4. Protocol Specification

The Information Retrieval application protocol specifies the formats and procedures governing the transfer of information between a Z39.50 origin/target pair. A unit of information, passed between two peer applications is called an "application protocol data unit" or APDU. Sections 4.1 and 4.2 respectively describe the formats (the Abstract Syntax) and rules (Protocol Procedures) for exchange of the Z39.50 APDUs. In addition, sections 4.3 and 4.4 respectively describe rules for extensibility and conformance requirements.

4.1 Abstract Syntax and ASN.1 Specification of Z39.50 APDUs

This section describes the abstract syntax of the Z39.50 APDUs, using the ASN.1 notation defined in ISO 8824. The comments included within the ASN.1 specification constitute part of the standard.

For all EXTERNAL types, whenever the EXTERNAL content is a structure whose abstract syntax is defined using ASN.1, "encoding" (within the ASN.1 definition of the EXTERNAL type in ISO 8824) shall select the CHOICE "single-ASN1-type".

```
Z39-50-APDU DEFINITIONS ::=  -- "Z39-50-APDU" registered in Appendix A.
BEGIN   -- ANSI Z39.50 V3D10 - 4/94
--
EXPORTS OtherInformation, Term, AttributeSetId, AttributeList, AttributeElement, ElementSetName,
SortElement, DatabaseName, Specification, OriginSuppliedCommonParameters, CharacterString, IntUnit,
Unit, StringOrNumeric, Query, Records, ResultSetId;
--
PDU ::= CHOICE{
 initRequest                      [20] IMPLICIT InitializeRequest,
 initResponse                     [21] IMPLICIT InitializeResponse,
 searchRequest                    [22] IMPLICIT SearchRequest,
 searchResponse                   [23] IMPLICIT SearchResponse,
 presentRequest                   [24] IMPLICIT PresentRequest,
 presentResponse                  [25] IMPLICIT PresentResponse,
 deleteResultSetRequest           [26] IMPLICIT DeleteResultSetRequest,
 deleteResultSetResponse          [27] IMPLICIT DeleteResultSetResponse,
 accessControlRequest             [28] IMPLICIT AccessControlRequest,
 accessControlResponse            [29] IMPLICIT AccessControlResponse,
 resourceControlRequest           [30] IMPLICIT ResourceControlRequest,
 resourceControlResponse          [31] IMPLICIT ResourceControlResponse,
 triggerResourceControlRequest    [32] IMPLICIT TriggerResourceControlRequest,
 resourceReportRequest            [33] IMPLICIT ResourceReportRequest,
 resourceReportResponse           [34] IMPLICIT ResourceReportResponse,
 scanRequest                      [35] IMPLICIT ScanRequest,
 scanResponse                     [36] IMPLICIT ScanResponse,
                                         -- [37] through [42] reserved
 sortRequest                      [43] IMPLICIT SortRequest,
 sortResponse                     [44] IMPLICIT SortResponse,
 segmentRequest                   [45] IMPLICIT Segment,
 extendedServicesRequest          [46] IMPLICIT ExtendedServicesRequest,
 extendedServicesResponse         [47] IMPLICIT ExtendedServicesResponse,
 close                            [48] IMPLICIT Close}
```

```
   InitializeRequest ::= SEQUENCE{
          referenceId                              ReferenceId OPTIONAL,
          protocolVersion                          ProtocolVersion,
          options                                  Options,
          preferredMessageSize                     PreferredMessageSize,
          maximumRecordSize                        MaximumRecordSize,
          idAuthentication                 [7]     ANY OPTIONAL, -- see note below
          implementationId                         ImplementationId OPTIONAL,
          implementationName                       ImplementationName OPTIONAL,
          implementationVersion                    ImplementationVersion OPTIONAL,
          userInformationField                     UserInformationField OPTIONAL,
|         otherInfo                                OtherInformation OPTIONAL}
   --Note:
|  -- For idAuthentication, the type ANY is retained for compatibility with earlier versions.
|  -- For interoperability, the following is recommended:
   --   IdAuthentication [7] CHOICE{
   --      open              VisibleString,
   --      idPass            SEQUENCE {
|  --                                groupId      [0]     IMPLICIT VisibleString OPTIONAL,
|  --                                userId       [1]     IMPLICIT VisibleString OPTIONAL,
|  --                                password     [2]     IMPLICIT VisibleString OPTIONAL },
   --      anonymous         NULL,
|  --      other             EXTERNAL    (May use access control formats here. See Appendix ACC)
   --                                          }
   InitializeResponse ::= SEQUENCE{
          referenceId                              ReferenceId OPTIONAL,
          protocolVersion                          ProtocolVersion,
          options                                  Options,
          preferredMessageSize                     PreferredMessageSize,
          maximumRecordSize                        MaximumRecordSize,
          result                           [12]    IMPLICIT BOOLEAN,
          implementationId                         ImplementationId OPTIONAL,
          implementationName                       ImplementationName OPTIONAL,
          implementationVersion                    ImplementationVersion OPTIONAL,
          userInformationField                     UserInformationField OPTIONAL,
          otherInfo                                OtherInformation OPTIONAL}
    ProtocolVersion  ::=      [3]   IMPLICIT BIT STRING{
|                version-1                         (0),    -- this bit should always be set, but does not
|                                                          -- correspond to any Z39.50 version
                 version-2                         (1),    -- "version 2 supported"
|                                                          -- this bit should always be set
                 version-3                         (2)     -- "version 3 supported"
   -- Values higher than 'version-3' should be ignored. Both the Initialize and Initialize Response APDUs
   -- include a value string corresponding to the supported versions. The highest common version is selected
   -- for use. If there are no versions in common, "Result" in the Init Response should indicate "reject".
   -- Note: Versions 1 and 2 are identical. Systems supporting version 2 should indicate support for version
   -- 1 as well, for interoperability with systems that indicate support for version 1 only (e.g. ISO 10163
   -- implementations).
                 }
```

```
Options  ::=      [4] IMPLICIT BIT STRING{
            search                          (0),
            present                         (1),
            delSet                          (2),
            resourceReport                  (3),
            triggerResourceCtrl             (4),
            resourceCtrl                    (5),
            accessCtrl                      (6),
            scan                            (7),
            sort                            (8),
                                              -- (9) reserved
            extendedServices                (10),
            level-1Segmentation             (11),
            level-2Segmentation             (12),
            concurrentOperations            (13),
            namedResultSets                 (14)}
   PreferredMessageSize      ::=   [5]     IMPLICIT INTEGER
   MaximumRecordSize         ::=   [6]     IMPLICIT INTEGER
   ImplementationId          ::=  [110]    IMPLICIT VisibleString
   ImplementationName        ::=  [111]    IMPLICIT VisibleString
   ImplementationVersion     ::=  [112]    IMPLICIT VisibleString
   UserInformationField      ::=  [11]     EXTERNAL

SearchRequest ::= SEQUENCE{
      referenceId                             ReferenceId OPTIONAL,
      smallSetUpperBound           [13]    IMPLICIT INTEGER,
      largeSetLowerBound           [14]    IMPLICIT INTEGER,
      mediumSetPresentNumber       [15]    IMPLICIT INTEGER,
      replaceIndicator             [16]    IMPLICIT BOOLEAN,
      resultSetName                [17]    IMPLICIT VisibleString,
      databaseNames                [18]    IMPLICIT SEQUENCE OF DatabaseName,
      smallSetElementSetNames      [100]   ElementSetNames OPTIONAL,
      mediumSetElementSetNames     [101]   ElementSetNames OPTIONAL,
      preferredRecordSyntax                PreferredRecordSyntax OPTIONAL,
      query                        [21]    Query,
            -- following two parameters may be used only if version 3 is in effect.
      additionalSearchInfo         [203]   IMPLICIT OtherInformation OPTIONAL,
      otherInfo                            OtherInformation OPTIONAL}

      Query  ::=  CHOICE
            {type-0       [0]     ANY,
             type-1       [1]     IMPLICIT RPNQuery,
             type-2       [2]     OCTET STRING,
             type-100     [100]   OCTET STRING,
             type-101     [101]   IMPLICIT RPNQuery}

      RPNQuery ::= SEQUENCE{
                  attributeSet  AttributeSetId,
                  rpn           RPNStructure}
```

```
RPNStructure ::= CHOICE{
        op              [0] Operand,
        rpnRpnOp        [1] IMPLICIT SEQUENCE{
                            rpn1        RPNStructure,
                            rpn2        RPNStructure,
                            op          Operator }}
--
    Operand ::= CHOICE{
        attrTerm        AttributesPlusTerm,
        resultSet       ResultSetId,
|               -- if version 2 is in effect:
|               --   - if query type is 1, one of the above two must be chosen;
|               --   - resultAttr (below) may be used only if query type is 101.
        resultAttr      ResultSetPlusAttributes
                        }
    AttributesPlusTerm ::= [102] IMPLICIT SEQUENCE{
                        attributes      AttributeList,
                        term            Term}
    ResultSetPlusAttributes ::= [214] IMPLICIT SEQUENCE{
                        resultSet       ResultSetId,
