantSetId      [1] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
                                    -- Applies to the triples below, when variantSetId omitted. If
                                    -- globalVariantSetId omitted, default applies.
    triples                 [2]  IMPLICIT SEQUENCE OF SEQUENCE{
                                    variantSetId    [0]  IMPLICIT OBJECT IDENTIFIER OPTIONAL,
                                                            -- if omitted, globalVariantSetId (above)
                                                            -- applies,unless that too is omitted, in which
                                                            -- case, default used.
                                    class           [1]  IMPLICIT INTEGER,
                                    type            [2]  IMPLICIT INTEGER,
                                    value           [3]  CHOICE{
                                                            INTEGER,
                                                            GeneralString,
                                                            OCTET STRING,
                                                            OBJECT IDENTIFIER,
                                                            BOOLEAN,
                                                            Unit,
                                                            IntUnit,
                                                            NULL}}}
END
```

Appendix RSC

Resource Report Formats

(Normative)

This standard defines and registers the resource report format resource-1.

The object identifier for resource-1 is **{Z39-50-resourceReport 1}.**

```
ResourceReport-Format-Resource-1
{Z39-50-resourceReport resource-1 (1)} DEFINITIONS ::=
BEGIN
--
ResourceReport ::= SEQUENCE{
estimates      [1]   IMPLICIT SEQUENCE OF Estimate,
message        [2]   IMPLICIT VisibleString}
--
Estimate ::= SEQUENCE{
type               [1]   IMPLICIT EstimateType,
value              [2]   IMPLICIT INTEGER,  -- the actual estimate
currency-code      [3]   IMPLICIT INTEGER OPTIONAL}
-- code for representation of currencies defined in ISO 4217-1990.
-- Applicable only to monetary estimates.
--
EstimateType ::= INTEGER{
currentSearchRecords      (1),   -- estimated no. records in current (incomplete) result set for search
finalSearchRecords        (2),   -- estimated no. records that will be in result set if search completes
currentPresentRecords     (3),   -- estimated number of records in current (incomplete) set of
                                 -- records to be returned on Present
finalPresentRecords       (4),   -- estimated number of records that will be in the set of records
                                 -- to be returned by Present if Present completes
currentOpTimeProcessing   (5),   -- processing time (in .001 CPU seconds) used by operation so far
finalOpTimeProcessing     (6),   -- estimated total processing time (in .001 CPU seconds) that will
                                 -- be used by this operation if it completes
currentAssocTime          (7),   -- estimated processing time used by association (in .001 CPU sec.)
currentOperationCost      (8),   -- estimated cost for this operation so far
finalOperationCost        (9),   -- estimated cost for this operation if it completes
currentAssocCost          (10),  -- estimated cost for this association so far
finalOpTimeElapsed        (11),  -- estimated elapsed time for operation if it completes (in .001 sec.)
percentComplete           (12),  -- estimated percent complete
currentSearchAssCost      (13),  -- estimated search cost for association so far
currentPresentAssCost     (14),  -- estimated present cost for this association so far
currentConnectAssCost     (15),  -- estimated connect time cost for association so far
currentOtherAssCost       (16)   -- estimated other cost (not included in 13-15) for association so far
                          }
END
```

Appendix ACC

Access Control Formats

(Normative)

This standard defines and registers the access control format definitions below, and assigns the following object identifiers:

```
prompt-1      {Z39-50-accessControl  1}
des-1         {Z39-50-accessControl  2}
krb-1         {Z39-50-accessControl  3}
cop-1         {Z39-50-accessControl  4}
```

Access control formats are defined for use within the parameters securityChallenge and securityChallengeResponse of the AccessControlRequest and AccessControlResponse APDUs, and idAuthentication of the InitializeRequest APDU.

AccessControlFormat-prompt-1

```
{Z39-50-AccessControl prompt-1 (1)} DEFINITIONS ::=
BEGIN
PromptObject ::= CHOICE{
      challenge   [1] IMPLICIT PromptInfo,
      response    [2] IMPLICIT PromptInfo}
  PromptInfo ::= SEQUENCE OF SEQUENCE {
|     promptString  [1] IMPLICIT GeneralString,
      promptEnum  [2] IMPLICIT INTEGER{
|                          groupId        (0),
                           userId         (1),
                           password       (2),
                           oldPassword    (3)} OPTIONAL,
|     promptValue  [3] IMPLICIT GeneralString OPTIONAL}
END
```

AccessControlFormat-des-1

```
{Z39-50-accessControlFormat  des-1 (2)} DEFINITIONS ::=
BEGIN
      DES-RN-Object ::= CHOICE {
         challenge      [1]      IMPLICIT DRNType,
         response       [2]      IMPLICIT DRNType}
      DRNType ::= SEQUENCE{
               userId              [1]      IMPLICIT OCTET STRING OPTIONAL,
               salt                [2]      IMPLICIT OCTET STRING OPTIONAL,
               randomNumber  [3]      IMPLICIT OCTET STRING}
END
```

AccessControlFormat-krb-1

```
{Z39-50-accessControlFormat  krb-1 (3)} DEFINITIONS ::=
BEGIN
    KRBObject ::= CHOICE {
      challenge   [1]    IMPLICIT KRBRequest,
      response    [2]    IMPLICIT KRBResponse}
    KRBRequest ::= SEQUENCE{
          service           [1]   IMPLICIT VisibleString,
          instance          [2]   IMPLICIT VisibleString OPTIONAL,
          realm             [3]   IMPLICIT VisibleString OPTIONAL}
       -- target requests a ticket for the given service, instance, and realm
    KRBResponse ::= SEQUENCE{
          userid            [1]   IMPLICIT VisibleString OPTIONAL,
          ticket            [2]  IMPLICIT OCTET STRING}
       -- origin responds with a ticket for the requested service
END
```

AccessControlFormat-cop-1

