INTERNATIONAL STANDARD

ISO 23950

First edition
1998-07-15

Information and documentation — Information retrieval (Z39.50) — Application service definition and protocol specification

Information et documentation — Recherche d'information (Z39.50) — Définition du service de l'application et spécification du protocole

Reference number
ISO 23950:1998(E)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard ISO 23950 was prepared by ANSI/NISO (as ANSI/NISO Z39.50-1995) and was adopted, under a special “fast-track procedure”, by Technical Committee ISO/TC 46, *Information and documentation*, in parallel with its approval by the ISO member bodies.

Annexes 1 to 12 form an integral part of this International Standard. Annexes 13 to 16 are for information only.

Contents

List of Figures and Tables

Foreword

(Informative)

ISO 23950 is identical in text to ANSI/NISO Z39.50-1995 (except for certain style discrepancies between ISO and ANSI standards, for example, ISO standards have "Annexes" where ANSI standards have "Appendices"), with identical name: *Information Retrieval (Z39.50) Application Service Definition and Protocol Specification*. Note that "Z39.50" is explicitly incorporated into the name of both standards, in order to avoid any possible confusion that these are different standards, and because "Z39.50" is commonly used to refer to the service and protocol defined by this standard. Note that related standards ISO 10162 and ISO 10163 are superseded by the approval of this standard, ISO 23950. Throughout the remainder of this foreword, references to "Z39.50-1995" pertain to ANSI/NISO Z39.50-1995, which is identical to ISO 23950. References to Z39.50-1988, Z39.50-1992, and Z39.50-1994, refer to earlier versions, not identical to ISO 23950.

ANSI/NISO Z39.50-1995, Information Retrieval (Z39.50) Application Service Definition and Protocol Specification, was a revision of ANSI/NISO Z39.50-1992. Because draft versions of Z39.50-1995 were referred to as Z39.50-1994, implementors should take note that any draft referred to as Z39.50-1994 is not the latest version of this standard. Z39.50-1995 is the final, approved version of the standard which was preceded by various drafts referred to as Z39.50-1994.

This protocol was originally proposed in 1984 for use with bibliographic information. Interest in Z39.50 broadened and in 1990 the Z39.50 Implementors Group (ZIG) was established. Members of the ZIG include manufacturers, vendors, consultants, information providers, and universities who wish to access or provide access to various types of information, including bibliographic, text, image, financial, public utility, chemical, and news. ZIG membership is open to all interested parties.

In 1989 the Z39.50 Maintenance Agency was formed and administered at the Library of Congress. It was assigned to revise Z39.50-1988 to achieve bit-compatibility with the International Standard, ISO 10162/10163, Search and Retrieve, SR. At that time, various enhancements to support a wide range of information retrieval activities had been proposed for the 1992 version. However, many of these features were not fully developed, and their incorporation into the 1992 standard would have caused significant delay. Consequently the maintenance agency deferred the proposed new features with a commitment to implementors that development of the required features would proceed, and that the next version would be a compatible superset of the 1992 standard. Z39.50-1992 replaced and superseded Z39.50-1988, and became a compatible superset of SR.

In 1992 the maintenance agency conducted a formal survey of Z39.50 implementors to determine the relative importance of proposed new features. The survey's purposes were (a) to begin to narrow the list to a manageable set, (b) to determine whether the proposed features were adequately specified and understood, and (c) to gauge their perceived cost and complexity. The survey results revealed certain features to be indispensable, and that certain others features could be eliminated from further consideration. For a third set of features, the survey was inconclusive and the disposition of those features eventually was determined by consensus.

Development of Z39.50-1995 began in late 1991. For each meeting of the ZIG, from December 1991 through April 1994, the maintenance agency developed a revised draft. Implementors carefully scrutinized each draft and discussed them at length both over the ZIG Internet mail list, and at the ZIG meeting. Comments and discussion for each draft and the agreements reached at each ZIG meeting were incorporated into a subsequent draft. In April 1994, the ZIG recommended that the draft be put into final form.

The 1992 version came to be known as "version 2," and the 1995 version, "version 3." However, although these version designations do have specific *protocol* significance, they do not refer to versions of the *standard*. Z39.50-1992 specifies protocol version 2; Z39.50-1995 specifies protocol versions 2 and 3.

Although Z39.50-1992 replaced and superseded Z39.50-1988 (which is obsolete) the relationship between Z39.50-1992 and Z39.50-1995 is quite different: Z39.50-1995 is a compatible superset of the 1992 version. An implementor may obtain complete details of version 2 from the Z39.50-1995 document, and build an implementation compatible with Z39.50-1992.

Basics of the Protocol

The protocol specifies formats and procedures governing the exchange of messages between a client and server, thus enabling the client to (a) request that the server search a database and identify records that meet specified criteria, and (b) retrieve some or all of the identified records.

The client may initiate requests on behalf of a user; the protocol addresses communication between the client and server (which may reside on different computers); it does not address interaction between the client and user.

Z39.50-1992 provides the following basic capabilities, all of which are supported in this standard as well. The client may send a search, indicating one or more databases, and including a query as well as parameters which determine whether records identified by the search should be returned as part of the response. The server responds with a count of records identified and possibly some or all of the records. The client may then retrieve selected records. The client assumes that records selected by the search form a "result set" (an ordered set, order determined by the server), and records may be referenced by position within the set. Optional capabilities include:

- The client may specify an *element set* indicating data elements to retrieve in cases where the client does not wish to receive complete database records. For example, the client might specify "If 5 or fewer records are identified, transmit 'full' records; if more than 5 records are found, transmit 'brief' records."
- The client may indicate a *preferred syntax* for response records, for example, USMARC.
- The client may *name* a result set for subsequent reference.
- The client may *delete* a named result set.
- The server may impose *access control* restrictions on the client by demanding authentication before processing a request.
- The server may provide *resource control* by sending an unsolicited or solicited status report; the server may suspend processing and allow the client to indicate whether to continue.

Query Formulation

This standard fully specifies and mandates support of the *type-1* query, expressed by individual search terms, each with a set of attributes, specifying, for example, type of term (subject, name, etc.), whether it is truncated, and its structure. The server is responsible for mapping attributes to the logical design of the database. Terms may be combined in a type-1 query, linked by Boolean operators. Terms and operators are expressed in Reverse Polish Notation.

Attribute Sets

The attributes associated with a search term belong to a particular attribute set, whose definition is *registered*, that is, assigned a unique and globally recognized *attribute-set-id*, an *Object Identifier*, which is included within the query.

Annex 3, ATR, defines and registers the attribute-set *bib-1*, which specifies various attributes useful for bibliographic queries. The bib-1 attribute set was developed by the bibliographic community; it is intended that attribute sets will be developed and registered as needed by other communities. Additional attribute sets may be registered outside of the standard.

Response Records

The protocol distinguishes two types of records that may occur in response messages from the server: database and diagnostic records.