                        attributes      AttributeList}
    AttributeList ::=   [44]  IMPLICIT SEQUENCE OF AttributeElement
--
    Term ::= CHOICE{
        general         [45]   IMPLICIT OCTET STRING,
                                -- values below may be used only if version 3 is in effect
        numeric                 [215]  IMPLICIT INTEGER,
        simpleCharacterString   [216]  IMPLICIT GeneralString,
|       fancyCharacterString    [217]  IMPLICIT CharacterString,
|       external                [218]  IMPLICIT EXTERNAL,
|       integerUnit             [219] IMPLICIT IntUnit,
|       null                    [220] IMPLICIT NULL}    -- (220 is highest tag used)

    Operator ::= [46] CHOICE{
                        and             [0] IMPLICIT NULL,
                        or              [1] IMPLICIT NULL,
                        and-not         [2] IMPLICIT NULL,
|                                           -- if version 2 is in effect:
|                                           --   - for query type 1, one of the above three must be chosen;
|                                           --    - prox (below) may be used only if query type is 101.
                        prox            [3] IMPLICIT ProximityOperator
                                            }
    AttributeElement  ::=  SEQUENCE{
        attributeSet        [1]     IMPLICIT AttributeSetId OPTIONAL,
|                                           -- must be omitted if version 2 is in effect.
                                            -- if included, overides value of attributeSet
                                            -- in RPNQery above, but ONLY for THIS attribute.
        attributeType       [120]  IMPLICIT INTEGER,
        attributeValue      [121]  IMPLICIT INTEGER}
```

```
ProximityOperator ::= SEQUENCE{
        exclusion               [1] IMPLICIT BOOLEAN OPTIONAL,
        distance                [2] IMPLICIT INTEGER,
        ordered                 [3] IMPLICIT BOOLEAN,
        relationType            [4] IMPLICIT INTEGER{
                                        lessThan                (1),
                                        lessThanOrEqual         (2),
                                        equal                   (3),
                                        greaterThanOrEqual      (4),
                                        greaterThan             (5),
                                        notEqual                (6)},
        proximityUnitCode       [5] CHOICE{
                                        known       [1] IMPLICIT KnownProximityUnit,
                                        private     [2] IMPLICIT INTEGER}}
--
    KnownProximityUnit ::= INTEGER{
                                character       (1),
                                word            (2),
                                sentence        (3),
                                paragraph       (4),
                                section         (5),
                                chapter         (6),
                                document        (7),
                                element         (8),
                                subelement      (9),
                                elementType     (10),
                                byte            (11) -- Version 3 only
                                }


SearchResponse ::= SEQUENCE{
        referenceId                             ReferenceId OPTIONAL,
        resultCount                     [23]    IMPLICIT INTEGER,
        numberOfRecordsReturned                 NumberOfRecordsReturned,
        nextResultSetPosition                   NextResultSetPosition,
        searchStatus                    [22]    IMPLICIT BOOLEAN,
        resultSetStatus                 [26]    IMPLICIT INTEGER{
                                                        subset      (1),
                                                        interim     (2),
                                                        none        (3)} OPTIONAL,
        presentStatus                           PresentStatus  OPTIONAL,
        records                                 Records OPTIONAL,
|               -- following two parameters may be used only if version 3 is in effect.
        additionalSearchInfo            [203]   IMPLICIT OtherInformation OPTIONAL,
        otherInfo                               OtherInformation OPTIONAL}
```

```
PresentRequest ::= SEQUENCE{
        referenceId                             ReferenceId OPTIONAL,
        resultSetId                             ResultSetId,
        resultSetStartPoint             [30]    IMPLICIT INTEGER,
        numberOfRecordsRequested        [29]    IMPLICIT INTEGER,
        additionalRanges                [212]   IMPLICIT SEQUENCE OF Range OPTIONAL,
|                       -- may not be included if version 2 is in effect.
|       recordComposition                       CHOICE{
|                                                       simple          [19]    ElementSetNames,
|                                                               -- must choose 'simple' if version 2 is in effect
|                                                       complex         [209]   IMPLICIT CompSpec} OPTIONAL,
        preferredRecordSyntax                   PreferredRecordSyntax OPTIONAL,
        maxSegmentCount                 [204]   IMPLICIT INTEGER OPTIONAL, -- level 1 or 2
        maxRecordSize                   [206]   IMPLICIT INTEGER OPTIONAL, -- level 2 only
        maxSegmentSize                  [207]   IMPLICIT INTEGER OPTIONAL, -- level 2 only
        otherInfo                               OtherInformation OPTIONAL}
--
| --Segment PDU may only be used when version 3 is in effect, and only when segmentation is in effect.
Segment ::= SEQUENCE{
        referenceId                             ReferenceId OPTIONAL,
        numberOfRecordsReturned                 NumberOfRecordsReturned,
        segmentRecords                  [0]     IMPLICIT SEQUENCE OF NamePlusRecord,
        otherInfo                               OtherInformation OPTIONAL}
--
PresentResponse ::= SEQUENCE{
        referenceId                             ReferenceId OPTIONAL,
        numberOfRecordsReturned                 NumberOfRecordsReturned,
        nextResultSetPosition                   NextResultSetPosition,
        presentStatus                           PresentStatus,
        records                                 Records OPTIONAL,
        otherInfo                               OtherInformation OPTIONAL}
--
-- begin auxiliary definitions for search and response APDUs
   -- begin definition of records
  Records ::= CHOICE{
        responseRecords                 [28]    IMPLICIT SEQUENCE OF NamePlusRecord,
|       nonSurrogateDiagnostic          [130]   IMPLICIT DefaultDiagFormat,
|       multipleNonSurDiagnostics       [205]   IMPLICIT SEQUENCE OF DiagRec}
NamePlusRecord  ::=  SEQUENCE{
        name            [0] IMPLICIT DatabaseName OPTIONAL,
        record          [1] CHOICE{
                retrievalRecord         [1] EXTERNAL,
|               surrogateDiagnostic     [2] DiagRec,
                                            -- only the above two values, [1] and [2], may be
                                            -- used unless 'level 2 segmentation' is in effect
                startingFragment        [3] FragmentSyntax,
                intermediateFragment    [4] FragmentSyntax,
                finalFragment           [5] FragmentSyntax}}
```

```
|    DiagRec ::= CHOICE{
|                        defaultFormat               DefaultDiagFormat,
|                        externallyDefined           EXTERNAL}
        FragmentSyntax ::= CHOICE{
                externallyTagged            EXTERNAL,
                notExternallyTagged         OCTET STRING}
|       DefaultDiagFormat::= SEQUENCE{
                diagnosticSetId             OBJECT IDENTIFIER,
                condition                   INTEGER,
                addinfo                     VisibleString}
    -- end definition of records
  --
        Range  ::= SEQUENCE{
                startingPosition    [1]     IMPLICIT INTEGER,
                numberOfRecords     [2]     IMPLICIT INTEGER}
  --
|       ElementSetNames ::= CHOICE {
                generic             [0]     IMPLICIT ElementSetName,
                databaseSpecific    [1]     IMPLICIT SEQUENCE OF SEQUENCE{
                                                    dbName       DatabaseName,
                                                    esn          ElementSetName}}
        ElementSetName                      ::= [103]     IMPLICIT VisibleString

        PreferredRecordSyntax               ::= [104]     IMPLICIT OBJECT IDENTIFIER
        NumberOfRecordsReturned             ::= [24]      IMPLICIT INTEGER
        NextResultSetPosition               ::= [25]      IMPLICIT INTEGER
        PresentStatus        ::=    [27]    IMPLICIT INTEGER{
                                                    success      (0),
                                                    partial-1    (1),
                                                    partial-2    (2),
                                                    partial-3    (3),
                                                    partial-4    (4),
                                                    failure      (5)}
  -- being definition of composition specification
|  CompSpec ::= SEQUENCE{
        selectAlternativeFlag       [1] IMPLICIT BOOLEAN,
        generic                     [2] IMPLICIT Specification OPTIONAL,
        dbSpecific                  [3] IMPLICIT SEQUENCE OF SEQUENCE{
                                                    db     [1] DatabaseName,
                                                    spec   [2] IMPLICIT Specification} OPTIONAL,
                                            -- At least one of [2] and [3] must occur.
        recordSyntax                [4] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL}
  Specification ::= SEQUENCE{