```
{Z39-50-accessControlFormat  cop-1 (4)} DEFINITIONS ::=
BEGIN
   CopyrightObject ::= CHOICE{
      challenge   [1]    IMPLICIT Copyright,
      response    [2]    IMPLICIT Accepted}
Copyright  ::=    VisibleString
Accepted   ::=    BOOLEAN
-- Request contains 'copyright', the text of the copyright message to be displayed to the user.
-- Response contains 'accepted', which indicates whether the user had been shown,
-- and accepted, the terms of the copyright.  This is not intended to be legally binding,
-- but provides a good-faith attempt on the part of the target to inform the
-- user of the copyright.
END
```

Appendix EXT

Extended Services Defined by this Standard

(Normative)

This standard defines and registers the Extended Services listed below, and assigns the following object identifiers:

PersistentResultSet	{Z39-50-ExtendedServices 1}
PersistentQuery	{Z39-50-ExtendedServices 2}
PeriodicQuerySchedule	{Z39-50-ExtendedServices 3}
ItemOrder	{Z39-50-ExtendedServices 4}
DatabaseUpdate	{Z39-50-ExtendedServices 5}
ExportSpecification	{Z39-50-ExtendedServices 6}
ExportInvocation	{Z39-50-ExtendedServices 7}
UsageAccounting	{Z39-50-ExtendedServices 8}

EXT.1 provides service descriptions, and EXT.2 provides ASN.1 definitions.

EXT.1 Service Definitions

An Extended Service is carried out by an Extended Service (ES) task, which is invoked by an ES operation. The ES Service is described in 3.2.9.1. Execution of the ES Operation results in the creation of a task package, represented by a database record in the ES database. A task package contains parameters, some of which are common to all task packages regardless of package type, and others which are specific to the task type. Among the common parameters, some are supplied by the origin as parameters in the ES request, and others are supplied by the target. The specific parameters are derived from the Request-parameter-package parameter of the ES request. Table EXT-1 provides a summary of common parameters. Their descriptions are supplied in 3.2.9.1. For those parameters listed as both "origin supplied" and "target supplied", the target may override the origin supplied value.

Common Task Package Parameter	Origin Supplied	Target Supplied
userId	x	
packageName	x	
targetReference		x (opt)
retentionTime	x (opt)	x
permissionsList	x (opt)	x
creationDateTime		x
taskStatus		x
description	x (opt)	

Table EXT-1: parameters common to all Extended Service.

EXT.1.1 Persistent Result Set Extended Service

The Persistent Result Set Extended Service allows an origin to request that the target create a persistent result from a transient result set. A transient result set is one that has been created by a Search or Sort operation during the current Z-association. The Persistent Result Task has no effect on the transient result set; it remains available for use by the Z-association. The persistent result set is saved for later use, either during the current or a different Z-association. It may subsequently be deleted, by deletion of the task package.

Note: the origin may thus cause deletion of the persistent result set, by deleting the task package, if the origin user has "delete" permission for that package.

A Present (using the ResultSetName element specification), against the Persistent Result Set Parameter Package returns a Parameter Package which contains a target-supplied transient result set name, which may be used during the same Z-association, wherever a result set name may be used (i.e in a type-1 query, Present, Sort, or Delete).

The parameters of the Persistent Result Set Extended Service are those shown in table EXT-1 as well as those in table EXT-2.

Specific Task Parameter	Origin Supplied	Target Supplied
resultSetName	x	x (if appl)
replaceOrAppend	x (if appl)	
numberOfRecords		x (if appl)

Table EXT-2: Specific Parameters for Persistent Result Set

resultSetName The name of a transient result set. In the origin request package it is the name of an existing transient result set. When the origin retrieves the parameter package, it contains a result set name which the origin may then use to refer to a transient result set representing the persistent result set represented by the package. (The target supplies this parameter only when the task package is retrieved, not on an ES response.)

replaceOrAppend This parameter is valid only on an ES request whose function is "modify" (valid only when the origin user has "modify-contents" permission). If the value is "replace" the specified result set is to replace the existing persistent result set. If the value is "append" it is to be appended to the existing result set.

numberOfRecords The target supplies this parameter when the task package is retrieved (not on an ES response). It is the total number of records contained in the persistent result set.

EXT.1.2 Persistent Query Extended Service

The Persistent Query Extended Service allows an origin to request a target to save a Z39.50 Query for later reference, during the same or a subsequent Z-association.

The parameters of the Persistent Query Extended Service are those shown in table EXT-1 as well as those in table EXT-3.

Specific Task Parameter	Origin Supplied	Target Supplied
queryType	x	
query	x	
databaseNames	x (opt)	

Table EXT-3: Specific Parameters for Persistent Query

queryType The type of the query.

query In the request package, either the actual query to be saved or the name of another persistent query to be copied into this package. In the parameter package, the actual query.

databaseNames Optional list of databases.

EXT.1.3 Periodic Query Schedule Extended Service

The Periodic Query Schedule Extended Service allows an origin to request that the target establish a Periodic Query Schedule. The origin can also request that the schedule be "activated", either as part of the initial request to Create the schedule, or as part of a subsequent request to modify the schedule. The parameters of the Periodic Query Schedule Extended Service are those shown in table EXT-1 as well as those in table EXT-4.

Specific Task Parameter	Origin Supplied	Target Supplied
activeFlag	x	
querySpec	x	
databaseNames	x (opt)	
period	x	x (opt)
expiration	x (opt)	x (opt)
resultSetPackage	x (opt)	x (if appl)
replaceIndicator	x (if appl)	
alertVehicle	x (opt)	
alertDestination	x (opt)	
exportParameters	x (opt)	
lastQueryTime		x
lastResultNumber		x
numberSinceModify		x (opt)

Table EXT-4: Specific Parameters for Periodic Query Schedule

activeFlag On a Create request, if this flag set, the Periodic Query Schedule is to be activated immediately (upon receipt and validation of its parameters); otherwise the schedule is to be Created but not activated. On a Modify request (which may contain as little as just the ActiveFlag), the origin may activate or deactivate the schedule. In the parameter package, this parameter indicates whether the schedule is active.

querySpec Either (a) a query type and query, or (b) the name of a Persistent Query Package. If (a), then the databaseNames parameter is required.

databaseNames A list of databases; required if the parameters querySpec takes the form (a) above. If querySpec takes form (b): if this parameter is omitted, the databases specified in queryPackage apply; otherwise the specified databases will be overridden for this Periodic Query task.

period The time period between invocations of the query. The target may override the period specified by the origin. Period may be a number of days, a frequency (e.g. daily, business daily, weekly, monthly), or "continuous" meaning the search is to be run continuously (or at the target's discretion).

expiration An end time/date for the target to discontinue execution of this Periodic Query. The target may override the origin supplied value. If the origin supplies a value and the target does not support expiration, the target should reject the ES request.

resultSetPackage The name of a Persistent Result Set package. Optional in the request; if included, it is the name of an existing package. If omitted in the request, the target is to create a persistent result set.

replaceOrAppend Indicates whether the target is to replace the contents of the result set each time the query is invoked, or append any new results to the end of the result set. The latter allows the target to continually extend the result set by appending new records. If the value of the parameter Period is 'continuous" it is recommended that the value of this parameter be 'append'.

alertVehicle The alert delivery mechanism (FAX, X.400, Pager, Z39.50 Resource Control) to be used to automatically deliver Alerts triggered by receipt of new Periodic Query results.

alertDestination The destination address for Alerts triggered by receipt of new Periodic Query results (e.g, FAX #, X.400 address, pager #).

exportParameters A Export Parameter Package, or the actual contents of an Export Package to be used with this Periodic Query. It is included only if the origin wants newly posted results to be exported; if so new results may also be posted to ResultSetName if also specified.

lastQueryTime The last time this Periodic Query was invoked.

lastResultNumber Number of new records obtained the last time this query was invoked.

numberSinceModify Total number of records obtained via invocation of this Query since the last time this Periodic Query Package was modified.

EXT 1.4 Item Order Extended Service

The Item Order Extended Service allows an origin to submit an item order request to the target. The task package identifies the requested item, either by specifying a database record identified by a result set, or by using a variation of the Interlibrary Loan Request PDU (see ISO 10161).

The parameters of the Document Order Extended Service are those shown in table EXT-1 as well as those in table EXT-5.

Specific Task Parameter	Origin Supplied	Target Supplied
requestedItem	x (optional)	
contactInformation	x (optional)	
additionalBillingInformation	x (optional)	
statusOrErrorReport		x

Table EXT-5: Task-Specific Parameters for Document Order

requestedItem Identifies the requested item, either by a result set item, a modified ILL PDU request, or both. A result set item includes a result set name (the name of a transient result set belonging to the current Z-association) and an ordinal number of an entry within the result set. A "modified ILL PDU request" is based on the ILL Request APDU of the Interlibrary Loan protocol (ISO 10161).

ContactInformation A name, phone number, and electronic mail address of a contact-person.

AdditionalBillingInformation Includes payment method, credit card information, customer reference, and customer purchase order number.

StatusOrErrorReport Provides status or error information. This parameter is based on the StatuOrErrorReport APDU of the ILL protocol.

EXT 1.5 DatabaseUpdate Extended Service

The databaseUpdate Extended Service allows an origin to request that the target update a database: insert one or more new records, or replace or delete one or more existing records.

Note: this service definition does not address concurrency; if multiple users try to update the same record, it may be that only the first request served by the target will update the intended data, and the remaining request may update a record whose content has changed.

The parameters of the databaseUpdate Extended Service are those shown in table EXT-1 as well as those in table EXT-6.

Specific Task Parameter	Origin Supplied	Target Supplied
action	x	
records	x	x (if applicable)
fullRecordsRequested	x	
updateStatus		x

Table EXT-6: Task-Specific Parameters for Update

action Insert, Replace, or Delete.

records In the ES request: the origin supplies the records. For Insert or Replace, the origin supplies whole records. For Replace or Delete, the records must all contain identifying keys. For Delete, only the identifying key might be required by the target, however, a target might require more of the record to ensure that the correct record is deleted. For Insert, identifying keys might or might not be required by the target.

In the task package, the target will supply a response record for each of the records supplied by the origin; each response record is a surrogate diagnostic record or a retrieval record. Each retrieval record includes an updateStatus and may otherwise consist of only a record id. It may include the full record, depending on the value of the parameter fullRecordsRequested.

fullRecordsRequested The origin indicates whether it want the target to include the full records in the task package.

updateStatus One of the following:

Success	-	Update performed successfully.
Partial	-	Update failed for one or more records.
Failure	-	Target rejected execution of the task.

EXT 1.6 Export Specification Extended Service

The Export Specification Extended Service allows an origin to request that the target establish an export specification. Once established, the export specification may be subsequently invoked (repeatedly) by an Export Invocation Extended Services task; there may in fact be multiple invocations running simultaneously.

An Export Specification includes a delivery destination, delivery mechanism, and other parameters, which control the delivery of a unit of information (one or more result set records). The destination might be a printer or some other device. The delivery mechanism could include fax, X.400, FTAM, UUCP, FTP, or a target-supported print device. The parameters of the Export Specification Extended Service are those shown in table EXT-1 as well as those in table EXT-7.

Specific Task Parameter	Origin Supplied	Target Supplied
exportVehicle	x	
composition	x	
exportDestination	x	
The following parameters may apply only to specific vehicles or destinations:		
paperType-Size	x (optional)	
deliverImages	x (optional)	
orientation	x (optional)	
textJustification	x (optional)	
fontStyle	x (optional)	
fontSize	x (optional)	
fontMetric	x (optional)	
lineSpacing	x (optional)	
numberOfColumns	x (optional)	
verticalMargins	x (optional)	
horizontalMargins	x (optional)	
pageOrdering	x (optional)	
concatenatePages	x (optional)	
termHighlighting	x (optional)	
footnoteLocation	x (optional)	

Table EXT-7: Task-Specific Parameters for Export Specification

exportVehicle Fax, E-mail type (e.g, X.400) print device, FTP, FTAM, etc.

composition Record syntax, element specification, variants, etc. of the records to be Exported.

exportDestination An address or other destination instruction (e.g. phone number) corresponding to the export vehicle (e-mail address, printer address, fax number).

paperType-Size Type and size of the output paper media (when relevant); e.g, A-1, B, C, etc.

deliverImages A boolean flag indicating whether images included in the records are to be delivered or omitted.

orientation Portrait or landscape.

textJustification Left, right, both, center.

fontStyle, fontSize, and fontMetric: Font style, size, and metric to be used in output formatting, if applicable.

lineSpacing Single, double, etc.

numberOfColumns Number of text columns on a printed page.

verticalMargins Indentation inches from top and bottom.

horizontalMargins Indentation inches from left and right.

pageOrdering Forward or reverse.

concatenatePages A boolean flag indication whether to begin each document on a new page.

termHighlighting A boolean flag indicating whether to highlight terms.

footnoteLocation Desired location of footnotes: inline, end of page, end of each document, end of last document, or none.

EXT 1.7 Export Invocation Extended Service

The Export Invocation Extended Service allows an origin to invoke an export specification established by an Export Specification task as described in EXT 1.5. which may be subsequently invoked by an Export Invocation Extended Service task. The parameters of the Export Invocation Extended Service are those shown in table EXT-1 as well as those in table EXT-8.

Specific Task Parameter	Origin Supplied	Target Supplied
exportSpecification	x	
resultSetId	x	
resultSetRecords	x	
numberOfCopies	x	
overideParameters	x (optional)	
mediaQuantity	x (optional)	
paginationType		x (optional)
pagesSoFar		x (optional)
cost		

Table EXT-8: Task-Specific Parameters for Export Invocation

exportSpecification packageName of an export specification.

resultSetId Result set from which records are selected for export.

resultSetRecords Records to be exported, according to the export specification identified by the paramter exportSpecification, and qualified by the parameter overideParameters. This parameter may specify that all records in the result set are to be exported, or it may specify a set of ranges of result set records, in which case the last range may indicate that all records beginning with a specific record are to be exported.

NumberOfCopies Number of copies requested, (if applicable, for example, for printing).

overideParameters The request may include any of the parameters in table EXT-5, whose value overides the corresponding value in the export specification referenced by this request.

mediaQuantity Number of pages, message packets, etc., in the information to be exported.

paginationType Publisher/pagination category used for formatting, if applicable.

pagesSoFar Number of pages printed so far, if applicable.

cost cost to export this information.

EXT 1.8 Usage Accounting Report Extended Service

For information delivered via an external delivery system, the external system may be required to capture data about the usage of the re-distributed information. This usage data may include the number of users who received a particular document, or the price (perhaps including packet delivery charges) of documents delivered.

The Usage Accounting Report Extended Service allows an origin to send a usage accounting report to the target, containing usage accounting records identifying the specific information that was delivered (e.g, document, image), the number of end users who received it, final (retail) document retrieval and network delivery price data, and possibly, external delivery system user identification data (which could be a user account number, department number, group id, or some combination of these, to be used for chargeback billing by the external system's billing process).

The parameters of the Usage Accounting Extended Service are those shown in table EXT-1 as well as those in table EXT-9.

Specific Task Parameter	Origin Supplied	Target Supplied
Note: The following parameters form a set that may recur. Each occurrence represents a usage accounting record.		
deliveredInfoId	x	
origin SpecificInfo	x (opt)	
numberCopiesDelivered	x	
pricingCategory	x (opt)	
finalPrice	x (opt)	
networkDeliveryPrice	x (opt)	
originSpecificUserInfo	x (opt)	
deliveryDateTime	x (opt)	

Table EXT-9: Task-Specific Parameters for Usage Accounting

deliveredInfoId An identifier of the record or document that was delivered to the end-user.

originSpecificInfoId Optional data used by the Origin to uniquely identify the information delivery operation being confirmed. Its format is determined by the specific audit/logging needs of the origin. This parameter is not meaningful unless numberCopiesDelivered is 1.

numberOfCopiesDelivered Number of copies of the record that were delivered by the origin.

pricingCategory Allows the origin to distinguish among different categories of pricing. Values are:

- 'N' - finalPrice specifies normal pricing;
- 'I' - the delivery of this information was used for internal purposes (integration testing, customer services, training, etc.);
- 'Pxx'- information was delivered based on a special promotional offer. 'xx' is two decimal digits representing the percentage of discount provided (relative to the base/wholesale price).

finalPrice The final price (or retail price, in the case of wholesale delivery to external system agents) charged to the end user for access to and retrieval of the information. A currency code is included.

networkDeliveryPrice A network message-handling (e.g, packet data delivery) fee determined by an external delivery agent or system. This is is addition to the finalPrice (when present). A currency code included.

originSpecificUserInfo User identification data, which may serve as audit trail data for the origin, and which would generally be used for periodic reconciliation of usage records between the origin (as provider) and the delivery agent. However, it could be used to provide a chargeback billing service for agents willing to pay for this service. This parameter is meaningful only if numberCopiesDelivered is 1.

deliveryTimeDate Time and date that the information was delivered to its final destination. It would be used in combination with originSpecificUserInfo to provide audit trail details for the external system agent. This parameter is meaningful only if numberCopiesDelivered is 1.

EXT.2 ASN.1 Definitions of Extended Services Parameter Package