Annex 5, REC, registers object identifiers for various MARC formats, including USMARC, UKMARC, Norway MARC and CANMARC; these object identifiers accompany database records returned by the server. The appendix defines several other types of record formats, and provides for registration of additional record formats.

Diagnostic records are similarly accompanied by an object identifier which identifies their format. Annex 4, ERR, defines and registers two diagnostic record formats (one of which was defined in Z39.50-1992) that include various diagnostic codes useful for bibliographic applications. Additional diagnostic record formats may be registered.

New Features

Provided below is a summary of the enhancements in Z39.50-1995 (versus the 1992 version). The designations "version 2" and "version 3" refer to protocol version; "Z39.50-1992" and "Z39.50-1995" refer to the respective standards. Thus where a particular feature is described as "new in Z39.50-1995," that generally means it applies in either protocol version. An example is Scan: an implementor may add the Scan service to an existing implementation of Z39.50-1992 without incorporating any other new features.

The enhancements described below fall into four categories: search, retrieval, new services and facilities, and miscellaneous enhancements.

Search

Attributes. A number of enhancements pertain to attributes and attribute sets. In version 3, attributes may be combined from different attribute sets within a single query (even for a single search term). This presents two advantages: First, it is useful when searching multiple databases (although version 2 supports multiple-database searches, all attributes within a query must belong to a single attribute set, which inhibits the ability to search multiple databases, unless those databases are similar). Second, new attribute sets may now be defined with less replication.

Version 3 provides two further enhancements allowing flexibility in the definition of attribute sets. First, new data types for attribute values are defined (in version 2 only numeric values are allowed). Second, an attribute set definition may now list alternative sets of evaluation rules (for example, whether the server is allowed to substitute an attribute that it thinks is more appropriate), and the query may select one of the alternatives. The enhanced bib-1 attribute set definition exploits this new feature.

The bib-1 definition in Z39.50-1995 also included many new attributes (as well as all of the attributes in Z39.50-1992).

Extended Result Set Model. The basic model of a result set is developed in Z39.50-1992; the 1995 version describes an "extended result set model," which supports extended proximity searching.

The extended model also supports a new version 3 search function, *restriction*, which is (in effect) an operation on a result set. It permits selection of records from a result set, based on specified attributes.

Search Term. The search term for a query may take on a variety of data types in version 3. (In version 2 a search term is binary and thus essentially has no data type, so the type is often described by a structure attribute.) This enhancement will simplify queries (as well as attribute set definitions) by reducing the need for structure attributes.

Intermediate Results. In Z39.50-1995 the server may provide information per query *component* (i.e., per sub-query, per database), as part of the Search response (version 3 only), or as part of resource-control when the server reports on the progress of the search. The server may also create and provide access to a result set for individual query components.

Retrieval

Segmentation. In version 2, a retrieval response is limited to a single message; the server attempts to fit the requested records into the message, and if it cannot, it simply fits as many as it can. The client might want to retrieve, for example, ten thousand records, knowing it cannot retrieve them in a single message. Typically the client will request all ten thousand records, wait for the response, determine how many records are retrieved, and then send another request for the remaining records. This works well in many environments but is unacceptably slow for high-speed networks. The server must await a request before sending each set of records, which introduces a delay; the delay may be negligible for conventional networks, but is intolerable for high-speed networks. In version 3 a server may respond to a retrieval request with multiple consecutive response messages without intervening requests.

A more serious segmentation problem occurs when a *single* record is too large to fit in a single message. Version 3 thus introduces a second level of segmentation: an individual record may span response messages. A client or server may choose to support either level of segmentation, or no segmentation (in which case version 2 rules apply).

Retrieval Tools. The ZIG worked intensively over two years to develop an extensive model and suite of tools for a wide range of retrieval functions to support various retrieval applications, in particular, document retrieval. The model is detailed in Annex 14, RET. Several new object classes were designated in Z39.50-1995 (schemas, tagSets, variants) and specific objects from these and other classes are defined. Annex RET provides detailed semantics for these objects and describes how they are used together to provide a variety of document retrieval capabilities. Following are a few examples:

- A single database record might include a number of documents. The client may discover and retrieve a specific document, rather than the entire database record.
- The client may retrieve a specific portion of a document, logical or physical, for example, specific pages, a specific chapter, a specific caption, all captions, or all images. The client might retrieve just headings, for example, all chapter or section headings.
- A document might be available in a wide variety of formats (e.g., PostScript, SGML), languages, presentation parameter (e.g., line length, lines per page, columns), and other variants. The client may discover what variants are supported for a document, as well as information associated with a particular variant form: for example the cost to retrieve the document according to a specific variant, or its size. Finally, the client may then retrieve the document (or specific portion) according to the desired variant.
- Associated with a document, for a given search, may be *hits*: pointers to terms (within the document) relevant to the search. The client might retrieve hits along with a document to quickly locate the satisfying portions. Or the client might retrieve only the hits (ranked in order of importance), and subsequently retrieve only the indicated satisfying portions.

New Services and Facilities

Scan and Sort. Scan and Sort were new services in Z39.50-1995. These are used respectively to scan terms in a list or index, and to sort a result set.

Scan is currently the only service in the Z39.50 Browse facility, but it is intended that various other browse capabilities will be added in future versions.

Extended Services. Extended Services was a new facility in Z39.50-1995. It includes a new Z39.50 service, the *Extended Services service*, used to initiate a specific extended service task, which is executed outside of the Z39.50 session and whose progress may be monitored using Z39.50 services. Specific extended services include: save a result set, set a periodic query schedule, export a document, order a document, and update a database.

Explain. The new Explain facility allows a client to retrieve details of the server implementation: general features (description, contact information, hours of operation, restrictions, usage cost, etc.) databases available for searching, indexes, attribute sets, attribute details, schemas, record syntaxes, sort capabilities, and extended services. The server maintains Explain information in a special database that may be accessed by the client using the Z39.50 search and retrieval facilities. The format of the Explain information is detailed in the standard.

Some Explain information is transparent to the client, intended for direct display to the client-user, and is so designated (e.g., “general features”). Some Explain information is intended to be shared by client and user. For example, the client may retrieve a list of searchable databases; for each database in the list the client might display an *informal* name, an icon, and a brief description. Meanwhile the client would retain the *actual* database name to be used in a protocol message, which probably would not be displayed. Some Explain information may be completely transparent to the user. For example, the client may retrieve information about attributes supported for a database and use that information when formulating a query (when converting a user-supplied query to a Z39.50 type-1 query).

Miscellaneous Enhancements

Termination and Re-initialization. Version 3 includes a more flexible approach to termination of a Z39.50 session, to allow, in effect, re-initialization without taking down the network connection.

Concurrent Operations. Multiple concurrent operations are allowed in version 3. In version 2, operations are strictly serial.

Diagnostics. Most Z39.50 services include diagnostic capability. In version 2 a diagnostic must conform to a specific format defined within the standard. In version 3, diagnostic formats may be externally defined and registered. One such (new) format is defined, along with a comprehensive set of diagnostics.