     schema            [1] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
     elementSpec       [2] CHOICE{
                                primitive     [1]     IMPLICIT ElementSetName,
|                               external      [2]     IMPLICIT EXTERNAL} OPTIONAL}
  -- end definition of composition specification
  -- end auxiliary definitions for search and response APDUs
```

```
DeleteResultSetRequest ::= SEQUENCE{
        referenceId                                    ReferenceId OPTIONAL,
        deleteFunction                      [32]    IMPLICIT INTEGER{
                                                            list    (0),
                                                            all     (1)},
        resultSetList                                  SEQUENCE OF ResultSetId OPTIONAL,
        otherInfo                                      OtherInformation OPTIONAL}
--
DeleteResultSetResponse ::= SEQUENCE{
        referenceId                                    ReferenceId OPTIONAL,
        deleteOperationStatus               [0]     IMPLICIT DeleteSetStatus,
        deleteListStatuses                  [1]     IMPLICIT ListStatuses OPTIONAL,
        numberNotDeleted                    [34]    IMPLICIT INTEGER OPTIONAL,
        bulkStatuses                        [35]    IMPLICIT ListStatuses OPTIONAL,
        deleteMessage                       [36]    IMPLICIT VisibleString OPTIONAL,
        otherInfo                                      OtherInformation OPTIONAL}
--
        ListStatuses ::= SEQUENCE OF SEQUENCE{
                                id      ResultSetId,
                                status DeleteSetStatus}

        DeleteSetStatus ::= [33] IMPLICIT INTEGER{
                        success                                 (0),
                        resultSetDidNotExist                    (1),
                        previouslyDeletedByTarget               (2),
                        systemProblemAtTarget                   (3),
                        accessNotAllowed                        (4),
                        resourceControlAtOrigin                 (5),
                        resourceControlAtTarget                 (6),
                        bulkDeleteNotSupported                  (7),
                        notAllRsltSetsDeletedOnBulkDlte         (8),
                        notAllRequestedResultSetsDeleted        (9),
                        resultSetInUse                          (10)}
--
--
  AccessControlRequest ::= SEQUENCE{
        referenceId                     ReferenceId OPTIONAL,
        securityChallenge               CHOICE{
                                                simpleForm              [37] IMPLICIT OCTET STRING,
                                                externallyDefined       [0]  EXTERNAL},
        otherInfo                       OtherInformation OPTIONAL}

  AccessControlResponse ::= SEQUENCE{
        referenceId                     ReferenceId OPTIONAL,
        securityChallengeResponse       CHOICE{
                                                simpleForm              [38] IMPLICIT OCTET STRING,
                                                externallyDefined       [0]  EXTERNAL},
        otherInfo                       OtherInformation OPTIONAL}
```

```
ResourceControlRequest ::= SEQUENCE{
        referenceId                         ReferenceId OPTIONAL,
        suspendedFlag              [39]     IMPLICIT BOOLEAN OPTIONAL,
        resourceReport             [40]     ResourceReport OPTIONAL,
        partialResultsAvailable    [41]     IMPLICIT INTEGER{
                                                    subset      (1),
                                                    interim     (2),
                                                    none        (3)} OPTIONAL,
        responseRequired           [42]     IMPLICIT BOOLEAN,
        triggeredRequestFlag       [43]     IMPLICIT BOOLEAN OPTIONAL,
        otherInfo                           OtherInformation OPTIONAL}
--
ResourceControlResponse ::= SEQUENCE{
        referenceId                         ReferenceId OPTIONAL,
        continueFlag               [44]     IMPLICIT BOOLEAN,
        resultSetWanted            [45]     IMPLICIT BOOLEAN OPTIONAL,
        otherInfo                           OtherInformation OPTIONAL}
--
TriggerResourceControlRequest ::= SEQUENCE{
        referenceId                         ReferenceId OPTIONAL,
        requestedAction            [46]     IMPLICIT INTEGER{
                                                    resourceReport     (1),
                                                    resourceControl    (2),
                                                    cancel             (3)},
        prefResourceReportFormat   [47]     IMPLICIT ResourceReportId OPTIONAL,
        resultSetWanted            [48]     IMPLICIT BOOLEAN OPTIONAL,
        otherInfo                           OtherInformation OPTIONAL}
--
ResourceReportRequest ::= SEQUENCE{
        referenceId                         ReferenceId OPTIONAL,
        opId                       [210]    IMPLICIT ReferenceId OPTIONAL,
        prefResourceReportFormat   [49]     IMPLICIT ResourceReportId OPTIONAL,
        otherInfo                           OtherInformation OPTIONAL}
ResourceReportResponse ::= SEQUENCE{
        referenceId                         ReferenceId OPTIONAL,
        resourceReportStatus       [50]     IMPLICIT INTEGER{
                                                    success            (0),
                                                    partial            (1),
                                                    failure-1          (2),
                                                    failure-2          (3),
                                                    failure-3          (4),
                                                    failure-4          (5),
                                                    failure-5          (6),
                                                    failure-6          (7)},
        resourceReport             [51]     ResourceReport OPTIONAL,
        otherInfo                           OtherInformation OPTIONAL}
--
        ResourceReport          ::=     EXTERNAL
        ResourceReportId        ::=     OBJECT IDENTIFIER
```

```
   --Scan
   --
   ScanRequest ::= SEQUENCE{
     referenceId                                ReferenceId OPTIONAL,
     databaseNames                      [3]     IMPLICIT SEQUENCE OF DatabaseName,
|    attributeSet                               AttributeSetId OPTIONAL,
|    termListAndStartPoint                      AttributesPlusTerm,
     stepSize                           [5]     IMPLICIT INTEGER OPTIONAL,
     numberOfTermsRequested             [6]     IMPLICIT INTEGER,
     preferredPositionInResponse        [7]     IMPLICIT INTEGER OPTIONAL,
     otherInfo                                  OtherInformation OPTIONAL}

   ScanResponse ::= SEQUENCE{
    referenceId                                 ReferenceId OPTIONAL,
    stepSize                            [3]     IMPLICIT INTEGER OPTIONAL,
    scanStatus                          [4]     IMPLICIT INTEGER {
                                                        success     (0),
                                                        partial-1   (1),
                                                        partial-2   (2),
                                                        partial-3   (3),
                                                        partial-4   (4),
                                                        partial-5   (5),
                                                        failure     (6) },
    numberOfEntriesReturned             [5]     IMPLICIT INTEGER,
    positionOfTerm                      [6]     IMPLICIT INTEGER OPTIONAL,
    entries                             [7]     ListEntries  OPTIONAL,
    attributeSet                        [8]     IMPLICIT AttributeSetId OPTIONAL,
    otherInfo                                   OtherInformation OPTIONAL}

   -- begin auxiliary definitions for Scan
   ListEntries ::= CHOICE {
       entries                          [1]     IMPLICIT SEQUENCE OF Entry,
|      nonsurrogateDiagnostics          [2]     IMPLICIT SEQUENCE OF DiagRec}
   --
   Entry  ::= CHOICE {
        termInfo                        [1]     IMPLICIT TermInfo,
        surrogateDiagnostic             [2]     DiagRec}
   --
   TermInfo ::= SEQUENCE {
|     term                                      Term,
      suggestedAttributes                       AttributeList OPTIONAL,
|     alternativeTerm                   [4]     IMPLICIT SEQUENCE OF AttributesPlusTerm OPTIONAL,
      globalOccurrences                 [2]     IMPLICIT INTEGER OPTIONAL,
|     byAttributes                      [3]     IMPLICIT OccurrenceByAttributes OPTIONAL,
      otherTermInfo                             OtherInformation OPTIONAL}
```

```
OccurrenceByAttributes ::= SEQUENCE OF SEQUENCE{
   attributes              [1]    AttributeList,
   occurrences CHOICE{
               global      [2]    INTEGER,
               byDatabase  [3]    IMPLICIT SEQUENCE OF SEQUENCE{
                                         db                  DatabaseName,
                                         num            [1]  IMPLICIT INTEGER OPTIONAL,
                                         otherDbInfo         OtherInformation OPTIONAL}} OPTIONAL,
   otherOccurInfo                 OtherInformation OPTIONAL}
-- end auxiliary definitions for Scan


-- Sort
SortRequest  ::= SEQUENCE{
       referenceId                      ReferenceId OPTIONAL,
|      inputResultSetNames      [3]     IMPLICIT SEQUENCE OF VisibleString,
       sortedResultSetName      [4]     IMPLICIT VisibleString,
       sortSequence                     SortSequence,
       otherInfo                        OtherInformation OPTIONAL}
--
SortResponse  ::= SEQUENCE{
       referenceId                      ReferenceId OPTIONAL,
       sortStatus               [3]     IMPLICIT INTEGER{
                                                success      (0),