```
ESCommon DEFINITIONS ::=
BEGIN
EXPORTS CommonParameters, CommonTaskPackageStructure
IMPORTS OriginSuppliedCommonParameters FROM Z39-50-APDU;
--
CommonParameters ::= SEQUENCE{
|       packageName         [0] IMPLICIT VisibleString OPTIONAL,
        originSupplied      [1] IMPLICIT OriginSuppliedCommonParameters,
        targetSupplied      [2] IMPLICIT SEQUENCE{
                            targetReference     [1] IMPLICIT OCTET STRING OPTIONAL,
|                           creationDateTime    [2] IMPLICIT GeneralizedTime,
                            taskStatus          [3] IMPLICIT INTEGER{
                                                        pending     (0),
                                                        active      (1),
                                                        complete    (2),
                                                        aborted     (3)}}}

CommonTaskPackageStructure ::= CHOICE{
    requestPackage      [1] IMPLICIT SEQUENCE{
                            toKeep          [1] OriginPartToKeep,
                            notToKeep       [2] OriginPartNotToKeep},
    taskPackage         [2] IMPLICIT SEQUENCE{
                            commonPart      [1] IMPLICIT CommonParameters,
                            specificPart    [2] IMPLICIT SEQUENCE{
                                                originPart      [1] OriginPartToKeep,
                                                targetPart      [2] TargetPart}}}
-- Each task package definition is defined as this type, and each individually defines OriginPartToKeep,
-- OriginPartNotToKeep, and Target Part.
END
```

ESFormat-PersistentResultSet

```
{Z39-50-ExtendedService PersistentResultSet (1)} DEFINITIONS ::=
BEGIN
IMPORTS CommonParameters FROM ESCommon, OriginSuppliedCommonParameters,
CommonTaskPackageStructure FROM Z39-50-APDU;

PersistentResultSet ::= CommonTaskPackageStructure

OriginPartToKeep ::= NULL
        -- for persistent result, nothing supplied by origin stays in package.

OriginPartNotToKeep ::= SEQUENCE{
        resultSetName        [1]     IMPLICIT VisibleString, -- transient result set supplied on request
        replaceOrAppend      [2]     IMPLICIT INTEGER{
                                             replace         (1),
                                             append          (2)} OPTIONAL
                                     -- only if function is "modify"
                                     }

TargetPart ::= SEQUENCE{
        resultSetName        [1] IMPLICIT VisibleString OPTIONAL,
                                     -- name of persistent result set. Meaningful only when package
                                     -- is presented (i.e. not on ES response)
        numberOfRecords      [2] IMPLICIT INTEGER OPTIONAL}
END
```

ESFormat-PersistentQuery

```
{Z39-50-ExtendedService PersistentQuery (2)} DEFINITIONS ::=
BEGIN
IMPORTS CommonParameters FROM ESCommon, Query, OriginSuppliedCommonParameters,
CommonTaskPackageStructure FROM Z39-50-APDU;

--
PersistentQuery ::= CommonTaskPackageStructure
--

OriginPartToKeep ::= SEQUENCE{
        query                [1] Query,
        dbNames              [2] IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
        additionalSearchInfo [3] OtherInformation OPTIONAL}

OriginPartNotToKeep  ::= NULL
TargetPart           ::= NULL
END
```

ESFormat-PeriodicQuerySchedule

```
{Z39-50-ExtendedService PeriodicQuerySchedule (3)} DEFINITIONS ::=
BEGIN
IMPORTS CommonParameters FROM ESCommon, Query, AlertVehicle, AlertDestination, Export,
CommonTaskPackageStructure FROM Z39-50-APDU;

PeriodicQuerySchedule ::= CommonTaskPackageStructure

OriginPartToKeep ::=SEQUENCE{
        activeFlag              [1] IMPLICIT BOOLEAN,
|       querySpec               [2] CHOICE{
|                                       actualQuery     [1]     Query,
|                                       packageName     [2]     IMPLICIT VisibleString},
        dbNames                 [3] IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
        period                  [4] CHOICE{
                                        numberDays      [1]     IMPLICIT INTEGER,
                                        frequency       [2]     IMPLICIT INTEGER{
                                                                 daily          (1),
                                                                 businessDaily  (2),
                                                                 weekly         (3),
                                                                 monthly        (4),
                                                                 continuous     (5),
                                                                 other          (6)}},
        expiration              [5] IMPLICIT GeneralizedTime OPTIONAL,
|       replaceOrAppend         [6] IMPLICIT INTEGER{
|                                       replace         (1),
|                                       append          (2),
|                                       createNew       (3)     -- only if origin and target have agreement
|                                                               -- about naming convention for the
|                                                               -- resulting package
|                                                               },
        alertVehicle            [7] AlertVehicle OPTIONAL,
        alertDestination        [8] AlertDestination OPTIONAL,
        exportParameters        [9] CHOICE{
                                        packageName     [1]     IMPLICIT VisibleString,
                                        exportPackage   [2]     Export} OPTIONAL}
| -- Following two definitions are temporary:
|       AlertVehicle            :=  GeneralString
|       AlertDestination        :=  GeneralString

| OriginPartNotToKeep ::= CHOICE{
|       nothing                 [1] NULL,
|       resultSetPackage        [2] IMPLICIT VisibleString}
TargetPart ::= SEQUENCE{
        lastQueryTime           [1]     IMPLICIT GeneralizedTime,
        lastResultNumber        [2]     IMPLICIT INTEGER,
        numberSinceModify       [3]     IMPLICIT INTEGER OPTIONAL,
|       resultSetPackage        [4]     IMPLICIT VisibleString}
END
```

ESFormat-ItemOrder

```
{Z39-50-ExtendedService ItemOrder (4} DEFINITIONS ::=
BEGIN
IMPORTS CommonParameters FROM ESCommon, CommonTaskPackageStructure FROM Z39-50-
APDU;
--
ItemOrder ::= CommonTaskPackageStructure

OriginPartToKeep ::= SEQUENCE{
    itemRequestByIllPdu    [1] IMPLICIT ModifiedILLRequestPDU OPTIONAL,
                                  -- must occur if itemRequestByResultSetItem below is not included.
    contact                [2] IMPLICIT SEQUENCE{
                               name    [1]   IMPLICIT VisibleString OPTIONAL,
                               phone   [2]   IMPLICIT VisibleString OPTIONAL,
                               email   [3]   IMPLICIT VisibleString OPTIONAL} OPTIONAL,
    addlBilling            [3] IMPLICIT SEQUENCE{
                               paymentMethod          [1]   CHOICE{
                                      billInvoice                    [0] IMPLICIT NULL,
                                      prepay                         [1] IMPLICIT NULL,
                                      depositAccount                 [2] IMPLICIT NULL,
                                      creditCard                     [3] IMPLICIT CreditCardInfo,
                                      cardInfoPreviouslySupplied     [4] IMPLICIT NULL,
                                      privateKnown                   [5] IMPLICIT NULL,
                                      privateNotKnown                [6] IMPLICIT EXTERNAL},
                               customerReference      [2] IMPLICIT VisibleString OPTIONAL,
                               customerPONumber       [3] IMPLICIT VisibleString OPTIONAL}
                                                                 OPTIONAL}
CreditCardInfo   ::= SEQUENCE{
        type          [1] IMPLICIT VisibleString,
        name          [2] IMPLICIT VisibleString,
        expDate       [3] IMPLICIT VisibleString,
        number        [4] IMPLICIT VisibleString}

OriginPartNotToKeep ::= CHOICE{
        nothingToKeep                      [1] IMPLICIT NULL,
        itemRequestByResultSetItem         [2] IMPLICIT SEQUENCE{
                               -- must occur if ItemRequest above not supplied.
                                                resultSetId       [1]   IMPLICIT VisibleString,
                                                item              [2]   IMPLICIT INTEGER}}

TargetPart ::= StatusOrErrorReport

-- Begin auxiliary definitions
ModifiedILLRequestPDU ::= SEQUENCE{
    originRequestNumber   [1]      IMPLICIT GeneralString OPTIONAL,
    serviceDateTime       [2]      IMPLICIT GeneralizedTime OPTIONAL,
    requesterId           [3]      IMPLICIT PersonOrInstitution OPTIONAL,
    responderId           [4]      IMPLICIT PersonOrInstitution OPTIONAL,
    deliveryAddress       [6]      IMPLICIT DeliveryAddress OPTIONAL,
```