Access Control Formats. Z39.50-1992 provides access control, but does not define any access control formats. Z39.50-1995 defined formats for encryption and authentication, and a format allowing the server to prompt the client for arbitrary information.

Character Set Support. A new data type, “International String,” has been introduced for character strings. Its definition allows greater flexibility for a client and server to agree to the use of a particular language and one or more character sets during a session.

Units. New data types are introduced for support of units. These definitions allow standard representations to be used to represent unit type and unit. For example, unit type might be “mass,” and unit, “kilogram.”

Extensibility and Negotiation. Version 3 provides a powerful extensibility feature. Each protocol message includes a field designated for information whose format is to be defined externally. These externally defined formats will be registered and maintained by the Z39.50 Maintenance Agency as provisional extensions to the standard and for experimental use and possible consolidation into a subsequent version.

In Z39.50-1995 the concept of a “negotiation record” was introduced. The client may include a negotiation record within the initialization message to propose that some condition be in effect for the session (for example, the use of a particular language and one or more character sets). The server may respond, indicating whether the proposal is accepted, or indicate a counter-proposal.

The negotiation record is an application of the new extensibility feature. Negotiation records will be defined externally and maintained by the Z39.50 Maintenance Agency.

Information and documentation — Information retrieval (Z39.50) — Application service definition and protocol specification

1. Introduction

This standard is one of a set of standards produced to facilitate the interconnection of computer systems. It is positioned with respect to other related standards by the Open Systems Interconnection (OSI) basic reference model (ISO 7498). This standard defines a protocol within the application layer of the reference model, and is concerned in particular with the search and retrieval of information in databases.

1.1 Scope and Field of Application

This standard defines the Information Retrieval Application Service (section 3) and specifies the Information Retrieval Application Protocol (section 4). The service definition describes services that support capabilities within an application; the services are in turn supported by the Z39.50 protocol. The description neither specifies nor constrains the implementation within a computer system. The protocol specification includes the definition of the protocol control information, the rules for exchanging this information, and the conformance requirements to be met by implementation of this protocol.

Intended for systems supporting information retrieval services, and for organizations such as information services, universities, libraries, and union catalogue centers, this standard addresses connection-oriented, program-to-program communication. It does not address interchange of information with terminals or via other physical media.

1.2 Version

There have been three publications of Z39.50: Z39.50-1988, Z39.50-1992, and Z39.50-1995; and there has been one publication of the Search and Retrieve Protocol, ISO 10163-1:1993. The three publications: Z39.50-1992, ISO 10163-1:1993, and Z39.50-1995 (but not Z39.50-1988) each incorporate the concept of a protocol version, and three protocol versions are defined: version 1, version 2, and version 3. ISO 10163-1:1993 is based on protocol version 1; Z39.50-1992 is based on protocol version 2; Z39.50-1995 is based on protocol version 2 as well as protocol version 3. (There is no protocol version associated with Z39.50-1988.)

This International Standard, ISO 23950, is based on version 2 and version 3. It assumes that version 1 and version 2 are identical, thus implementations that support version 2 automatically support version 1 (otherwise, version 1 is not explicitly mentioned elsewhere in this standard). Procedures within this standard that apply specifically to version 2 or version 3 are noted as such.

1.3 Referenced Standards

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ANSI/NISO Z39.53-1994, Codes for the Representation of Languages for Information Interchange.
ANSI/NISO Z39.58-1992, Common Command Language for Online Interactive Information Retrieval.
ISO 2709:1996 — Information and documentation - Format for information interchange.
ISO 4217:1990 — Codes for the representation of currencies and funds.
ISO 7498:1984 — Information processing systems - Open systems interconnection - Basic reference model.
ISO 8649:1987 — Information processing systems - Open systems interconnection - Service definition for the association control service element.

ISO 8650:1987 — Information processing systems - Open systems interconnection - Protocol specification for the association control service element.
ISO 8777:1993 — Information and documentation - Commands for interactive text searching.
ISO 8822:1988 — Information processing systems - Open systems interconnection - Connection oriented presentation service definition.
ISO 8824:1990 — Information processing systems - Open systems interconnection - Specification of Abstract Syntax Notation One (ASN.1).
ISO 8825:1990 — Information processing systems - Open systems interconnection - Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1).
ISO 10160:1997 — Information and documentation - Open systems interconnection - Interlibrary loan application service definition.
ISO 10161-1:1997 — Information and documentation - Open system interconnection - Interlibrary loan application protocol specification - Part 1: Protocol specification.
ISO 10163-1:1993 — Information and documentation - Open systems interconnection - Search and retrieve application protocol specification - Part 1: Protocol specification.
Note: Although this ISO standard supercedes ISO 10163-1, there are provisions within this standard intended to maintain compatibility with ISO 10163-1, because of existing implementations of that standard.
ISO — International register of coded character sets.

2. Definitions

For the purposes of this International Standard the following definitions apply.

A-association—See Application association.

Abstract database record—An abstract representation of the information in a database record. An abstract database record may be formed by the application of an abstract record structure (defined by a schema) to the database record. An element specification may be applied to an abstract database record forming another instance of the abstract database record.

Abstract record structure—The primary component of a database schema. An abstract record structure applied to a database record results in an abstract database record.

Abstract syntax—A description of a particular data type using an abstract syntax notation. It can be referenced by an OID (object identifier).

Abstract syntax notation—A language that allows the denotation of data types in a representation-independent manner. ASN.1 is an example.

Access point—A unique or non-unique key that can be specified either singly or in combination with other access points in a search for records. An access point may be equivalent to an element (defined by an abstract syntax), derived from a set of one or more elements, or unrelated to any element.

Access point clause—An operand of a type-1 query (informal).

Aggregate present response—Segment requests (if any) together with the Present response, for a Present operation.

APDU—See **Application Protocol Data Unit**.

Application association—A communication session between a database user and a database provider. It may consist of one or more consecutive Z-associations.

Application Protocol—The rules governing the format and exchange of information between an origin and target.

Application Protocol Control Information —Information conveyed by an application protocol data unit.

Application Protocol Data Unit—A unit of information, transferred between origin and target, whose format is specified by the Z39.50 protocol, consisting of application-protocol-information and possibly application-user-data.

AppliedVariant—One of three usages for a variant specification. The applied variant is the variant specification that the target applies to an element included in a retrieval record. See also **variantRequest** and **supportedVariant**.

ARS—See **Abstract record structure.**

ASN.1—Abstract Syntax Notation One, as specified in ISO 8824 and ISO 8825.

Attribute—A characteristic of a search term, or one of several characteristic components which together form a characteristic of a search term.

Attribute element—An attribute represented by a pair of components: an attribute type and a value of that type.

Attribute list—A set of attribute elements and the attribute set id to which it belongs. An attribute list is combined with a search term to form an operand of a type-1 query. Usually, one of the attribute elements from the set corresponds to a normalized access point, against which the term (as qualified by the other attribute elements) is matched.

Attribute set—A set of attribute types, and for each, a list of attribute values. Each type is represented by an integer, unique within that set (as identified by its attribute set id), and each value for a given type is unique within that type.