                                                partial-1    (1),
                                                failure      (2)},
       resultSetStatus          [4]     IMPLICIT INTEGER{
                                                empty        (1),
                                                interim      (2),
                                                unchanged    (3),
|                                               none         (4)} OPTIONAL,
|      diagnostics              [5]     IMPLICIT SEQUENCE OF DiagRec OPTIONAL,
       otherInfo                        OtherInformation OPTIONAL}

-- begin auxiliary definitions for Sort
       SortSequence ::=   [202] CHOICE{
         generic                        [1] IMPLICIT SortElementList,
         datbaseSpecific                [2] IMPLICIT SEQUENCE OF SEQUENCE{
                                           databaseName            DatabaseName,
                                           dbSort            [0]   IMPLICIT SortElementList}}
       SortElementList ::= SEQUENCE OF SortKey
           -- The order of occurence of the SortKeys within the sequence is from major to minor.
--
       SortKey ::= SEQUENCE{
         sortElement                    SortElement,
         sortRelation           [1]     IMPLICIT INTEGER{
                                                ascending          (0),
                                                descending         (1)},
```

```
        caseSensitivity          [2]    IMPLICIT INTEGER{
                                                caseSensitive       (0),
                                                caseInsensitive     (1)},
|     missingValueAction         [3]    CHOICE{
|                                         abort             [1] IMPLICIT NULL,
|                                         null              [2] IMPLICIT NULL,
|                                                               --supply a null value for missing value
|                                       missingValueData  [3] IMPLICIT OCTET STRING} OPTIONAL}

      SortElement ::= CHOICE{
              sortfield          [0]    IMPLICIT VisibleString,
                                        -- an element, element-group-tag, or alias supported by the target
|                                       -- and denoting a set of elements associated with each record.
|             elementSpec        [1]    IMPLICIT Specification,
|             sortAttributes     [2]    IMPLICIT SEQUENCE{
|                                               id      AttributeSetId,
|                                               list    AttributeList}}
-- end auxiliary definitions for sort


--
ExtendedServicesRequest  ::= SEQUENCE{
        referenceId                                  ReferenceId OPTIONAL,
        function                      [3]            IMPLICIT INTEGER {
                                                             create              (1),
                                                             delete              (2),
                                                             modify              (3)},
        packageType                   [4]     IMPLICIT OBJECT IDENTIFIER,
|       packageName                   [5]     IMPLICIT VisibleString OPTIONAL,
|                                                    -- mandatory for 'modify' or 'delete'
|       originCommonTaskParameters    [6]     IMPLICIT OriginSuppliedCommonParameters OPTIONAL,
|                                                    -- Mandatory for 'create'.  Should be included on
|                                                    --'modify' if any of the common parameters are being
|                                                    -- modified. Not needed on 'delete'.
|       originSpecificParameters      [7]     IMPLICIT EXTERNAL OPTIONAL,
|                                                    -- Similarly, mandatory for 'create'; included on
|                                                    -- 'modify' if specific parameters being modified; not
|                                                    -- necessary on 'delete'. For the 'EXTERNAL', use
|                                                    -- OID of Task package, and select CHOICE [1]:
|                                                    -- requestPackage.
|       waitAction                    [8]     IMPLICIT INTEGER{
|                                                    wait                (1),
|                                                    waitIfPossible      (2),
|                                                    dontWait            (3)},
        elements                              ElementSetName OPTIONAL,
        otherInfo                             OtherInformation OPTIONAL}
--
```

```
ExtendedServicesResponse ::= SEQUENCE{
        referenceId                                     ReferenceId OPTIONAL,
        operation-status            [3]                 IMPLICIT INTEGER{
                                                                done                (1),
                                                                queued              (2),
                                                                failure             (3)},
        diagnostics                 [4]                 IMPLICIT SEQUENCE OF DiagRec OPTIONAL,
        taskPackage                 [5]                 IMPLICIT EXTERNAL OPTIONAL,
        otherInfo                                       OtherInformation OPTIONAL}

OriginSuppliedCommonParameters ::= SEQUENCE{
        userId              [1] IMPLICIT VisibleString OPTIONAL,
        retentionTime       [2] IMPLICIT INTEGER OPTIONAL,
        permissions         [3] IMPLICIT SEQUENCE OF SEQUENCE{
                                userId                  [1]     IMPLICIT VisibleString,
                                allowableFunctions      [2]     IMPLICIT SEQUENCE OF INTEGER{
                                                        delete                  (1),
                                                        modifyContents          (2),
                                                        modifyPermissions       (3),
                                                        present                 (4),
                                                        invoke                  (5)} OPTIONAL},
        description         [4] IMPLICIT VisibleString OPTIONAL}



Close ::= SEQUENCE{
        referenceId                                     ReferenceId OPTIONAL,
        closeReason                                     CloseReason,
        diagnosticInformation               [3]         IMPLICIT VisibleString OPTIONAL,
        resourceReport                      [4]         ResourceReport OPTIONAL,
        otherInfo                                       OtherInformation OPTIONAL}
--
        CloseReason ::=     [211] IMPLICIT INTEGER{
                                        finished                (0),
                                        shutdown                (1),
                                        systemProblem           (2),
                                        costLimit               (3),
                                        resources               (4),
                                        securityViolation       (5),
                                        protocolError           (6),
                                        lackOfActivity          (7),
                                        peerAbort               (8),
                                        unspecified             (9)}
--
-- Global auxiliary definitions
        ReferenceId             ::=         [2]         IMPLICIT OCTET STRING
        DatabaseName            ::=         [105]       IMPLICIT VisibleString
        ResultSetId             ::=         [31]        IMPLICIT VisibleString
        AttributeSetId          ::=                     OBJECT IDENTIFIER
```

```
-- OtherInformation
       OtherInformation    ::= [201] IMPLICIT SEQUENCE OF SEQUENCE{
              category                            [1]   IMPLICIT InfoCategory OPTIONAL,
              information                    CHOICE{
                     characterInfo                [2]    IMPLICIT VisibleString,
                     binaryInfo                   [3]    IMPLICIT OCTET STRING,
                     externallyDefinedInfo        [4]    IMPLICIT EXTERNAL,
                     oid                          [5]    IMPLICIT OBJECT IDENTIFIER}}
--
       InfoCategory ::= SEQUENCE{
                     categoryTypeId     [1]    IMPLICIT OBJECT IDENTIFIER OPTIONAL,
                     categoryValue      [2]    IMPLICIT INTEGER}


-- Definition of Units
       IntUnit ::= SEQUENCE{
              value          [1] IMPLICIT INTEGER,
              unitUsed       [2] IMPLICIT Unit}
--
-- IntUnit is used when value and unit are supplied together. Unit, alone, is used when just specifying a
-- unit (without a value).  For example, IntUnit is used in Term, in an RPNQuery, or it can be the datatype
-- of an element within a retrieval record. Unit (alone) would be used in an element request, when
-- requesting data be returned according to a particular unit.
--
       Unit ::= SEQUENCE{
              unitSystem          [1] StringOrNumeric OPTIONAL,             -- e.g. 'SI'
              unitType            [2] StringOrNumeric OPTIONAL,             -- e.g. 'mass'
              unit                [3] StringOrNumeric OPTIONAL,             -- e.g. 'kilograms'
              scaleFactor         [4] IMPLICIT INTEGER OPTIONAL             -- e.g. 9 means 10**9
                             }

--CharacterString

CharacterString ::=  SEQUENCE{
          abstract-syntax      OBJECT IDENTIFIER,
          string-value         SEQUENCE{
                                 by     OBJECT IDENTIFIER,
                                 as     OCTET STRING}}

StringOrNumeric ::= CHOICE{
       string          [1] IMPLICIT GeneralString,
       numeric         [2] IMPLICIT INTEGER}

END --IR DEFINITIONS
```