```
-- modifiedILLRequestPDU (continued)
    deliveryService                 CHOICE{
                                          physical     [7]     IMPLICIT GeneralString,
                                          electronic   [50]    IMPLICIT SEQUENCE OF
                                                               ElectronicDeliveryService} OPTIONAL,
    billingAddress           [8]     IMPLICIT DeliveryAddress OPTIONAL,
    levelAndExpiryInfo       [12]    IMPLICIT SearchType OPTIONAL,
    supplyMedium             [13]    IMPLICIT SEQUENCE OF SEQUENCE{
                                              -- a list, in order of preference
                                          type                 [0] MediumType,
                                          characteristics      [1] IMPLICIT GeneralString OPTIONAL},
  placeOnHold                [14]    IMPLICIT BOOLEAN OPTIONAL, -- if omitted, responder-choice
  customer                   [15]    IMPLICIT SEQUENCE{
                                              name         [0] GeneralString OPTIONAL,
                                              status       [1] GeneralString OPTIONAL,
                                              id           [2] GeneralString OPTIONAL} OPTIONAL,
  itemId                     [16]    IMPLICIT ItemId OPTIONAL,
  supplement                         CHOICE{
                                         ext       [17]    IMPLICIT EXTERNAL,
                                         local             SupplementalItemDescription} OPTIONAL,
  cost                       [18]    IMPLICIT CostInfo OPTIONAL,
  copyrightCompliance        [19]    IMPLICIT GeneralString OPTIONAL,
  retryFlag                  [21]    IMPLICIT BOOLEAN,
  requesterNote              [46]    IMPLICIT GeneralString OPTIONAL}

PersonOrInstitution ::= SEQUENCE{
    symbol      [0] CHOICE{
                        personSymbol            [0] GeneralString,
                        institutionSymbol       [1] GeneralString} OPTIONAL,
    name        [1] CHOICE{
                        personName              [0] GeneralString,
                        institutionName         [1] GeneralString} OPTIONAL}

DeliveryAddress ::= SEQUENCE{
    postalAddress            [0] IMPLICIT PostalAddress OPTIONAL,
    electronicAddress        [1] IMPLICIT SystemAddress OPTIONAL}

PostalAddress ::= SEQUENCE{
    name                     [0] IMPLICIT PersonOrInstitution OPTIONAL,
    extendedAddress          [1] IMPLICIT GeneralString OPTIONAL,
    streetAndNumber          [2] IMPLICIT GeneralString OPTIONAL,
    postOfficeBox            [3] IMPLICIT GeneralString OPTIONAL,
    city                     [4] IMPLICIT GeneralString OPTIONAL,
    region                   [5] IMPLICIT GeneralString OPTIONAL,
    country                  [6] IMPLICIT GeneralString OPTIONAL,
    postalCode               [7] IMPLICIT GeneralString OPTIONAL,
    attention                [8] IMPLICIT GeneralString OPTIONAL}
```

```
SystemAddress ::= SEQUENCE{
        telecomServiceId            [0] IMPLICIT GeneralString OPTIONAL,
        telecomServiceAddress       [1] IMPLICIT GeneralString OPTIONAL}

ElectronicDeliveryService ::= SEQUENCE{
        -- first four parameters intended for use in an automated mode.
    mode                    [0]  IMPLICIT OBJECT IDENTIFIER OPTIONAL,
                                  --   identifies type of electronic delivery serivce
    modeParameters          [1] IMPLICIT OCTET STRING OPTIONAL, -- defined by "mode"
    documentTypeId          [2] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
    docTypeParameters       [3] IMPLICIT OCTET STRING OPTIONAL, -- Defined by doc type id
    description             [4] IMPLICIT GeneralString OPTIONAL,
                                  -- Description of service and/or document type, to be used when there
                                  -- is no oid. May be present instead of, or in addition to, above four
                                  --parameters.
    details                 [5] CHOICE{
                                deliveryAddress      [0] IMPLICIT SystemAddress,
                                deliveryId           [1] IMPLICIT PersonOrInstitution},
    nameOrCode              [6] IMPLICIT GeneralString OPTIONAL,
                                  -- Human-readable identifier or correlation info for item as shipped
    deliveryTime            [7] IMPLICIT GeneralizedTime OPTIONAL
                                  -- Preferred delivery time
                                                    }

SearchType ::= SEQUENCE{
    levelOfService  [0] GeneralString OPTIONAL, -- max 1 character
    needBeforeDate      [1] IMPLICIT GeneralizedTime OPTIONAL,
    expiryFlag          [2] IMPLICIT INTEGER{
                            needBeforeDate    (1),
                              -- If this value selected, then needBeforeDate above specifies expiry date.
                            otherDate         (2),
                            noExpiry          (3)},
    expiryDate          [3] IMPLICIT GeneralizedDate OPTIONAL
                        -- alternative expiry date, can be used only when expiry-flag is set to "Other-Date"
                            }

MediumType ::= SEQUENCE OF INTEGER{
    printed             (1),
    photocopy           (2),
    microform           (3),
    filmOrVideo         (4),
    audioRecording      (5),
    machineReadable     (6),
    other               (7)}
```

```
ItemId ::= SEQUENCE{
    itemType                [0]     IMPLICIT INTEGER{
                                        monograph (1),
                                        serial    (2),
                                        other     (3)} OPTIONAL,
    heldMediumType          [1]     IMPLICIT MediumType OPTIONAL,
    callNumber              [2]     IMPLICIT GeneralString OPTIONAL,
    author                  [3]     IMPLICIT GeneralString OPTIONAL,
    title                   [4]     IMPLICIT GeneralString OPTIONAL,
    subTitle                [5]     IMPLICIT GeneralString OPTIONAL,
    sponsoringBody          [6]     IMPLICIT GeneralString OPTIONAL,
    placeOfPublication      [7]     IMPLICIT GeneralString OPTIONAL,
    publisher               [8]     IMPLICIT GeneralString OPTIONAL,
    seriesTitleNumber       [9]     IMPLICIT GeneralString OPTIONAL,
    volumeIssue             [10]    IMPLICIT GeneralString OPTIONAL,
    edition                 [11]    IMPLICIT GeneralString OPTIONAL,
    publicationDate         [12]    IMPLICIT GeneralString OPTIONAL,
    publDateOfComponent     [13]    IMPLICIT GeneralString OPTIONAL,
    authorOfArticle         [14]    IMPLICIT GeneralString OPTIONAL,
    titleOfArticle          [15]    IMPLICIT GeneralString OPTIONAL,
    pagination              [16]    IMPLICIT GeneralString OPTIONAL,
    natBiblioNo             [17]    IMPLICIT EXTERNAL OPTIONAL,
    iSBN                    [18]    IMPLICIT GeneralString OPTIONAL, -- ISO 2108
    iSSN                    [19]    IMPLICIT GeneralString OPTIONAL, -- ISO 3297
    systemNo                [20]    IMPLICIT EXTERNAL OPTIONAL,
    additionalNoLetters     [21]    IMPLICIT GeneralString OPTIONAL,
    verificationRefSrc      [22]    IMPLICIT GeneralString OPTIONAL}

CostInfo  ::= SEQUENCE{
    accountNumber           [0] IMPLICIT GeneralString OPTIONAL,
    maximumCost             [1] IMPLICIT SEQUENCE{
                                    currencyCode    [0] IMPLICIT GeneralString OPTIONAL,
                                        -- values defined in ISO 4217-1990
                                    amount          [1] IMPLICIT GeneralString} OPTIONAL,
    reciprocalAgreement     [2] IMPLICIT BOOLEAN,
    willPayFee              [3] IMPLICIT BOOLEAN,
    paymentProvided         [4] IMPLICIT BOOLEAN}

SupplementalItemDescription ::= SEQUENCE{
    supplierDocumentID          [1] IMPLICIT GeneralString OPTIONAL,
    supplierDocumentIDsource    [2] IMPLICIT GeneralString OPTIONAL,
    publisherDocumentID         [3] IMPLICIT GeneralString OPTIONAL,
        -- next four parameters specific to type of item, e.g. "patent"
    number                      [4] IMPLICIT GeneralString OPTIONAL,  -- e g. patent number
    issuingCountry              [5] IMPLICIT GeneralString OPTIONAL,  -- e.g. country that issued
                                                                      -- patent
    type                        [6] IMPLICIT GeneralString OPTIONAL,  -- e.g. patent type
    reportNumber                [7] IMPLICIT GeneralString OPTIONAL}  -- e.g. number assigned
                                                                      -- by  patent office
```

```
StatusOrErrorReport ::= SEQUENCE{
    originRequestNumber      [1]     IMPLICIT GeneralString,
    serviceDateTime          [2]     IMPLICIT GeneralizedTime,
    requesterId              [3]     IMPLICIT PersonOrInstitution OPTIONAL,
    responderId              [4]     IMPLICIT PersonOrInstitution OPTIONAL,
    statusReport             [44]    IMPLICIT StatusReport OPTIONAL,
    errorReport              [45]    IMPLICIT ErrorReport OPTIONAL,
    note                     [46]    IMPLICIT GeneralString OPTIONAL}

ErrorReport ::= SEQUENCE{
    correlationInfo        [0] IMPLICIT GeneralString,
    reportSource           [1] IMPLICIT INTEGER{
                               user        (1),
                               provider    (2)},
    userErrorReport        [2] CHOICE{
                               securityProblem    [2]  IMPLICIT GeneralString,
                               unableToPerform    [3]  IMPLICIT INTEGER{
                                                          notAvailable          (1),
                                                          resourceLimitation    (2),
                                                          other                 (3)}} OPTIONAL,
                            -- occurs iff report-source is "user"
    providerErrorReport    [3] IMPLICIT GeneralString OPTIONAL
                               -- occurs iff report-source is "provider"
                           }

StatusReport ::= SEQUENCE{
    dateRequested      [0]     IMPLICIT GeneralizedDate OPTIONAL,
    author             [1]     IMPLICIT GeneralString OPTIONAL,
    title              [2]     IMPLICIT GeneralString OPTIONAL,
    authorOfArticle    [3]     IMPLICIT GeneralString OPTIONAL,
    titleOfArticle     [4]     IMPLICIT GeneralString OPTIONAL,
    currentState       [99]    IMPLICIT INTEGER{
                                   notSupplied          (1),
                                   inProcess            (3),
                                   forward              (4),
                                   cancelled            (7),
                                   shipped              (8),
                                   received             (9),
                                   notReceived          (100),
                                   dDSloanQueue         (102),
                                   dDSforwarded         (103),
                                   unfilledCopyright    (104),
                                   filledCopyright      (105)}}
END
```

ESFormat-Update

```
{Z39-50-ExtendedService Update (5)} DEFINITIONS ::=
BEGIN
IMPORTS CommonParameters, OriginSuppliedCommonParameters FROM ESCommon,
CommonTaskPackageStructure, DatabaseName, Records FROM Z39-50-APDU;

Update ::= CommonTaskPackageStructure

OriginPartToKeep ::= SEQUENCE{
    action                      [1] IMPLICIT INTEGER{
                                        insert      (1),
                                        replace     (2),
                                        delete      (3)},
    fullRecordsRequested        [2] IMPLICIT BOOLEAN}