Attribute set id—An OID that identifies an attribute set, to which an attribute element (within an attribute list) belongs.

Attribute type—A component of an attribute element. An attribute set defines one or more attribute types and assigns an integer to each type (it also defines values specific to each type). For example, bib-1 assigns the integer 1 for the attribute type "Use."

Attribute value—A component of an attribute element. An attribute set defines one or more values for each attribute type that it defines. For example, bib-1 defines the Use attribute "personal name."

Client—The application that includes the origin; the database user.

Client system—The system on which the client resides.

Composition specification—A specification that may be included in a Present request to indicate the desired composition (elements and record syntax) of the retrieval records. It includes a schema identifier, element specification, and record syntax identifier.

Conditionally confirmed service—A service that may be invoked as confirmed or non-confirmed. It is defined in terms of a request (from the origin or target) followed possibly by a response (from the peer). For example, Resource-control is a conditionally confirmed service, initiated by the target. See also **Non-confirmed service** and **Confirmed service**.

Confirmed service—A service that is defined in terms of a request (from the origin or target) followed by a response (from the peer). For example, Search is a confirmed service, initiated by the origin; Access-control is a confirmed service initiated by the target. See also **Non-confirmed service** and **Conditionally-confirmed service**.

Database—A collection of information units containing related information. Each unit is a database record.

Database record—A local data structure representing an information unit in a database.

Database schema—A common understanding shared by the origin and target of the information contained in the records of the database, which allows the subsequent selection of portions of that information via an element specification. A schema defines an abstract record structure, which, when applied to a database record, results in an abstract database record.

Data element—See **Element.**

Element—A unit of information defined by a schema.

ElementRequest—A request, included with an element specification, for the retrieval of a specific element. The element request may include a variantRequest, indicating the desired variant form of the element.

Element set name—An element specification in the form of a primitive name.

Element specification—An instance of an element specification format, or an element set name. An element specification transforms an abstract database record into another instance of the abstract database record (this may be a null transformation). The element specification selects elements from the abstract database record, and possibly also specifies variant forms for those elements.

Element specification format—A structure used to express an element specification.

Element specification identifier—The object identifier of an element specification format, or an element set name.

Exceptional record size—The maximum size of the record that may be included in a Present response, in the special case when a single, exceptionally large record (i.e., larger than preferred-message-size) is requested.

Facility—A logical group of Z39.50 services; in some cases, a single service. For example, the Retrieval facility consists of the Present service and the Segment service; the Search facility consists of the Search service. Alternatively, a facility might not consist of services, but instead might use services of other facilities. For example, the Explain facility does not define any services, but uses the Search and Present services.

Final fragment—A fragment that ends at the end of a record. See **Fragment.**

Fragment—A proper substring of a record. (This definition is meaningful only in the context of level-2 segmentation, described in section 3.3.3; within that section, a record is considered to be a string of bytes.)
GRS—Generic Record Syntax.

Initiating request—A request that initiates an operation.

Intermediate fragment—A fragment that neither starts at the beginning nor ends at the end of a record. See **Fragment**.

IR—Information Retrieval.

Item—(1) A result set item. (2) A bibliographic item; see ISO 10160.

Maximum segment size—The largest allowable segment of an aggregate Present response (when segmentation is in effect).

Name—A linguistic construct, expressed in some language, that corresponds to an object. A name denotes (i.e., identifies) the object to which it is bound.

Non-confirmed service—A service that is defined in terms of a request from the origin or target, with no corresponding response. For example, Segment is a non-confirmed service initiated by the target. See also **Confirmed service**.

Object identifier—An unambiguous, globally-recognized, registered identifier for a data object, assigned by a registration authority.

OID—See **Object identifier**.

Operation—An initiating request and the corresponding terminating response, along with intervening related messages. For example, a Search operation always includes a Search request and Search response, and may also include access control and resource control messages. Multiple concurrent operations may occur within a Z-association.

Operation type—The name of an initiating request. For example a Search request initiates an operation whose type is “search.”

Origin—The entity that initiates a Z-association and initiates operations during the Z-association.

Origin service-user—That portion of a client that makes requests upon the origin. See **Service-user**.

OSI—Open Systems Interconnection.

P-context—See **Presentation context**.

Preferred message size—The maximum size of a Search response or Present response when no segmentation is in effect. It is expressed in terms of the sum of the sizes (in bytes) of the response records, not including protocol control information.

Presentation context—The pairing of an abstract syntax with a transfer syntax, negotiated by the presentation layer, in order for that abstract syntax to be used during the application association.

Primitive—See **Service primitive**.

Primitive name—A name whose internal structure is not required to be understood or have significance to users of the name.
Note: **primitive name** is not related to **primitive**.

Record syntax—An abstract syntax requested by the origin or used by the target to represent retrieval records. For a complete definition, see section 3.6.3.

Response record—A retrieval record or surrogate diagnostic record representing a database record in a Search response or (aggregate) Present response.

Result set—A local data structure used as a selection mechanism for the transfer of records, identified by a query. Its logical structure is a named, ordered list of result set items, and, possibly, unspecified information that may be used as a surrogate for the search that created the result set.

Result set item—A database name, a pointer to a record within the database, and possibly additional, unspecified information associated with the record.

Result set record—An idiomatic expression referring to the database record represented by a result set item. See **Result set**.

Retrieval record—The exportable structure defined by the application of a record syntax to an abstract database record.

RPN query—A search query represented in Reverse Polish Notation (RPN) format.

Schema—See **Database schema**.

Segment—A message that is sent (or is in preparation for transmission) by the target as part of an aggregate Present response, i.e., a Segment request or Present response.

Server—The application that includes the target; the database provider.

Server system—The system on which the server resides.

Service—(1) A Z39.50 service, as in the "search" service; (2) an extended service, as in the "persistent result set extended service"; (3) the service-provider.

Service primitive—An abstract, implementation-independent representation of an interaction between the service-user and the service-provider. The four types of service primitives are: Request, Indication, Response, and Confirmation.

Service-provider—An abstraction of the totality of those entities (the origin and target) that provide a service to peer service-users. The concept of service-provider is employed to facilitate the specification of protocol procedures. It is used only in section 4.2.2 to describe the protocol model.
Note: the service-provider is not related to the database provider or to the provider of telecommunication services.

Service-user—An origin service-user or a target service-user. That portion of a client or server that makes requests upon the origin or target respectively. The concept of service-user is employed to facilitate the specification of protocol procedures. It is used only in 4.2.2 to describe the protocol model.
Note: The service-user is not related to the database user.

Simple Present response—An aggregate Present response consisting of a single segment, i.e., consisting of a Present response only, and no Segment requests.

Starting fragment—A fragment that starts at the beginning of a record. See **Fragment**.

SupportedVariant—One of three usages for a variant specification. A supportedVariant is a variant specification that the target lists as supported for a particular element. See also **appliedVariant** and **variantRequest**.