4.2 Protocol Procedures

Section 4.2 specifies services required by the protocol, the protocol model, state tables, and handling of protocol errors. Sections 4.3 and 4.4 specify rules for extensibility, and conformance requirements, respectively.

4.2.1 Services Required

The Information Retrieval protocol may be used in conjunction with the presentation layer and the association control service element. If so, it assumes services as described in 4.2.1.1 and 4.2.1.2.

4.2.1.1 Service Provided by the Presentation Layer

The protocol may use the presentation service as defined in ISO 8822 to provide a presentation connection for communication between a Z39.50 origin/target pair. If so, this section describes the Presentation services required.

The presentation services required are those contained in the presentation kernel functional unit and the session duplex functional unit. (The association control service element may have additional requirements for presentation services.) All Information Retrieval protocol data units will be mapped onto the P-Data service. The communication service that supports this protocol is a connection-oriented service using the P-DATA service, defined in ISO 8822 in an established application association, in combination with the ACSE, ISO 8649.

A Z39.50 origin establishes application-associations as necessary with the target with whom it is engaged in Z39.50 activity. The Z39.50 application-service-element (ASE) may then use the P-DATA service defined in ISO 8822 directly to transmit Z39.50 APDUs. This provides a connection-oriented interaction between Z39.50 systems. A single application-association can be used to support a series of Z-Associations. A single system can be engaged in multiple application associations with multiple remote systems simultaneously.

4.2.1.2 Association Control Services

The complete application service may include the association control service element (ACSE), and one or more specific application services, such as the Information Retrieval application service.

ACSE, defined in ISO 8649, is used to establish an A-association, and provides association management. There are three distinct phases during the life of an A-association: establishment, information transfer, and termination. ACSE provides services for the establishment and termination phases, including the selection of an application context, specifying information including the set of service elements which are valid during the information transfer phase.

It is assumed that prior to APDUs being exchanged the Information Retrieval service user will handle the association control services required to establish an association with an application context encompassing the Information Retrieval service. The application context "basic-Z39.50-ac" is defined and registered in Appendix CTX.

4.2.2 Protocol Model

To specify protocol procedure, the abstract, implementation-independent concepts of service-user, service-provider, and service primitive are used.

A service-provider provides a communication path between two service users. In this model, the service-provider is analogous to the application layer composed of the Z39.50 origin/target pair. The client is modeled as a service-user together with an origin, and the server as a service-user together with a target. The two service users are referred to as the origin service-user and target service-user.