OriginPartNotToKeep ::= Records -- imported from PDUs

TargetPart ::= SEQUENCE{
    updateStatus        [1] IMPLICIT INTEGER{
                                success     (1),
                                partial     (2),
                                failure     (3)},
    returnedRecords     [2] Records OPTIONAL
                            -- omitted if status is failure
                }
END
```

Appendix USR

User Information Formats

(Normative)

UserInformation formats are defined for the parameters userInformationField in the Init and InitResponse APDUs, additionalSearchInfo in the Search and SearchResponse APDUs, and otherInfo in all APDUs.

This standard defines and registers the userInformation format SearchResult-1, defined for use within a SearchResponse APDU, and SearchRequest-1, defined for use within a Search APDU. The following object identifiers are assigned:

```
SearchResult-1          {Z39-50-UserInfoFormat 1}
SearchRequest-1         {Z39-50-UserInfoFormat 2}
```

UserInfoFormat-searchResult-1

```
{Z39-50-UserInfoFormat searchResult-1 (1)} DEFINITIONS ::=
BEGIN
IMPORTS DatabaseName, ResultSetId, Term, Query FROM Z39-50-APDU;
SearchInfoReport ::= SEQUENCE OF SEQUENCE{
    subqueryId               [1]  IMPLICIT INTEGER OPTIONAL,
                                      -- Shorthand identifier of subquery.
    subqueryExpression       [2]  QueryExpression OPTIONAL,
                                      -- Client's subquery expression.
                                      -- One of these may be whole query.
    subqueryInterpretation   [3]  QueryExpression OPTIONAL,
                                      -- How target interpreted subquery.
    subqueryWeight           [4]  IMPLICIT INTEGER OPTIONAL,
                                      -- Relative weight of this subquery.
    subqueryComment          [5]  IMPLICIT SEQUENCE OF INTEGER {
                                      stopword      (1),
                                      tryPhonetic   (2),
                                      tryStemming   (3),
                                      weakenedProx  (4)} OPTIONAL,
    resultsByDB              [6]  IMPLICIT ResultsByDB OPTIONAL}

ResultsByDB ::= SEQUENCE OF SEQUENCE{
    databaseName         [1] DatabaseName OPTIONAL,
                                  -- If omitted, applies across all databases.
    count                [2] IMPLICIT INTEGER OPTIONAL,
                                  -- Number of database records which matched.
    targetAssignedName        ResultSetId OPTIONAL
                                  -- Target assigned name by which subQuery is available.
                              }
```

```
QueryExpression ::=  CHOICE {
                    term        [1] Term,
                    query       [2] Query,
                    text        [3] IA5String }
END
```

UserInfoFormat-searchRequest-1

```
{Z39-50-UserInfoFormat searchRequest-1 (2)} DEFINITIONS ::=
BEGIN
IMPORTS AttributeElement, AttributeList FROM Z39-50-APDU;

AdditionalSearchInfoRequest ::= CHOICE{
    exact           [1] IMPLICIT NULL,              -- Do it exactly like I said.
    fuzzy           [2] IMPLICIT NULL,              -- Change it as you see fit.
    almostExact     [3] IMPLICIT ByAttributes,      -- Do it like I said, except for these attributes.
    almostFuzzy     [4] IMPLICIT ByAttributes       -- Change as you see fit, except for these
                                                    -- attributes.
                                            }
--
-- If 'almostExact' or 'almostFuzzy' is chosen, the origin includes a list of attributes, in ByAttributes
-- structure below, for which it says explicitly what, if anything, the target may change them to.
--
    ByAttributes ::= SEQUENCE OF SEQUENCE {
        attribute           [1] IMPLICIT AttributeElement,
        mayReplaceWith      CHOICE{
                    mayNotReplace       [1] IMPLICIT NULL,
                                            -- That is, treat the attribute as exact.  In the case of
                                            -- 'almostExact', this choice is not valid, as it is
                                            -- redundant.
                    anything            [2] IMPLICIT NULL,
                                            -- That is, the server has freedom to change the attribute
                                            -- as it sees fit. In the case of almostFuzzy this choice
                                            -- is not valid, as it is redundant.
                    list                    AttributeList
                                            -- May replace with one from the list.
                        }
```

Appendix ESP

Element Specification Formats

(Normative)

This Standard defines and registers the element specification format eSpec-1.

The object identifier for eSpec-1 is {Z39-50-ElementSpec 1}

```
ElementSpecificationFormat-eSpec-1
{Z39-50-ElementSpec eSpec-1 (1)} DEFINITIONS ::=
BEGIN
IMPORTS Variant FROM RecordSyntax-generic, StringOrNumeric FROM Z39-50-APDU;
--
ES-1 ::= SEQUENCE{
    elementSetNames         [1] IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
                                -- Origin may supply one or more element set names, each
                                -- specifying a set of desired elements. Target to return union.
    defaultVariantSetId     [2] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
    defaultVariantRequest   [3] IMPLICIT Variant OPTIONAL,
                                -- If defaultVariantSetId is supplied, it applies whenever
                                -- variantRequest does not include variantSetId. If
                                -- DefaultVariantRequest is supplied, then EVERY element is
                                -- subject to a variant, either the the variantRequest supplied with
                                -- the element, or if it is not supplied, the default.
    elements                [4] IMPLICIT SEQUENCE OF ElementRequest OPTIONAL}
--
--
ElementRequest::= SEQUENCE{
    deliveryElement       [1] CHOICE{
                            esn         [1] IMPLICIT VisibleString,
                            elements    [2] IMPLICIT SEQUENCE OF TagPath
                                    -- each in this sequence is a schema element. All the
                                    -- elements are combined into a single "deliveryElement",
                                    -- and variant request and deliveryElementTag are applied.
                                },
    variantRequest        [2] IMPLICIT Variant OPTIONAL,
    deliveryElementTag    [3] IMPLICIT INTEGER OPTIONAL
                            -- Target to deliver using this tag, if supplied.
}
```

```
TagPath ::= SEQUENCE OF CHOICE{
    -- tagPath for a single schema element; may be one of several comprising a "deliveryElement".
    tag             [1] IMPLICIT SEQUENCE{
                            tagType             [1] IMPLICIT INTEGER OPTIONAL,
                                                        -- if omitted, schema tagType applies.
                            tagValue            [2] StringOrNumeric,
                        tagOccurrence           [3] Occurrences OPTIONAL
                                                    -- default is "first occurrence"
                                            },
|   wildThing   [2] Occurrences,
|                   -- Get Nth "thing" at this level, regardless of tag, for each N specified by "Occurrences"
|                   -- (which may be 'all' meaning match every element at this level). E.g. if "Occurrences"
|                   -- is 3, get third element regardless of its tag or the tag of the first two elements.
    wildPath    [3] IMPLICIT NULL
                    -- Match any tag at this level or below that is on a path for which next tag in this
                    -- TagPath sequence occurs. WildPath may not be last member of the TagPath sequence.
                                                }
--