Surrogate diagnostic record—A diagnostic record supplied in place of a retrieval record, representing a database record.

Tag—The identifier of an element (or of a node of the tagPath representing an element). It consists of a tagType and a tagValue.

TagPath—A sequence of nodes from the root of a tree to the node that the tagPath represents (when the elements of a record are represented hierarchically, as a tree). Each node of a tagPath is represented by a tag. The end-node might be a leaf-node, in which case the tagPath represents an element; otherwise the tagPath represents the subtree whose root is that node.

TagSet—The tagValues (and recommended data types) for a set of elements.

TagSetId—an object identifier serving as a persistent identifier for a tagSet.

TagType—A short-hand (integer) identifier for a tagSet. A schema definition may assign a tagType to a TagSetId, to identify a particular tagSet (within the context of the schema definition).

TagValue—The identifier of an element (or of a node of the tagPath representing an element). It may be either integer or string, and it is qualified by a tagType.

Target—The entity that accepts a Z-association.

Target service-user—That portion of a server that makes requests upon the target. See **Service-user**.

Terminating response—A response that ends an operation.

Transfer syntax—A syntax that when paired with an abstract syntax forms a record syntax.

Triple—A 3-tuple (i.e., an n-tuple, where n = 3).

Type-1 query—See **RPN Query**.

Variant—One of possibly several forms in which an element is available for retrieval. The origin may request, or the target present, an element according to a specific variant. The target may indicate what variants are available for an element.

Variant list—A list provided by the target of the supportedVariants for a particular element.

VariantRequest—One of three usages for a variant specification. A variantRequest is a variant specification occurring within an element request. See also **appliedVariant** and **supportedVariant**.

Variant set—A definition of a set of classes; for each class, a set of types; and for each type, a set of values. A variant specification consists of a set of variantSpecifiers from a particular variant set.

Variant set identifier—An OID identifying a variant set.

Variant Specification—A variantRequest, appliedVariant, or supportedVariant. A variant specification is a sequence of triples, each of which is a variantSpecifier.

Variant Specifier—A component of a variant specification. It consists of a class, a type defined for that class, and a value defined for that type.

Z-association—See **Z39.50-association**.

Z39.50-association—A session, explicitly established by the origin and either explicitly terminated by the origin or target, or implicitly terminated by termination of the A-association. Communication between origin and target is via a Z39.50-association within an application association. There may be multiple, consecutive Z-associations within an A-association.

3. Information Retrieval Service

The Information Retrieval service definition describes an activity between two applications: an initiating application, the client, and a responding application, the server. The server is associated with one or more databases.

Communication between the client and server is carried out by the Z39.50 protocol, which is specified in section 4. The specification is logically divided into procedures pertaining to the client and procedures pertaining to the server. The portions of the client and server that carry out the Z39.50 protocol procedures are referred to respectively as the Z39.50 origin and the Z39.50 target.

3.1 Model and Characteristics of the Information Retrieval Service

Communication between origin and target is via a Z39.50-Association (Z-association) within an application association (A-association; see 4.2.1.2). A Z-association is explicitly established by the origin and may be explicitly terminated by either origin or target, or implicitly terminated by termination of the A-association. There may be multiple consecutive Z-associations within an A-association. There may be multiple consecutive as well as concurrent operations (see 3.5) within a Z-association.

The roles of origin and target may not be reversed within a Z-association. A Z-association cannot be restarted, thus once a Z-association is terminated no status information is retained except information that is explicitly saved.

The service definition describes services and operations; models for these are described in 3.1.1 and 3.1.2. Services are grouped by facilities; the Z39.50 facilities and services are defined in 3.2.

3.1.1 Z39.50 Services

Z39.50 services are carried out by the exchange of messages between the origin and target. A message is a *request* or a *response*. Services are defined to be *confirmed, non-confirmed,* or *conditionally-confirmed.*

A confirmed service is defined in terms of a request (from the origin or target) followed by a response (from the peer). For example, Search is a confirmed service initiated by the origin; the Search service is defined in terms of a Search request from the origin followed by a Search response from the target. Access-control is an example of a confirmed service initiated by the target.

A non-confirmed service is defined in terms of a request from the origin or target, with no corresponding response. For example, TriggerResourceControl is a non-confirmed service initiated by the origin; Segment is a non-confirmed service initiated by the target.

A conditionally-confirmed service is a service that may be invoked as either a confirmed or non-confirmed service. It is defined in terms of a request (from the origin or target) followed possibly by a response (from the peer). For example, Resource-control is a conditionally-confirmed service initiated by the target.

3.1.2 Z39.50 Operations

This standard describes eight *operation types*: Init, Search, Present, Delete, Scan, Sort, Resource-report, and Extended-services.

A request from the origin of a particular operation type initiates an operation of that type (for example a Search request initiates a Search operation) which is terminated by the respective response from the target. Only the origin may initiate an operation, and not all origin requests do so (see 3.4).

A request that initiates an operation is called an *initiating request* and a response that ends an operation is called a *terminating response*. From the origin perspective, an operation begins when it issues the initiating request, and ends when it receives the terminating response. From the target perspective, the operation begins when it receives the initiating request and ends when it sends the terminating response. An operation consists of the initiating request and the terminating response, along with any intervening related messages (see 3.4).

3.1.3 Model of a Database

The objective of this standard is to facilitate the open interconnection of clients and servers for applications where clients search and retrieve information from server databases. The ways in which databases are implemented differ considerably; different systems have different styles for describing the storage of data and the means by which it can be accessed. A common abstract model is therefore used in describing databases, to which an individual system can map its implementation. This enables different systems to communicate in standard and mutually understandable terms for the purpose of searching and retrieving information from a database. The search and retrieval models are described in 3.1.4 and 3.1.5.

The term *database*, as used in this standard, refers to a collection of records. Each record is a collection of related information, treated as a unit. The term *database record* refers to a local data structure representing the information in a particular record. Associated with a database are one or more sets of *access points* that can be specified in a search for database records (see 3.1.4), and one or more sets of *elements* that may be retrieved from a database record (see 3.1.5). An access point is a unique or non-unique key that can be specified either singly or in combination with other access points in a search for records. An access point may, but need not, be related to an element; it can be equivalent to an element, derived from a set of one or more elements, or unrelated to any element.

3.1.4 Searching a Database

A query is applied to a database, specifying values to be matched against the access points of the database. The subset of records formed by applying a query is called the *result set* (see 3.1.6). A result set may itself be referenced in a subsequent query and manipulated to form a new result set.

A search request specifies one or more databases and includes a query. The type-1 query defined in this standard (see 3.7) consists of either a single access point clause, or several access point clauses linked by logical operators. For example, In the database named "Books" find all records for which the access point 'title word' contains the value "evangeline" AND the access point 'author' contains the value "longfellow."

Each access point clause consists of a search term and attributes. The attributes qualify the term; usually, one of the attributes corresponds to a normalized access point against which the term (as qualified by the other attributes) is matched. Each attribute is a pair representing an attribute type and a value of that type (for example, type might be "access point" and value "author"; or type might be "truncation" and value "left").