A service primitive is an element of interaction between a service-user and the service-provider. There are four types of service primitives: Request, Indication, Response and Confirmation. For a confirmed service initiated by the origin (i.e., for Z39.50: Init, Search, Present, Delete, Resource-report, Sort, Scan, Extended-services) they are used as follows:

1) Request - A primitive issued by the origin service-user to the service-provider in order to invoke some procedure.
2) Indication - A primitive issued by the service-provider to the target service-user to indicate that a procedure has been invoked by its peer.
3) Response - A primitive issued by the target service-user to the service-provider at the completion of the procedure previously invoked by an indication.
4) Confirmation - A primitive issued by the service-provider to the origin service-user to complete the procedure previously invoked by a request.

Notes:

(1) For a confirmed service initiated by the target (i.e., for Z39.50: Access-control and Resource-control) the roles of origin and target are reversed.

(2) For a non-confirmed service (i.e., for Z39.50: Segment, Trigger-resource-control, Close) only the Request and Indication primitives are used.

Primitives are conceptual and their use neither specifies nor precludes any specific implementation of a service. Only primitives that correspond to some element of the service involving the exchange of information between systems are defined.

From the perspective of the service-user, the service-provider is system-independent. For the exchange of protocol however, a distinction is made between those portions of the service-provider residing on the client and server (respectively, the origin service-provider and the target service-provider). The sequence of interactions for a confirmed service initiated by the origin is:

1) Request Primitive from origin service-user to service-provider.
2) Protocol Message from origin service-provider to target service-provider.
3) Indication Primitive from service-provider to target service-user.
4) Response Primitive from target service-user to service-provider.
5) Protocol Message from target service-provider to origin service-provider.
6) Confirmation Primitive from service-provider to origin service-user.

   Notes:

   (1) For a confirmed service initiated by the target, the roles of origin and target are reversed.

   (2) For a non-confirmed service, only steps 1 through 3 apply.

The following illustrates the sequence of interactions which occur for a Search operation:

1) Search request from origin service-user to service-provider.
2) Search APDU (Application Protocol Data Unit) from origin service-provider to target service-provider.
3) Search indication from service-provider to target service-user.
4) Search response from target service-user to service-provider.
5) Search-response APDU from target service-provider to origin service-provider.
6) Search confirm from service-provider to origin service-user.

Note: The interfaces between service user and service-provider, as represented by steps 1 and 6 for the client, and by steps 3 and 4 for the server, are described solely to facilitate the specification of protocols. These steps do not represent intersystem communication, and therefore, the means by which they are implemented are not constrained by this specification. In an actual implementation, step 4, for example, might consist of <u>several</u> messages from the target service user to service-provider. On the other hand, both the target service user and service-provider could be combined in a single program, in which case steps 3 and 4 might not have any real physical manifestation.

4.2.3 State Tables

This section defines Information Retrieval Protocol Machines (IRPMs) in terms of state tables. For both origin and target, there are three protocol machines defined, one for the Z-Association (called the "Z-machine") and two for Z39.50 operations (called "O-machine"s). There is one O-machine for a Present operation and one O-machine for any other type operation, excluding Init which is included in the Z-machine.

There is one instance of the Z-machine (within a given application association) each for the origin and target; there may be multiple concurrent instances of the O-machines.

Each state table shows the inter-relationship between the state of an operation or Z-Association, the incoming events that occur in the protocol, the actions taken, and, finally, the resulting state. The state tables do not constitute a formal definition of the IRPM. They are included to provide a more precise specification of the protocol procedures. The following conventions are used in the state tables.

State Table Cells The intersection of an incoming event (row) and a state (column) forms a cell. A blank cell represents the combination of an incoming event and a state that is not defined for the IRPM. A non blank cell represents an incoming event and state that is defined for the IRPM. Such a cell contains one or more actions, separated by semi-colons (;). The last such action specified is always a transition to the resulting state, in parenthesis.

Invalid Intersections Blank cells indicate an invalid intersection of an incoming event and state. The state tables define correct operation only. They do not specify actions to be taken in response to incorrect operation (for example, erroneous protocol control information, incorrect protocol control actions, etc.). Such actions are not within the scope of the specification, although implementations must consider them.

Predicates Some actions are predicated on a certain condition, or "predicate". The notation for these actions takes one of the following two forms:

":[predicate] {actions}:", or

":[predicate] {actions} else {actions}:"

Where {actions} is either a single action or multiple actions separated by semicolon. The following predicates are defined:

Predicate	Meaning
resp	"Response required" on a Resource Control PDU.
noResp	No "response required" on a Resource Control PDU.
conc	Concurrent operations in effect.
noOps	No active operations.

Variables The following variables are used:

Variable	Meaning
<op>	An operation type (other that Init): search, present, delete, scan, sort, resource-report, Extended-services.
opCnt	Number of active operations.
retSt	Return state. Takes on integer value; the action "(retSt)" means "go to the state whose value is retSt".

Notes pertaining to the tables:

(1) Access-control and resource-control events, actions, and states are distinguished according to whether they pertain to an operation or to the Z-association. (If concurrent operation is not in effect, all pertain to an operation. During initialization, all pertain to the initialization operation.) Those which pertain to an operation are reflected in the operation state table, except for those which

occur during initialization (those are shown in part 1 of the Z-association table) and those which pertain to an aborted operation (those apply to part 3 of the Z-association table). Those which pertain to the Z-association are shown in part 2 of the Z-association table (except as noted in notes 4 and 5). All abbreviations for states, events, and actions for access- or resource-control beginning with "Z-" (e.g. "Z-Acc PDU") pertain to the Z-association. All others (e.g. "Acc PDU") pertain to an operation.

(2) During initialization, access control or resource control requests may be received by the origin but only if the origin has indicated support (though this is not reflected in the state tables). The origin may not send Trigger-resource-control, because initialization is not complete so it has not yet been sucessfully negotiated. Neither the origin nor target may initiate Close during initialization.

(3) "End-operation indication" is a pseudo-action by the O-machine and corresponding event to the Z-machine. The O-machine issues the indication to the Z-machine, which receives it, also as an indication. Its meaning is that an operation has ended (it is necessary for the Z-machine to keep track of the number of active operations so that it will know whether there are zero, one, or multiple concurrent active operations).

(4) After the origin sends a Close PDU, PDUs may arrive that were sent before the target receives the Close PDU. When the origin is in "Close sent" state, it ignores all such PDUs if they pertain to an (aborted) operation. If an Access-control request pertaining to the Z-association is received, it is similarly ignored. However, if a Resource-control request pertaining to the application is received, and if it specifies that "no response is required" it is passed to the application, because it may include useful information. If a resource-control request specifies "response required" it is ignored.

(5) After the target sends a Close PDU, it ignores any received PDUs until it receives a Close PDU. When the target is in "Close Recvd" state, it may send one or more Resource-control requests before sending the Close PDU, but they must indicate "no response required".