Occurrences ::= CHOICE{
|   all     [1] IMPLICIT NULL,
|   last    [2] IMPLICIT NULL,
    values  [3] IMPLICIT SEQUENCE{
                    start   [1] IMPLICIT INTEGER,
                                -- if 'start' alone included, then single occurrence.
                    count   [2] IMPLICIT INTEGER OPTIONAL}
END
```

Appendix VAR

Variant Sets

(Normative)

This standard defines and registers the variant set variant-1.

The object identifier for variant-1 is {Z39-50-variantSet 1}

This definition describes the classes, types, and values, for the variant set Variant-1, that may occur in a variant specification.

A variant specification is a sequence of triples; each triple is a variant specifier (as referenced by the identifier variantSpecifier in GRS-1 and ES-1). The first component of the triple is a "Class" (integer), the second is a "Type" (integer) defined within that class, and the third is a "Value" defined for that type (its datatype depends on the type).

The following classes, types, and values are defined for Variant-1:

Class	Type	Value(s)
1	1 = variant-id	OCTET STRING. Target may supply a (transient) id for a variant (when that variant is included in a variant list) and the origin may use that id (within the same Z-association) to request the element according to that variant. Origin may include a variant-id in conjunction with other classes, such as "auxiliary formatting".
2 = format		
	1 = mime-type	e.g. text, multipart, message
	2 = mime-subtype	e.g. plain, richtext, postscript, oda, pdf, zip, jpeg, gif, tiff, mpeg
	3 = non-mime	e.g. sgml

Note: If the desired format is available as a mime content type, it should be represented by mime-type and mime-subtype (two consecutive triples); if not, it should be represented as non-mime. A variant spec may have more than one triple of type 'format'; as noted, it might require two triples to represent a mime type and subtype; also it might be necessary to represent two formats if both composition and encoding are to be represented.

Class	Type	Value(s)
3 = auxiliary formatting		
	1 = auxiliary composition specification	e.g. DTD
	2 = language	GeneralString (from USMARC code list for languages)
	3 = character set	INTEGER: registration number from ISO International Register of Character Sets
	4 = version	target defined
	5 = line length	INTEGER
	6 = lines per page	INTEGER
	7 = dots per inch	INTEGER

4 = Piece

1 = typeOfFragment	INTEGER 1 = section If value is 1, follow with "what section" (type 2 component). 2 = relative fragment if value is 2, when used by origin, this means relative to target token (type 7); when used by target it means relative to the part in previous segment or response. Follow with "what relative fragment" (type 3 component). 3 = piece if value is 3, follow with consecutive components of type 4, 5, and 6.
2 = what section	INTEGER 1 = start 2 = middle 3 = end 4 = end for now 5 = whole
3 = what relative fragment	INTEGER 1 = next 2 = previous 3 = current
4 = how much	IntUnit
5 = start	INTEGER (same unit as "how much")
6 = step	INTEGER
7 = targetToken	OCTET STRING. May be used by target in a Present response to supply a token. May be used by origin in a subsequent request, to supply same token, to refer to this fragment.

5 = Supplemental variant Information

1 = cost	IntUnit or NULL
2 = size	IntUnit or NULL
4 = integrity	INTEGER or NULL
5 = separability	INTEGER or NULL Note: NULL values above apply when used in a variantRequest
3 = message	GeneralString

6 = Return metadata

1 = NoData	NULL
2 = variant list	NULL
3 = hits	NULL

7 Unit Information

1 = Units	Unit

Appendix SCH

Schema and TagSet Definitions

(Normative)

SCH.1 Schema Definitions

This standard registers the following object identifiers for Schemas:

WAIS	{Z39-50-schema 1}
GILS	{Z39-50-schema 2}

SCH.2 TagSet Definitions

This standard defines and registers the tag set definitions tagSet-1 and tagSet-2. The object identifier for these definitions are:

tagSet-1	{Z39-50-tagSet 1}
tagSet-2	{Z39-50-tagSet 2}

SCH.2.1 Definition of tagSet-1

Element	tag	recommended ASN.1 datatype
default schema	1	OBJECT IDENTIFIER
elementsOrdered	2	BOOLEAN
elementOrdering	3	GeneralString
defaultTagType	4	INTEGER
defaultVariantSetId	5	OBJECT IDENTIFIER
defaultVariantSpec	6	VariantSpec
processingInstructions	7	GeneralString
recordUsageRight	8	INTEGER
recordUsageStatement	9	GeneralString
rank	10	INTEGER
userMessage	11	GeneralString
url	12	GeneralString
record	13	structured
local control number	14	GeneralString
creation date	15	GeneralizedTime

dateOfLastModification	16	GeneralizedTime
dateOfLastReview	17	GeneralizedTime
score	18	INTEGER
wellKnown	19	GeneralString
bodyOfDisplay	20	GeneralString
updateStatus	22	INTEGER

defaultSchema Identifies the Schema, for cases where the origin does not specify a Schema in the request, or where the target uses a schema different than that requested by the origin.

elementsOrdered If 'true', then sibling elements (i.e. with the same immediate superior) are presented as follows: tagTypes are ascending, tag values for a given type are ascending, and all elements with string tags follow those with integer tags (and are not necessarily ordered).

elementOrdering How sibling elements with the same tagType are ordered:

1 = "Normal" consumption order (pages, frames).
2 = chronological -- E.g., Usenet news articles.
3 = semantic size; E.g., increasingly comprehensive abstracts.
4 = generality, E.g., thesaurus words, increasing generality; concentric object snapshots, zoom-out order.
5 = Elements explicitly undistinguished by order.
6 = undefined; may (or not) be ordered by private agreement.
7 = singleton; Never more than one occurrence.

defaultTagType The tag type that applies for any element for which tag type is not included.

defaultVariantSetId The Variant set identifier that applies when the target returns a variant specification for an element, but does not include a variant set identifier.

defaultVariantSpec If this element is present, then the specified variant applies to all subsequent elements, when applicable, which do not include a variant specification.

processingInstructions Recommendation by the target on how to display this record to the user.

recordUsageRight

1 = redistributable
2 = restricted
3 = licensePointer

recordUsageStatement The target is telling origin to display this rights statement.

rank The rank of this record within the result set. If there are N records in the result set, each record should have a unique rank from 1 to N.

userMessage A message that the target asks the origin to display to the user, regarding this record.

url Uniform resource locator.

record This element may be used for nested records. Also, may be used to wrap a whole (real) record when there is no root, so that the origin may request the record skeleton.

localControlNumber An identifier of the record, unique within database.

creationDate Date that the record was created.

dateOfLastModification The most recent date that this record was modified.

dateOfLastReview The most recent date that this record was verified.

score A normalized score assigned to the record by the target. Each record in the result set should have a score from 1 to N where N is the normalization factor (more than one record may have the same score).

wellKnown When an element is defined to be "structured into locally defined elements", the target may use this tag in lieu of locally defined tags.

BodyOfDisplay The target might combine several elements into this single element, into a display format, for display.

UpdateStatus For use by target, when presenting a DatabaseUpdate ES package. It is the update status of an individual database record within the retrieved task package.

SCH.2.2 Definition of tagSet-2

Element	tag	recommended ASN.1 datatype
title	1	GeneralString
name	2	GeneralString
date	3	GeneralizedTime

Appendix ERS

Extended Result Set Model

(Non-Normative)

Section 1.2 (Model of a Result Set) notes that in the extended result set model for searching, the target maintains unspecified information associated with each record, which may be used as a surrogate for the search that created the result set. Query specifications may indicate under what condition the extended model applies and the nature of the unspecified information. This appendix provides examples of information that the target might maintain to perform proximity operations requiring the extended model, or to evaluate retstriction operands.

1. Extended Result Set Model for Proximity

In the extended result set model for proximity, the target maintains information associated with each record represented by the result set, that may be used in a proximity operation as a surrogate for the search that created the result set.

Example:

Let R1 and R2 be result sets produced by Type-1 query searches on the terms 'cat' and 'hat'. In the extended result set model for proximity, the target maintains sufficient information associated with each entry in R1 and with each entry in R2 so that the proximity operation "R1 near R2" would be a result set equivalent to the result set produced by the proximity operation "cat near hat" ("near" is used informally to refer to a proximity test).

The manner in which the target maintains this information is not prescribed by the standard. The concept of "abstract position vectors" may used to describe the effect of the proximity test. A target system may implement the proximity test in any way that produces the desired results.

An abstract position vector might include a proximity unit and a sequence of position identifiers.

Example: Let R1 and R2 be result sets produced by searches on the terms 'cat' and 'hat'. Record 1000 contains 'cat' in paragraphs 10 and 100 and 'hat' in paragraphs 13 and 200. So record 1000 is represented in both R1 and R2. In R1, it might include the two position vectors (paragraph, 10) and (paragraph, 100). In R2, it might include the two position vectors (paragraph, 13) and (paragraph, 200). R3 = "R1 within 10 paragraphs of R2" would identify this record, and a position vector might be created (paragraph, 10, 13).

Subsequently, suppose R4 represents "rat before bat" and includes record 1000 with position vectors (paragraph, 5, 8) and (paragraph, 15, 18). Then:

- R3 'before and within 2 of' R4 would represent: "(cat near hat) before (rat before bat)" and in the resulting set, record 1000 might include position vector (paragraph, 10, 18);
- R3 'following and within 2 of' R4 might represent: "(cat near hat) after (rat before bat)" and in the resulting set, record 1000 might include position vector (paragraph, 5, 13).

Note: In these two examples, the position vectors might instead be (paragraph, 10, 13, 15, 18) instead of (paragraph, 10, 18); and (paragraph, 5, 8, 10, 13) instead of (paragraph, 5, 13). Different implementations might interpret extended proximity tests differently.

Neither the information that the target maintains (associated with result set entries to be used in the proximity operations) nor the manner in which the target maintains this information, is prescribed by the standard. The above is supplied as an example only.

2. Extended Result Set Model for Restriction

The Restriction operand specifies a result-set-id and a set of attributes. It might represent a set of database records identified by the specified result set, restricted by the specified attributes, as in example 1. It might represent a set of records from the database specified in the Search APDU, indirectly identified by the specified result set and restricted by the specified attributes, as in example 2.

Example 1:

Let R be the result set produced by a search on the term 'cat'. Result set position:

1 identifies record 1000, where 'cat' occurs in the title.
2 identifies record 2000, where 'cat' occurs in the title and as an author.
3 identifies record 3000, where 'cat' occurs in the title, as an author, and subject.

Then "R restricted to 'author'" might produce the result set consisting of the entries 2 and 3 of R.

In the extended result set model for restriction, the target maintains information that allows this type of search to be performed. In this example, the target might maintain the following information with the entries in result set R:

Result set position:

1 title
2 title, author
3 title, author, subject

Example 2:

In this example, there are two databases, R and C. R is a "registry" database containing records about chemical substances, each of which is identified by a unique registry number. C is a bibliographic database, containing bibliographic records for documents about chemical substances. The registry number is a searchable field in both databases. A registry number identifying a record in R may occur in one or more logical indexes for database C. For example, the "preparations" index for database C contains registry numbers of substances which are cited in its documents as being used in preparations.

In this example, a search is performed against database R, creating result set L, which will, in effect, contain registry numbers, representing records in database R, each of which uniquely identifies a chemical substance. A second search is performed against database C with the operand "L restricted to 'preparations'". This restriction is expressed by applying the "preparations" attribute to result set L. The search is performed by looking for registry numbers from result set L which occur in the "preparations" index for database C. The result set represents the records in C where a registry number contained in result set L occurs as a preparation.

In the extended result set model for restriction, the target maintains information that allows this type of search to be performed. In this example, the target might maintain, with each entry in L, a list of identifiers of records in C for which the registry number occurs as a preparation.

Neither the information that the target maintains (associated with result set entries to be used in the evaluation of a Restriction operand), nor the manner in which the target maintains this information, is prescribed by the standard. The above are supplied as an example only.

Appendix RET

Z39.50 Retrieval

(Non-normative)

Search and retrieval are the two primary functions of Z39.50. Searching is the selection of database records, based on origin-specified criteria, and the creation by the target of a result-set representing the selected records. Retrieval is, idiomatically speaking, the transfer of result set records from the target to the origin. This appendix addresses retrieval, and assumes the existence of a result set.

RET.1 Overview of Z39.50 Retrieval

Though retrieval is considered informally to be the transfer of "result set records", a result set logically does not contain records. Rather, it contains logical items; each item includes a pointer to a database record (the term "result set record" is an idiomatic expression used to mean "the database record represented by a result set item").

Moreover, a database record, as viewed by Z39.50, is purely a local data structure. In general Z39.50 retrieval does not transfer database records (that is, the target does not transfer the information according to its physical representation within the database), nor does Z39.50 necessarily transfer all of the information represented by a particular database record; it might transfer a subset of that information.

Thus the "transfer of a result set record" more accurately means: the transfer of some specified subset of the information in a database record (represented by that result set entry) according to some specified format. This exportable structure transferred is called a retrieval record. (Multiple retrieval requests for a given record may result in significantly different retrieval records, both in content and structure.)

Z39.50 retrieval thus allows the origin to specify:

(a) what subset of the information is desired, i.e. which logical information elements, and

(b) how those elements are to be represented, or formatted, for transfer.

Z39.50 retrieval thus has two primary functions, selection and representation:

- The selection of logical information elements from a database record.
  Note: "selection", with respect to retrieval, should not be confused with searching, which selects database records. Selection in retrieval pertains to selection of information elements from already-selected database records.
- The representation of those selected elements within a retrieval record.

"Representation" applies both individually to elements, and collectively to the collection of elements comprising a record. The origin may specify how individual element are to be formatted (via variants, described below) and how the collection of elements is to be packaged into a retrieval record (via a record syntax, described below). Thus representation is further classified as representation of element content or record formatting.

An auxiliary retrieval function is element identification. It is required both for selection and for record representation.

Summarizing, the primary retrieval functions are:

1. element selection,
2. representation of element content,
3. record formatting,

and an auxiliary function is:

4. element identification, required for
   - element selection, and
   - record formatting.

RET.2 Retrieval Objects

This appendix describes five object classes used for the functions listed above: schema definitions, tagSets, element specifications (elementSpecs), variant specifications (variantSpecs) and record syntaxes. This section provides an overview of these objects.

An elementSpec occurs within a Z39.50 Present request, and is used primarily for selection. In its most basic form, an elementSpec is a request for specified elements, and it may be viewed as a set of element requests (elementRequests). However, an elementSpec might have representation aspects: each elementRequest may include a variantSpec, i.e. a variantRequest.

> Definitional Note: A variantSpec occurring within an elementRequest is called a variantRequest. A variantSpec applied to an element by the target (when that element is included in a retrieval record) is called an appliedVariant. The target might provide a listing for a given element of the variantSpecs supported for that element; each is referred to as a supportedVariant.

A variantRequest is used primarily for element representation, to specify how an element is to be formatted. However, a variantRequest may have selection aspects: it might ask for a selected "piece" or fragment of an element.

A record syntax is applied to the set of elements selected by an elementSpec (and possibly transformed by appliedVariants) resulting in a retrieval record.

A tagSet provides names (and recommended data types) for a set elements. The names may be used in an elementRequest or within a record syntax, for element identification. A schema definition refers to a tagSet definition when defining an "abstract record structure" (see RET.4).

Summarizing:

- An elementSpec is used for element selection, and for representation of element content via variantRequest;
- A record syntax is used for record formatting;
- A tagSet is used for element identification, both within an elementSpec (for element selection) and a record syntax (for record formatting).
- A schema defines an abstract record structure.

RET.3 Element Specification Features

An elementSpec is included within a Present request to specify the desired elements to comprise a retrieval record. For example, the origin might wish to request the "author" and "title". The elementSpec, included in the Present request is used to express this, in one of two ways:

(a) A primitive name might be defined, for example "authorTitle", whose (static) definition means "return the author and title". This method has a significant limitation: an element set name must be defined for every possible combination of elements that might be requested.

(b) A more flexible specification might be used, allowing the origin to select elements dynamically rather than rely on a static definition.

For Z39.50 version 2, only (a) is allowed: the elementSpec must be an "element set name", whose ASN.1 type is VisibleString. Version 3 supports (b) by allowing the elementSpec to alternatively assume the ASN.1 type EXTERNAL. In that case, an external definition is used, which is presumably (though not necessarily) described in ASN.1.

The following illustrate some of the features that may be provided by an elementSpec, by progressively complex ASN.1 examples.

RET.3.1. Simple numeric tags

A simple elementSpec might specify a list of elements. The elementSpec definition could be:

```
ESpec            ::= SEQUENCE OF ElementRequest
ElementRequest   ::= INTEGER
```

Each element requested is represented by an integer; this is where the tagSet comes into play: A tagSet might be referenced by the element specification which might assign 1 to "title" and 2 to "author".

RET.3.2 String tags

In some applications it is not desirable to restrict element tags to integers. "String" tags are also useful. So the elementSpec might take the slightly more complex form:

```
ESpec ::=  SEQUENCE OF ElementRequest
ElementRequest ::= CHOICE{
                        numeric     [1] IMPLICIT INTEGER,
                        string      [2] IMPLICIT GeneralString}
```

In this case, the tagSet might declare that "author may also referenced by the string tag 'author', and title by 'title'".

RET.3.3 Tag Types

Often it will be necessary (or useful) to request several elements not all of which are defined by the same tagSet. The ability to do this presents an important benefit: the developer of a tagSet need not re-define elements already defined by another tagSet. However, it also presents a complexity: a tag must be "qualified" by reference to its tagSet.

This is accomplished via "tag-type". Though tagSets are registered objects and thus identified by object identifiers, a schema definition may assign an integer to a tagSet as short-hand identifier. The integer is called a "tagType", and it identifies the tagSet only within the context of a schema that assigns it. Schemas are described below.

Extending the above example to support tag types, the elementSpec might be defined as:

```
ESpec ::=  SEQUENCE OF ElementRequest
ElementRequest ::= SEQUENCE{
            tagType         [1] IMPLICIT INTEGER,
            tagValue        [2] CHOICE{
                                    numeric     [1] IMPLICIT INTEGER,
                                    string      [2] IMPLICIT GeneralString}
```

RET.3.4 Tag Occurrence

A database record often contains recurring elements. An origin might want to request the Nth occurrence of a particular tag (e.g. "the fourth image"). To introduce recurrence into the above example, the elementSpec might be as follows:

```
ESpec ::=  SEQUENCE OF ElementRequest
ElementRequest ::= SEQUENCE{
        tagType             [1] IMPLICIT INTEGER OPTIONAL,
        tagValue            [2] CHOICE{
                                    numeric     [1] IMPLICIT INTEGER,
                                    string      [2] IMPLICIT GeneralString}
        tagOccurrence       [3]  IMPLICIT INTEGER}
```

RET.3.5 Tag Paths

A database record is not always a flat set of elements. Often a database record is a hierarchical structure, where leaf-nodes contain information. An origin might request, for example "the fourth paragraph of section 3 of chapter 2 of book 1"; 'book', 'chapter', 'section', and 'paragraph' might be tags (defined by the schema). This example introduces the concept of a tag path, which is simply a nested sequence of tags (and each would be qualified by a type and occurrence). A tag path can be incorporated by replacing the second line of ASN.1 in the previous example, with:

ElementRequest ::= TagPath
TagPath ::= SEQUENCE OF SEQUENCE{

RET.3.6 VariantRequests

Finally, the origin may wish to qualify one or more elementRequests with a variantRequest, to specify a particular composition (e.g. PostScript), encoding, language, character set, special formatting (e.g. line length), or fragment.