Each attribute is qualified by an attribute set id which identifies the attribute set to which the attribute belongs. An attribute set specifies a set of attribute types, and for each, a list of attribute values.

3.1.5 Retrieving Records from a Database

Following the processing of a search, the result set is made available by the target, to the origin, for subsequent retrieval requests. When requesting the retrieval of a record from a result set, the origin may supply a *database schema* identifier, *element specification*, and *record syntax* identifier.

For the purpose of retrieving records from a result set, associated with each database are one or more schemas. A schema represents a common understanding shared by the origin and target of the information contained in the records of the database, to allow the subsequent selection of portions of that information via an element specification.

A schema defines an *abstract record structure* which, when applied to a database record results in an *abstract database record*, which is an abstract representation of the information in the record. An element specification applied to an abstract database record results in another instance of the abstract database record (the latter may be a null transformation). The element specification selects elements from the abstract database record, and may also specify variant forms for those elements.

The target applies a record syntax to an abstract database record, resulting in an exportable structure referred to as a *retrieval record*.

3.1.6 Model of a Result Set

In general, it is assumed that query processing does not necessarily require physical access to records; a result set is thus assumed to be the identification of (e.g., pointers to) records, as opposed to the actual set of records, selected by a query. (It is not assumed that the database records are locked. Methods of concurrency control, which would prevent modification or deletion of result set records, are not addressed by this standard.) A result set may be used as a selection mechanism for the transfer of records between systems; the result set itself is considered to be a purely local data structure and is not transferred (that is, records are transferred, but not the local pointers to the records).

For the purpose of retrieving records, the logical structure of a result set is that of a named, ordered list of items. Each item is a triple consisting of: (a) an ordinal number corresponding to the position of the triple in the list, (b) a database name, and (c) a unique identifier (of local significance only) of a record within the database named in (b).

A result set item is referenced by its position within the result set, that is, by (a).

For the purpose of searching, when a result set is used as an operand in a query, the logical structure is one of the following:

- Basic model: A set of pairs, each consisting of (b) and (c) of the above model for retrieval.
- Extended model: A set of triples, each consisting of (b) and (c) of the retrieval model; and including unspecified information associated with each record which may be used as a surrogate for the search that created the result set.

Note: Query specifications may indicate that the basic model applies, or under what condition the extended model applies, and the nature of the unspecified information. For the type-1 query, when version 2 is in effect, the basic model applies.

3.1.7 Model of Extended Services

The family of Z39.50 services includes the Extended Service (ES) service. "Extended services" refers to a class of services recognized by this standard, but which are not Z39.50 services (as described in 3.1.1). The ES service *is* a Z39.50 service, and an ES operation results in the initiation of an extended services *task*. The task is *not* considered part of the Z39.50 ES operation.

An ES operation is initiated by the origin, via an ES request. The ES response, which completes the operation, does not (necessarily) signal completion of the task; it may indicate for example that the task has started or is queued (or it might indicate that the task has been completed; in fact the ES request may specify that the task should be completed prior to the ES response). An ES task may have a lifetime beyond the Z-association.

Examples of extended services are: saving a result set or query, and exporting or ordering a document.

Each ES task is represented by a database record, called a *task package*, maintained by the target in a special database, the “extended services database.” The origin uses the ES request to cause creation of a task package on the ES database. The database may be searched, and records retrieved, by the Z39.50 Search and Retrieval facilities. The origin may search for packages of a particular type, or created by a particular user, or of a particular status (i.e., pending, active, or complete), or according to various other criteria. In particular, the origin may search the database after submitting an ES request (during the same or a subsequent Z-association) for a resulting task package to determine status information pertaining to the task, for example, to determine whether the task has started.

3.1.8 Explain

The origin may obtain details of the target implementation, including databases, attribute sets, diagnostic sets, record syntaxes, and element specifications supported. The origin obtains these details through the Z39.50 Explain facility. The target maintains this information in a database that the origin may access via the Z39.50 Search and Present facilities.

This “explain” database appears to the origin as any other database supported by the target, but it has a well-known name and a predefined record syntax. Also, certain search terms, corresponding to information categories, are predefined in order to allow a semantic level of interoperability. Each information category has its own record layout, and all are included in the Explain syntax.

3.2 Facilities of the Information Retrieval Service

Sections 3.2.1 through 3.2.11 describe the eleven facilities of the Information Retrieval service. Most consist of logical groups of services; in several cases, a facility consists of a single service. Additional services may be added to any facility in future versions of this standard. Following is a summary description of the eleven facilities.

Initialization Facility—Init Service: A confirmed service initiated by the origin to initiate an Init operation.

Search Facility—Search Service: A confirmed service initiated by the origin to initiate a Search operation.

Retrieval Facility—The Retrieval facility consists of two services:

1. Present Service: A confirmed service initiated by the origin to initiate a Present operation.
2. Segment Service: A non-confirmed service initiated by the target, during a Present operation.

   Note: a Present operation thus consists of a Present request followed by zero or more Segment requests followed by a Present response.

Result-set-delete Facility—Delete Service: A confirmed service initiated by the origin to initiate a Delete operation.

Browse Facility—Scan Service: A confirmed service initiated by the origin to initiate a Scan operation.

Sort Facility—Sort Service: A confirmed service initiated by the origin to initiate a Sort operation.

Access Control Facility—Access-control service: a confirmed service initiated by the target. It does not initiate an operation, and it might or might not be part of an active operation.

Accounting/Resource Control Facility—The Accounting/ Resource Control facility consists of three services:

1. Resource-control Service: A conditionally-confirmed service initiated by the target. It does not initiate an operation, and it might or might not be part of an active operation.
2. Trigger-resource-control Service: A non-confirmed service initiated by the origin during an operation.
3. Resource-report Service: A confirmed service initiated by the origin to initiate a Resource-report operation.

Explain Facility—The Explain facility does not include any services, but uses the services of the Search and Retrieval facilities.

Extended Services Facility—Extended-services Service: A confirmed service initiated by the origin to initiate an Extended-services operation.

Termination Facility—Close Service: A confirmed service initiated by the origin or target. It does not initiate nor is it part of any operation. It allows an origin or target to abruptly terminate all active operations and to initiate termination of the Z-Association. (Following termination of the Z-Association the origin may subsequently attempt to initialize another Z-Association using the Init service.)

3.2.1 Initialization Facility

The Initialization facility consists of the single service, Init.

3.2.1.1 Init Service

The Init service allows the origin to establish a Z-association. In the Init request, the origin proposes values for initialization parameters. In the Init response, the target responds with values for the initialization parameters; those values, which may differ from the origin-proposed values, are in effect for the Z-association. See Table 1.

If the target responds affirmatively (Result = ‘accept’), the Z-association is established. If the origin then does not wish to accept the values in the target response, it may terminate the Z-association, via the Close service (and may subsequently attempt to initialize again). If the target responds negatively, the origin may attempt to initialize again.