Definition of States

Origin States

Origin States for Z-association

0. Closed: awaiting an Init request from the application.
1. Init Sent: awaiting an Init-response PDU from the target.
2. Acc Recvd: during initialization the origin has received an Access-control PDU and is awaiting an Access-control response from the application.
3. Rsc Recvd: during initialization the origin has received a resource-control PDU and is awaiting a Resource-control response from the application.
4. Serial Idle: The Z-association is established, there are no active operations, and 'serial operations' is in effect.
5. Concurrent Idle: The Z-association is established, there are no active operations, and 'concurrent operations' is in effect.
6. Serial Active: There is an active operation and serial operations is in effect.
7. Concurrent Active: There is at least one active operation, and concurrent operations is in effect.
8. Z-Acc recvd: The origin has received an Access-control PDU pertaining to the Z-association and is awaiting an Access-control response from the application.
9. Z-Rsc recvd: The origin has received a Resource-control PDU pertaining to the Z-association and is awaiting a Resource-control response from the application.
10. Close sent: awaiting a Close PDU from the target.
11. Close Received: awaiting a Close response from the application.

Origin States for Operation

1. (for Present operation) Present sent: awaiting a Present-response PDU from the target. (for operation other than Present) <Op> sent: awaiting an <Op>-response PDU from the target.
2. Rsc revd: the origin has received a Resource-control request PDU pertaining to the operation and is awaiting a Resource-control response from the application.
3. Acc revd: the origin has received an Access-control request PDU pertaining to the operation and is awaiting an Access-control response from the application.

Target States

Target States for Z-association

0. Closed: awaiting an Init PDU from the origin.
1. Init recvd: awaiting an Init Response from the application.
2. Acc Sent: during initialization the target has sent an Access-control PDU and is awaiting an Access-control-response PDU from the origin.
3. Rsc sent: during initialization the target has sent a Resource-control PDU and is awaiting a Resource-control-response PDU from the origin.
4. Serial Idle: The Z-association is established, there are no active operations, and 'serial operations' is in effect.
5. Concurrent Idle: The Z-association is established, there are no active operations, and 'concurrent operations' is in effect.
6. Serial Active: There is an active operation and serial operations is in effect.
7. Concurrent Active: There is at least one active operation, and concurrent operations is in effect.
8. Z-Acc sent: The target has sent an Access-control PDU pertaining to the Z-association and is awaiting an Access-control-response PDU from the origin.
9. Z-Rsc sent: The target has sent a Resource-control PDU pertaining to the Z-association and is awaiting a Resource-control-response PDU from the origin.
10. Close sent: awaiting a Close PDU from the origin.
11. Close Received: awaiting a Close response from the application.

Target States for Operation

1. (for Present operation) Present sent: awaiting a Present response from the application.
   (for operation other than Present) <Op> sent: awaiting an <Op>-response PDU from the application.
2. Rsc revd: the target has sent a Resource-control PDU pertaining to the operation and is awaiting a Resource-Control-response PDU from the origin
3. Acc revd: the target has sent an Access-control PDU pertaining to the operation and is awaiting an Access-control-response PDU from the origin.

Events and actions

Listed below are the events and actions which appear in the tables. Those corresponding to a service primitive or APDU are listed first (in alphabetical order by the abbreviation used in the tables) followed by miscellaneous actions.

Abbreviation used	**Meaning**
<op> PDU	<operation type> PDU
<op> req	<operation type> request
<op> resp	<operation type> response
<op> resp ind	<operation type> response indication
<op> resp PDU	<operation type> Response PDU
Acc conf	Access-control confirm (operation)
Acc ind	Access-control indication
Acc PDU	Access-control PDU (operation)
Acc req	Access-control request (operation)
Acc resp	Access-control response (operation)
Acc resp PDU	Access-control-response PDU
Acc Resp PDU	Access-control-response PDU (operation)
AnyOpPdu	Any PDU belonging to an operation
AnyPdu	Any PDU
Close conf	Close confirm
Close ind	Close Indication
Close PDU	Close PDU
Close req	Close request
Close resp	Close response
Endop ind	End-operation indication
Init conf+	Init confirm (accept)
Init conf-	Init confirm (reject)
Init ind	Init indication
Init PDU	Init PDU
Init req	Init request
Init resp PDU+	Init-response PDU (accept)
Init resp PDU-	Init-response PDU (reject)
Init resp+	Init response (accept)
Init resp-	Init response (reject)
Prsnt resp ind	Present response indication
Prsnt resp PDU	Present-response PDU
Prsnt resp	Present response
Rsc conf	Resource-control confirm (operation)
Rsc ind	Resource-control indication
Rsc PDU	Resource-control PDU (operation)
Rsc req	Resource-control request (operation)
Rsc resp PDU	Resource-control-response PDU (operation)
Rsc resp PDU	Resource-control-response PDU
Rsc resp	Resource-control response (operation)
Seg PDU	Segment PDU
Seg req	Segment request
Seg ind	Segment Indication

(continued next page)

(events and actions -- continued)

Trigrc ind	Trigger-resource-control indication
Trigrc PDU	Trigger-resource-control PDU
Trigrc req	Trigger-resource-control request
Z-Acc conf	Access-control confirm (Z-association)
Z-Acc PDU	Access-control PDU (Z-association)
Z-Acc req	Access-control request (Z-association)
Z-Acc resp	Access-control response (Z-association)
Z-Acc resp PDU	Access-control-response PDU (Z-association)
Z-rsc req noResp	Resource-control request, "no response" (Z-association)
Z-Rsc conf	Resource-control confirm (Z-association)
Z-Rsc PDU	Resource-control PDU (Z-association)
Z-Rsc req	Resource-control request (Z-association)
Z-Rsc resp PDU	Resource-control-response PDU (Z-association)
Z-Rsc resp	Resource-control response (Z-association)

Miscellaneous actions

Abbreviation used	**Meaning**
Initiate <op> operation	1. Initiate an O-machine for an operation of type <op>. If <op> is Present, table 2 or 5 applies (for origin or target respectively); otherwise table 3 or 6 applies. 2. Origin: send <op> PDU. Target: issue <op> indication. 3. Set initial state for operation to 1. 4. If concurrent operations is in effect, increment opCnt by 1.
KillOps	Immediately terminate any active operations; all further PDUs pertaining to any of those operations are input to the Z-machine.
Set <variable> = <x>	Set the value of the specified variable to x.
(x)	Go to state x.
Decr	Decrement the variable opCnt by 1.
Exit	Terminate the O-machine.

Table 1, part 1 -- State Table for Origin Z39.50 Association: Initialization Phase

State Event	closed 0	Init sent 1	Acc recvd 2	Rsc recvd 3
Init req	Init PDU; (1)			
Init resp PDU+		Init conf+; set opCnt = 0; :[conc] (5) else (4):		
Init resp PDU-		Init conf-; (0)		
Acc PDU		Acc ind; (2)		
Acc resp			Acc resp PDU; (1)	
Rsc PDU		Rsc ind; :[resp] (3) else (1):		
Rsc resp				Rsc resp PDU; (1)