```
ESpec ::=  SEQUENCE OF ElementRequest
ElementRequest ::= SEQUENCE{
                        TagPath,
                        VariantRequest OPTIONAL}
```

Where TagPath is defined as in the previous example. Variants are described in RET.5.

RET.4 Schemas and TagSets

The primary component of a schema definition is an abstract record structure, ARS. It describes an abstract structure for a database record, in terms of a set of schema elements, and supplies information associated with each element, including whether it is mandatory, whether it is repeatable, and a definition of the element. It also describes the hierarchy of a record, if any (see RET.4.5); however in the following, simple example, the database record is flat.

Abstract Record Structure

Schema Element	Mandatory?	Repeatable?
headline	yes	No
name	no	Yes
date	no	No
score	no	No
recordIdentifier	no	No
objectElement	yes	No

Definitions

headline: A set of words that conveys the main idea of the record.
name: One or more individuals associated with the object element; it could, for example, be an author or an organization.
date: A date associated with the record.
score: Represents the numerical score of the record based on its relevance to the query.
recordIdentifier: An identifier of the record unique within the target system.
objectElement: contains object information for the record. It may be text, image, etc.

RET.4.1 TagSets

The schema must also provide an identifier for each element, so that it may be referenced in an element request or within a retrieval record. It might provide a "tag" for each element, either numeric or string. The schema could define the following tagSet:

Tag	Element	Recommended dataType
1	headline	GeneralString
2	name	GeneralString
16	date	GeneralizedTime
18	score	INTEGER
14	recordIdentifier	GeneralString
<locally defined string tag>	objectElement	GeneralString or OCTET STRING

In this example, for objectElement, the schema would indicate that the target is to assign some descriptive string tag. For example, if the element is a fingerprint file, the tag might be 'fingerPrintFile'. (In that case, the content of element "name" (tag 2) might identify the person who is the subject of the finger prints.) Since it is the only element in the ARS with a string tag, the origin will recognize it as the objectElement.

A schema may define a tagSet, and it need not be registered. The schema could simply assign an integer tagType to identify the tagSet. The tagSet could then be used only by that schema.

Some of the elements in the above example schema might also be included in another schema. For example, another schema might also define headline and name, and it should be able to use the same tags for these. For this purpose, tagSets may be registered, independent of schema definitions.

It is anticipated that there will be more than one, but not a large number of tagsets defined, and that many, if not all schemas will be able to define an ARS referencing one or more registered tag sets, without the need to define a new tagset. There will be more that one tagSet defined because it would be difficult to manage a single tagSet that meets the needs of all schemas.

RET.4.2 TagTypes

Thus on a Present request (to identify requested elements for retrieval) or in a Present response (within a retrieval record) elements are identified by their tag, and the tag is qualified by a tag type (an integer), identifying the tagSet to which it belongs. (The schema lists each tagSet referenced in its ARS and designates an integer to be used as the tag type for that tagSet.)

Z30.50 currently defines two tagSets (tagSet-1 and tagSet-2). Among the schema elements defined in the example above, headline and name are defined in tagSet-2; date, score, and recordIdentifier are defined in tagSet-1. (TagSet-2 does not define any explicit string tags, and therefore allows locally defined string tags, thus objectElement can be represented from tagSet-2).

The schema might provide the following mapping of tagType to tagSet:

1 --> tagSet-1
2 --> tagSet-2

In the notation below, where (x,y) is used, 'x' is the tagType and 'y' is the tag: The left column in the table above would be re-written as follows.

Tag
(2,1)
(2,2)
(1,16)
(1,18)
(1,14)
(2,<locally defined string tag>

RET.4.3 Recurring objectElement

The schema becomes only slightly more complex if multiple "object elements" (i.e. multiple occurrences of the element objectElement) are allowed. The schema could indicate that each occurrence of objectElement is to have a different string tag. (And of course the entry in the "repeatable" column in the ARS, for objectElement, would be changed from 'no' to 'yes'.)

For example, suppose a record includes a fingerprint file, photo, and resume, all describing an individual (and the element "name" might identify the individual that they describe). The string tags for these three elements respectively might be "fingerPrint", "photo", and "resume". The origin would recognize each of these elements as an occurrence of objectElement, because the schema designates that only objectElement may have a string tag. (This is not intended to imply that the origin would recognize the type of information, e.g. fingerprint, from its string tag; but the origin might display the string tag to the user, to whom it might be meaningful.)

RET.4.4 Arbitrary String Tags

However, a schema might not want to impose such a restriction, i.e. that all string tags are object elements. It might want to let the target send arbitrary locally defined elements (with string tags) that it feels might be useful; if the target does not recognize the tag it may ignore the element, or simply display the tag and value for the user. For instance, in the above example, the target might want to include an author (if applicable) with the string tag "author".

To allow this, the objectElement tags must be distinguishable from the arbitrary tags. One way to do this is for the schema to designate a specific tagType for all object elements.

The mapping of tagType to tagSet could be:

1 --> tagSet-1
2 --> tagSet-2 and arbitrary local elements
3 --> object elements

The ARS would be:

Tag path	Element	Mandatory?	Repeatable?
(2,1)	Headline	yes	No
(2,2)	Name	no	Yes
(1,16)	Date	no	No
(1,18)	Score	no	No
(1,14)	RecordIdentifier	no	No
(3,<stringTag>)	ObjectElement	yes	Yes

RET.4.5 Structured Elements

In the next example hierarchy is introduced; the ARS includes structured elements. In the examples above the ARSs are flat; all elements are "data" elements, that is, all are leaf-nodes.

The ARS below is part of a schema for a database in which each record describe an information resource. The notation (x,y)/(z,w) is used to mean the element (z,w) is a sub-element of the element (x,y). In the "Schema Element Name" column, indentation is used to indicate subordination. For example, distributorName, a data element, is a sub-element of the structured element distributor, which in turn is a sub-element of the structured element availability.

Abstract Record Structure:

Schema Element Tag Path	Schema Element Name
(2,50)	title
(2,3)	abstract
(2,51)	purpose
(2,52)	originator
(2,70)	availability
(2,70)/(2,90)	distributor
(2,70)/(2,90)/(2,4)	distributorName
(2,70)/(2,90)/(2,5)	distributorOrganization
(2,70)/(2,90)/(2,6)	distributorAddress
(2,70)/(2,90)/(2,13)	distributorTelephone
(2,70)/(2,55)	orderProcess
(2,70)/(2,25)	linkage
(2,94)	pointOfContact
(2,94)/(2,4)	contactName
(2,94)/(2,5)	contactOrganization
(2,94)/(2,6)	contactAddress
(2,97)	crossReference
(2,97)/(2,50)	crossReferenceTitle
(2,97)/(2,25)	crossReferenceLinkage
(2,27)	recordSource

The ARS describes an abstract database record consisting of title, abstract, purpose, originator, availability, point of contact, crossReference, and recordSource. These are the "top-level" elements, among which, Availability, pointOfContact, and CrossReference are structured elements, and the others are data elements. Availability consists of distributor, orderProcess, and Linkage; among these, distributor is a structured element.

In conjunction with this ARS, it may be necessary to define a tagSet, if all of the above element cannot be referenced from other, known tagSets. An example tagSet that would support this ARS follows:

Tag	Element Name	Recommended DataType
3	abstract	GeneralString
4	name	GeneralString
5	organization	GeneralString
6	address	GeneralString
13	telephone	GeneralString
25	linkage	GeneralString
27	recordSource	GeneralString
50	title	GeneralString
51	purpose	GeneralString
52	originator	GeneralString
70	availability	(structured)
90	distributor	(structured)
94	pointOfContact	(structured)
97	crossReference	(structured)

Note that the element "name" (4) as defined in this tagset is referenced in the ARS both within the structured elements distributor (as distributorName) and PointOfContact (as contactName). Several other of these tagSet elements are similarly used in multiple structures.

RET.5 Variants

An element might be available for retrieval in various forms, or "variants". The concept of an element variant applies in three cases:

- the origin may request an element (in a Present request) according to a specific variant;
- the target may present an element (in a Present response) according to a specific variant;
- the target may indicate what variants of a particular element are available.

Correspondingly, and more formally, a "variant specification" or "variantSpec" takes the form of a variantRequest, appliedVariant, or supportedVariant. In all cases, a variantSpec is a sequence of variantComponents, each of which is a triple (class, type, value). 'class' is an integer. 'type' is also an integer and a set of types are defined for each class. Values are defined for each type.

A variantSet definition is a registered object (whose object identifier is called a variantSetId) which defines a set of classes, types, and values that may be used in a variantComponent. A variantSpec is always qualified by its variantSetId, to provide context for the values that occur within the variantComponents (in the same manner that an RPN Query includes an attribute set id, to provide context for the integers within the attribute lists).

The variant set definition Variant-1 is defined within the Z39.50 standard, and the remaining discussion of variants is based on variant-1. It is also assumed that whatever eSpec is used allows a variantRequest to accompany any elementRequest (as in eSpec-1), and whatever record syntax is used allows an appliedVariant to accompany a returned element (as in GRS-1).

RET.5.1 Format

Within Variant-1, class 2 is "format". There are three types: mime-type, mime-subtype, and non-mime. A origin may request an element according to a particular mime-type and subtype by supplying two variant components, for example, if PostScript is desired, the variant would include the following two component:

(2, 1, application)
(2, 2, postscript)

The origin might wish to request an element in "text" format (i.e. text mime type), without specifying the subtype (i.e. "plain", "richtext", etc.) but rather leave that decision to the target. In that case it could simply supply a single component:

(2,1, text)

When the target presents the element it could indicate which subtype applies. For example, if "richtext" is used, the target could include the following two components within the appliedVariant for the element:

(2, 1, text)
(2, 2, richtext)

An origin may requests an element in postscript, with pdf compression, by supplying the postscript component pair above, along with the pdf component pair:

(2, 1, application)
(2, 2, postscript)
(2, 1, dx)
(2, 2, pdf)

RET.5.2 Auxiliary Formatting

Class 3 includes several "auxiliary formatting" types, allowing specification of language, character set, line length, lines per page, etc. A variantRequest for "PostScript with PDF compression, portuguese version, portuguese character set, line length 80 characters, 56 lines per page" may be expressed using the following variantComponents:

(2, 1, application)
(2, 2, postscript)
(2, 1, dx)
(2, 2, pdf)
(3, 2, por) -- 3 character language code for Portuguese
(3, 3, 84) -- ISO registration code for Portuguese character set
(3, 5, 80)
(3, 6, 56)

RET.5.3 Fragments

The target might return a fragment of an element; i.e. the returned portion either does not starts at the beginning, or does not end at the end of the element. The target would indicate, for example, that the fragment starts at the beginning, and is not the whole element, by indicating "type of fragment: section; what section: start" using the following component:

(4, 1, 1)
(4, 2, 1)

In addition, the target may supply a token, that the origin may subsequently use to refer to that fragment, using the following component:

(4, 7, xyz)

Where'xyz' is an opaque token, in the form of an OCTET STRING.

The origin can then request, say, the next fragment, using:

(4, 1, 2) -- type of fragment: "relative fragment"
(4, 3, 1) -- what relative fragment: "next"
(4, 7, xyz) -- relative to what fragment: "xyz"

An origin may have discovered (see supplemental variant information and metaData below) that a particular element is large, say 1000 pages, and wishes to retrieve the first ten pages. It may include the following components:

(4, 1, 3) -- piece
(4, 4, 10 pages) -- "10 pages" specified by IntUnit
(4, 5, 1) -- beginning with first page
(4, 6, 1) -- every page (i.e. don't skip)

RET.5.4 Supplemental Variant Information and MetaData

The origin may use variant-1 to discover the elements of a record (without retrieving the actual data) as well as information associated with the retrieval of the elements, for example, the cost to retrieve an element, or the size of an element.

The origin may retrieve the "skeleton" of a record (the record structure and the element tags but no data) by requesting the whole record (see the description of GRS-1) and specifying "noData" via variant-1 Class 6, Return Metadata, using the following component:

(6, 1, NULL)

After discovering the elements, the origin might issue a request for one or more elements, again without data, but requesting a list of all supportedVariants for each requested element, by including:

(6, 2, NULL)

In addition, the cost of each element, and its size may be requested. By including the following four components, the origin may request (for each requested element) no data, list of supportedVariant list, and for each variant both the cost to retrieve the element in that variant, and its size:

(6, 1, NULL)
(6, 2, NULL)
(5, 1, NULL)
(5, 2, NULL)

The target subsequently returns the list of availableVariant (via ElementMetaData in GRS-1) each variant listed may include the following components:

(5, 1, cost)
(5, 1, size)

RET.5.4.1 Hits

The origin may ask that the target return "hits" information for an element (where the hits are associated with the search that caused the element to be included in the result set). The variantRequest would include the component:

(6, 3, NULL)

The target returns Hits information within ElementMetaData in GRS-1. Each "hit" may be considered a fragment, and is represented by a "hit vector" consisting of: source word, location of fragment within (where, within the element, it starts, and its length), rank (1 through N, where there are N hits) and a token that the origin may subsequently use to refer to that fragment. Note that a source word might occur in several hit vectors, with different ranks.

Hit vectors might be returned along with data, or without data. In the latter case, the origin might subsequently use the hit vectors to determine which fragment(s) to retrieve, and use information within the hit vectors to retrieve the fragments -- either the tokens or the location information.

Note that a hit vector might be variant-dependent or variant-independent. If the hit vector includes location information (start and length of fragment) then it pertains to a specific variant, and the target should indicate the "appliedVariant" within the GRS-1 record. However, if the location information is omitted, and instead a targetToken is supplied, and if the appliedVariant is also omitted within GRS, then the hit vector is variant-independent.

HitVector may also be included along with data, in which case the origin might use them to highlight hits for display.

RET.6 Record Syntax

The target applies a record syntax to an abstract database record, forming a retrieval record. Record syntaxes fall into two categories: content-specific and generic. Content-specific record syntaxes include those of the MARC family, Explain, OPAC, and Summary (see Appendix REC). Generic record syntaxes are further categorized: they are "structured" (e.g. GRS-1) or "unstructured" (e.g. SUTRS).

Structured record syntaxes are able to identify TagSet elements. This section addresses structured, generic record syntaxes. Following is simple example.

```
GenericRecord ::= SEQUENCE OF SEQUENCE{
    tagType            ...........
    tagValue           ...........
    tagOccurrence      ...........
    elementData        ...........
    metaData           ...........   -- e.g. hits
    appliedVariant     ...........
```

The tagType identifies a specific tagSet, and tagValue identifies a tag within that set. Recall in the examples above illustrating tagSets, for each data element (i.e. non-structured element) there is a recommended dataType. So "ElementData" should be defined such that it may accommodate the possible data types listed in the various tagSets. The ASN.1 definition must anticipate the ASN.1 type of the element. This would be a trivial matter if elementData could assume the ASN.1 type ANY, but ANY is no longer considered to be a valid ASN.1 type. Therefore the ASN.1 definition of elementData must list (as a CHOICE) the possible types that the element might assume. For example, it might be defined as:

```
ElementData ::= CHOICE{
        os              OCTET STRING,
        int             INTEGER,
        date            GeneralizedTime,
        ext             EXTERNAL,
        string          GeneralString}
```

The example above does not support structured records. To support structure, recursion is introduced as follows:

```
GenericRecord ::= SEQUENCE OF TaggedElement
TaggedElement ::= SEQUENCE {
        tagType             .....
        tagValue            .....
        tagOccurrence       .....
        elementData         .....
        metaData            .....
        appliedVariant      .....

    ElementData ::= CHOICE{
        os              OCTET STRING,
        int             INTEGER,
        date            GeneralizedTime,
        ext             EXTERNAL,
        string          GeneralString
        subtree         SEQUENCE OF TaggedElement -- Recursive, for nested tags.
                        }
```