Table 1: Parameters of the Init Service

Parameter	Origin Request	Target Response
Version	x	x
Id/authentication	x (opt)	
Options	x	x
Preferred-message-size	x	x
Exceptional-record-size	x	x
Result		x
Implementation-id	x (opt)	x (opt)
Implementation-name	x (opt)	x (opt)
Implementation-version	x (opt)	x (opt)
User-information-field	x (opt)	x (opt)
Other-information	x (opt)	x (opt)
Reference-id	x (opt)	x (if appl)

3.2.1.1.1 Version. Both the origin and target indicate all versions that they support. The highest common version is selected for use, and is said to be ‘in force’, for the Z-association. If there are no versions in common, the target should indicate ‘reject’ for the parameter Result.
Notes:

1. Version numbers higher than the highest known version should be ignored.
2. 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 10163-1:1993 implementations).

3.2.1.1.2 Id/authentication. The origin and target agree, outside the scope of the standard, whether or not this parameter is to be supplied by the origin, and if so, to the value. This value is used by the target to determine whether the origin is authorized to enter into communication with the target.

3.2.1.1.3 Options. For each of the capabilities listed below, the origin proposes either ‘on’ or ‘off’ (meaning ‘in effect’ or ‘not in effect’ respectively) and the target responds correspondingly for each. The response determines whether the capability is in effect. The capabilities are:

- search
- present
- delete
- resource-report
- scan
- sort
- extended-services
- trigger-resource-control
- level 1 segmentation
- level 2 segmentation
- concurrent operations
- named result sets
- resource-control
- access-control.

Note: the above list of capabilities is subject to expansion in future versions of this protocol.

The following rules, describing how these capabilities are to be negotiated, are intended to allow interoperation even when the origin and target have not necessarily implemented the same capabilities.

The Options parameter consists of a string of Boolean flags, each corresponding to an individual capability. The origin might set the flag to ‘in effect’ for a capability unknown to the target. In that case it is recommended that the target set the corresponding flag to ‘not in effect’ in the response. However, if the origin sets a flag to ‘not in effect’ and the target sets the corresponding flag to ‘in effect’, and if the origin is not aware what capability that flag represents, it is recommended that the origin terminate the Z-association.

Search, present, delete, resource-report, scan, sort, and extended-services: for each of these operation types, the origin indicates whether it wishes to initiate operations of that type; if so, the target indicates whether it is willing to process an operation of that type. If the origin proposes ‘not in effect’ for a particular operation type, the target must also specify ‘not in effect’.
Notes:

1. The target indication that it is willing to process a Resource-report operation does not imply that it will include a resource report in the response.
2. Any of the above operation types may be negotiated for any version. In particular, Scan, Sort, and Extended-services may be negotiated when version 2 is in force, even though they are not defined in ANSI Z39.50-1992.

Trigger-resource-control: The origin may propose to submit Trigger-resource-control requests; if so, the target indicates whether it will accept Trigger-resource-control requests. If the origin proposes ‘not in effect’, the target must also specify ‘not in effect’.
Notes:

1. If the target specifies ‘in effect’ for Trigger-resource-control, but ‘not in effect’ for ‘resource-control’, then the origin may use only the Cancel function of Trigger-resource-control.
2. The target may indicate unwillingness to accept Trigger-resource-control requests even if it specifies ‘in effect’ for ‘resource-control’.
3. The target’s indication of willingness to accept Trigger-resource-control requests does not imply that the target will take any action as a result of a Trigger-resource-control request.

Level 1 and level 2 segmentation: The origin proposes one of the following:

- “no segmentation,” by specifying ‘not in effect’ for both level 1 and level 2 segmentation;
- “level 1 segmentation,” by specifying ‘in effect’ for level 1 and ‘not in effect’ for level 2 segmentation; or
- “level 2 segmentation,” by specifying ‘in effect’ for level 2 segmentation.

Notes:

1. If the origin proposes ‘in effect’ for level 2 segmentation then it may also propose ‘in effect’ for level 1 segmentation to indicate that if the target is unable to support level 2 segmentation, the origin wishes level 1 segmentation to be in effect.
2. “segmentation” is said to be ‘in effect’ if either level 1 or level 2 segmentation is in effect.
3. Segmentation may be in effect only when version 3 is in force.

The target response indicates which form of segmentation it intends to perform.

- If the target specifies neither level 1 nor level 2 then ‘no segmentation’ is in effect, regardless of what the origin has proposed.
- If the target specifies level 1 (but not level 2) segmentation, it will not perform level 2 segmentation, and the origin must be prepared to accept level 1 segmentation, regardless of what the origin has proposed.
- If the target specifies level 2 segmentation, the origin must be prepared to accept level 2 segmentation regardless of what it has proposed (the target value for level 1 should be ‘not in effect’).

When ‘no segmentation’ is in effect, the target response to a Present request must consist of a single message (a single “segment,” i.e., a Present response only, with no intervening Segment requests), containing an integral number of records. When ‘Level 1 segmentation’ is in effect the target may respond to a Present request with multiple segments (i.e., a Present response, with possibly one or more intervening Segment requests); each must contain an integral number of records. When ‘Level 2 segmentation’ is in effect the target may respond to a Present request with multiple segments, and individual records may span segments. Segmentation procedures are detailed in 3.3.

Concurrent-operations: The origin may propose to initiate concurrent operations; if so, the target indicates whether it will accept concurrent operations. If the origin proposes ‘not in effect’, the target must also specify ‘not in effect’. If concurrent operations is not in effect, then ‘serial operations’ is said to be in effect. Concurrent operations may be in effect only when version 3 is in force.

Named-result-sets: The origin may propose to use named-result sets (i.e., to specify result set names other than “default” as the value of Result-set-name within a Search request); if so, the target specifies whether it will support named-result-sets. If the origin proposes ‘not in effect’, the target must also specify ‘not in effect’.

Resource-control and access-control: The origin indicates whether it wishes the target to invoke Resource-control and/or Access-control (i.e., send Resource-control and/or Access-control requests). The target specifies whether it plans to invoke Resource-control and/or Access-control.
Notes:

1. If the target specifies ‘not in effect’ for resource-control (or access-control) then it will not invoke resource-control (or access control) even if the origin has proposed ‘in effect’.
2. If the origin proposes ‘not in effect’ for resource-control, and the target indicates ‘in effect’ for resource-control, indicating that it is not willing to suppress Resource-control requests, and if indeed the origin cannot accept Resource-control requests, the origin should terminate the Z-association.

3. If the origin proposes 'not in effect' for access-control, and if security requirements on the target system mandate that security (other than that which might be provided by the parameter Id/authentication) be invoked at the outset of a Z-association, then the target should reject the Z-association (by setting the parameter Result to 'reject', and specifying 'in effect' for 'access-control').

However, security may be invoked at different levels. In addition to authentication at the outset of a Z-association, security might be invoked to control access to a particular database, record, result-set, resource-report format, or use of an operation. Thus if the origin proposes 'not in effect' for access-control, and the target normally invokes security (other than at the Z-association level), the target need not necessarily reject the Z-association.