Table 1, part 2 -- State Table for Origin Z39.50 Association: Processing Phase

State Event	Serial Idle 4	Concurr-ent Idle 5	Serial Active 6	Concurrent Active 7	Z-Acc recvd 8	Z-Rsc recvd 9
<op> req	Initiate <op> operation; (6)	Initiate <op> operation; (7)		Initiate <op> operation; (7)	Initiate <op> operation; set RetSt = 7; (8)	Initiate <op> operation; set RetSt = 7; (9)
Endop ind			(4)	Decr; :[noOps] (5) else (7):	Decr; :[noOps] set RetSt = 5:; (8)	Decr; :[noOps] set RetSt = 5:; (9)
Z-Acc PDU		Acc ind; set RetSt = 5; (8)		Acc ind; set RetSt = 7; (8)		
Z-Acc resp					Acc resp PDU; (RetSt)	
Z-Rsc PDU		Rsc ind; :[resp] set RetSt = 5; (9) else (5):		Rsc ind; :[resp] set RetSt = 7; (9) else (7):		
Z-Rsc resp						Rsc Resp PDU; (RetSt)
Close req	Close PDU; (10)	Close PDU; (10)	Close PDU; KillOps; (10)	Close PDU; KillOps; (10)	Close PDU; KillOps; (10)	Close PDU; KillOps; (10)
Close PDU	Close ind; (11)	Close ind; (11)	Close ind; KillOps; (11)	Close ind; KillOps; (11)	Close ind; KillOps; (11)	Close ind; KillOps; (11)

Table 1, part 3 -- State Table for Origin Z39.50 Association: termination Phase		
State **Event**	**Close sent** **10**	**Close Recvd** **11**
AnyOpPdu	(10)	
Z-Rsc PDU	:[noResp] Rsc ind:; (10)	
Z-Acc PDU	(10)	
Close resp		Close PDU; (0)
Close PDU	Close conf; (0)	

Table 2: State Table for Origin Present Operation			
State **Event**	**Present sent** **1**	**Rsc recvd** **2**	**Acc recvd** **3**
Rsc PDU	Rsc ind; :[resp] (2) else (1):		
Rsc resp		Rsc resp PDU; (1)	
Acc PDU	Acc ind; (3)		
Acc resp			Acc resp PDU; (1)
Trigrc req	Trigrc PDU; (1)		
Seg PDU	Seg ind; (1)		
Prsnt resp PDU	Prsnt resp ind; Endop ind; exit		

Table 3: State Table for Origin Operation (other than Present)			
State **Event**	**<op> sent** **1**	**Rsc recvd** **2**	**Acc recvd** **3**
Rsc PDU	Rsc ind; :[resp] (2) else (1):		
Rsc resp		Rsc resp PDU; (1)	
Acc PDU	Acc ind; (3)		
Acc resp			Acc resp PDU; (1)
Trigrc req	Trigrc PDU; (1)		
<op> resp PDU	<op> resp ind; Endop ind; exit		

Table 4, part 1 -- State Table for Target Z39.50 Association: Initialization Phase

State Event	closed 0	Init recvd 1	Acc sent 2	Rsc sent 3
Init PDU	Init ind; (1)			
Init resp+		Init resp PDU+; set opCnt =0; :[conc] 5 else (4):		
Init resp-		Init resp PDU-; (0)		
Acc req		Acc PDU; (2)		
Acc resp PDU			Acc conf; (3)	
Rsc req		Rsc PDU; :[resp] (3) else (1):		
Rsc resp PDU				Rsc conf; (1)

Table 4, part 2 -- State Table for Target Z39.50 Association: Processing Phase

State Event	Serial Idle (4)	Concur-rent Idle 5	Serial Active 6	Concurrent Active 7	Z-Acc sent 8	Z-Rsc sent 9
<op> PDU	Initiate <op> operation; (6)	Initiate <op> operation; (7)		Initiate <op> operation; (7)	Initiate <op> operation; set RetSt = 7; (8)	Initiate <op> operation; set RetSt = 7; (9)
Endop ind			(4)	Decr; :[NoOps] (5) else (7):	Decr; :[Noops] set RetSt = 5:; (8)	Decr; :[Noops] set RetSt = 5:; (9)
Z-Acc req		Acc PDU; set RetSt = 5; (8)		Acc PDU; set RetSt = 7; (8)		
Z-Acc resp PDU					Acc conf; (RetSt)	
Z-Rsc req		Rsc PDU; :[resp] set RetSt = 5; (9) else (5):		Rsc PDU; :[resp] set RetSt = 7; (9) else (7):		
Z-Rsc resp PDU						Rsc resp; (RetSt)
Close req	Close PDU; (10)	Close PDU; (10)	Close PDU; KillOps; (10)	Close PDU; KillOps; (10)	Close PDU; KillOps; (10)	Close PDU; KillOps; (10)
Close PDU	Close ind; (11)	Close ind; (11)	Close ind; KillOps; (11)	Close ind; KillOps; (11)	Close ind; KillOps; (11)	Close ind; KillOps; (11)

Table 4, part 3 -- State Table for Target Z39.50 Association: termination Phase		
State **Event**	**Close sent** **10**	**Close Recvd** **11**
AnyPdu	(10)	
Z-Rsc req noResp		Rsc PDU; (10)
Close resp		Close PDU; (0)
Close PDU	Close conf; (0)	

Table 5 -- State Table for Target Present Operation			
State **Event**	**Present recvd** **1**	**Rsc sent** **2**	**Acc sent** **3**
Rsc req	Rsc PDU; :[resp] (2) else (1):		
Rsc resp PDU		Rsc conf; (1)	
Acc req	Acc PDU; (3)		
Acc resp PDU			Acc conf; (1)
Trigrc PDU	Trigrc ind; (1)		
Seg req	Seg PDU; (1)		
Prsnt resp	Prsnt resp PDU; Endop ind; exit		

Table 6: State Table for Target Operation (other than Present)			
State **Event**	**<op> recvd** **1**	**Rsctrl sent** **2**	**Acctrl sent** **3**
Rsc req	Rsc PDU; :[resp] (2) else (1):		
Rsc resp PDU		Rsc conf; (1)	
Acc req	Acc PDU; (3)		
Acc resp PDU			Acc conf; (1)
Trigrc PDU	Trigrc ind; (1)		
<op> resp	<op> resp PDU; Endop ind; exit		

4.2.4 Protocol Errors

Any events not listed in the tables of section 4.2.3 are not valid and are considered to be protocol errors. With exceptions specified in section 4.3, incorrectly formatted APDUs or APDUs with invalid data are also considered to be protocol errors. This standard does not specify the actions to be taken upon detection of protocol errors. An application context may contain such a specification.

4.3 Rules for Extensibility

All syntactical errors in received APDUs are considered to be protocol errors except for the following case: Unknown data elements, and unknown options within the Options data element, will be ignored on received Init APDUs.

4.4 Conformance

A system claiming to implement the procedures in this standard shall comply with the requirements in sections 4.4.1 and 4.4.2.

4.4.1 Static Requirements

The system shall:

a) act in the role of an origin (by sending Init, Search, and Present APDUs and receiving Init-response, Search-response and Present- response APDUs), or target (by responding properly to Init, Search, and Present APDUs with appropriate Init-response, Search-response and Present-response APDUs), or both; and,

b) support the syntax in section 4.1; and

c) support the Type-1 Query; and

d) support (at minimum) version 2 of the protocol.

4.4.2 Dynamic Requirements

The system shall exhibit external behavior consistent with having implemented:

a) an Information Retrieval ASE which follows all the procedures specified in sections 3, 4.1, 4.2, and 4.3;

b) assignment of values to APDU data elements according to the procedures of sections 3 and 4.1;

c) encoding of APDUs by applying the ASN.1 Basic Encoding Rules (ISO 8825) to the abstract syntax defined in section 4.1; and

d) the Type-1 query whose abstract syntax is defined in section 4.1 and whose structure and rules for evaluation are described in section 3.7.1.