The target might wish to invoke a security challenge during an Init operation to determine whether the origin is authorized to use a capability it has proposed. If the origin has proposed 'not in effect' for access-control, the target may simply refuse the use of that particular operation via the Options parameter.

If the origin proposes 'not in effect' for access-control, and the target chooses to accept the Z-association, and if the origin subsequently initiates an action that would precipitate an Access-control request (for example, if the origin issues a Search specifying a database for which it has not yet established credentials), the target should suppress the Access-control request and instead respond with an error status indicating that a security challenge is required but cannot be issued.

3.2.1.1.4 Preferred-message-size and Exceptional-record-size. The Init request contains the origin's proposed values of Preferred-message-size and Exceptional-record-size, specified in bytes. The Init response contains the Preferred-message-size and Exceptional-record-size that the target is going to use; these may be different from (and override) the values proposed by the origin. For both the request and response, Preferred-message-size must be less than or equal to Exceptional-record-size.

Exceptional-record-size is meaningful during a Present operation, and only in the special case when a single, exceptionally large record (i.e., larger than preferred-message-size) is requested in the Present request. In this special case, preferred-message-size may be overridden (for the present operation), so that a single record may be presented whose size may be as large as Exceptional-record-size. The fact that a single record is requested is how the origin signals that preferred-message-size may be overridden. Thus Exceptional-record-size must be greater than or equal to preferred-message-size. In the case where they are equal, Exceptional-record-size has no meaning (this is the way to signify that the special case will not apply during the Z-association).

The usage of these parameters is detailed in 3.3.

Note: The parameter Exceptional-record-size has the same meaning as the parameter Maximum-record-size defined in Z39.50-1992. The name of the parameter has been changed for clarity.

3.2.1.1.5 Result. The target indicates whether or not it accepts the Z-association by specifying a value of 'accept' or 'reject' in the parameter Result. (If 'reject' is indicated, the origin may send another Init request.)

3.2.1.1.6 Implementation-id, Implementation-name, and Implementation-version. The request or response may optionally include any of these three parameters. They are, respectively, an identifier (unique within the client or server system), descriptive name, and descriptive version, for the origin or target implementation. These three implementation parameters are provided solely for the convenience of implementors, for the purpose of distinguishing implementations.

3.2.1.1.7 User-information-field. This parameter may be used by the origin or target for additional information not specified by this standard.

3.2.1.1.8 Other-information. This parameter may be used by the origin or target for additional information not specified by the standard. This parameter may be used only if version 3 is in force.

Note: Care should be taken by the origin when using this parameter; the origin cannot ascertain that version 3 is in force before sending the Init request.

3.2.1.1.9 Reference-id. See 3.4.

3.2.2 Search Facility

The Search facility consists of the single service, Search.

3.2.2.1 Search Service

The Search service enables an origin to query databases at a target system, and to receive information about the results of the query.

The search request allows the origin to request that the target apply a query to a specified set of databases at the target, to identify records with the properties indicated by the query. The target creates a *result set*, which represents the set of records identified by the query, and the target maintains the result set for subsequent retrieval requests.

Depending on the parameters of the search, one or more records identified by the result set may be immediately retrieved as part of the search response. The result set is an ordered set; a record identified by an entry in the result set is referenced by the position of the entry within the result set (beginning with 1). See Table 2.

Table 2: Parameters of the Search Service

Parameter	Origin Request	Target Response
Query-type	x	
Query	x	
Database-names	x	
Result-set-name	x	
Replace-indicator	x	
Small-set-element-set-names	x (opt.)	
Medium-set-element-set-names	x (opt.)	
Preferred-record-syntax	x (opt.)	
Small-set-upper-bound	x	
Large-set-lower-bound	x	
Medium-set-present-number	x	
Response-records		x (if appl.)
Result-count		x
Number-of-records-returned		x
Next-result-set-position		x
Search-status		x
Result-set-status		x (if appl.)
Present-status		x (if appl.)
Additional-search-information	x (opt.)	x (opt.)
Other-information	x (opt.)	x (opt.)
Reference-id	x (opt.)	x (if appl.)

3.2.2.1.1 Query-type and Query. The parameter Query-type identifies the type of query, i.e the syntax of parameter Query. Six types are defined:

- *Type-0* may be used only when the origin and target have a priori agreement outside of the standard.
- *Type-1* is the Reverse Polish Notation (RPN) query specified in 3.7.
- *Type-2* is the ISO8777 type query, specified in ISO 8777.
- *Type-100* is the Z39.58 type query, specified in ANSI Z39.58.
- *Type-101* is the extended RPN (ERPN) query, an extension to the type-1 query to allow proximity searching and restriction of result sets by attributes. It is specified in 3.7.

  Note: The type-101 query is identical to the type-1 query with the following exception: For type-1, proximity and restriction are valid only when version 3 is in force. For type-101, proximity and restriction are valid both for version 3 and version 2 as well. (The definition of the type-101 query is independent of version.)
- *Type-102* is the Ranked List query, to be defined in a later version of this standard.

3.2.2.1.2 Database-names. The origin indicates the set of databases to which the Query applies.
Notes:

1. The target designates (through the Explain facility or through some mechanism outside of the standard) what databases may be specified on a Search request, and in what combinations they may be specified. For example, a target might specify that databases **A**, **B**, and **C** may be searched individually, and that **A** and **B** may be searched in combination (but not **A** and **C**, nor **B** and **C**).

2. Each database name specified by the target is a string of characters, and the string is case-insensitive. That is, for any character that is a letter, the origin may use either upper- or lower-case, regardless of how the target has specified the name.

3.2.2.1.3 Result-set-name and Replace-indicator. The parameter Result-set-name specifies a name to be given to the result set (to be created by the query) so that it may be subsequently referenced (within the same Z-association).

If a result set with the same name already exists at the target, the action taken depends on the value of the parameter Replace-indicator, as follows:

- If the value of Replace-indicator is 'on', then after processing the query, the existing result set whose name is specified by the parameter Result-set-name will be deleted, and a new result set by that name created. If the search cannot be processed, the content of the result set will be empty.
- If the value of Replace-indicator is 'off', the search is not processed, an error diagnostic is returned by the target, and the existing result set whose name is specified by the parameter Result-set-name is left unchanged.

If a result set does not exist with the name specified by the parameter Result-set-name, then a result set by that name is created by the target, and the parameter Replace-indicator is ignored. The initial content of the result set is empty. If no records are found by the query, the result set remains empty.

A target need not support, in general, the naming of result sets by the origin. However, a target must support at least the result set whose name is "default." If the origin specifies "default" as Result-set-name, then Replace-indicator must be 'on'.

A result set created by a Search request (that is, specified by the parameter Result-set-name) may be referenced in a subsequent Present request or as an operand in a subsequent Search request (for example, in a type-1 query). If a result set named "default" is created, it remains available for reference from the time it is created until the end of the Z-association during which it is created, or until either:

- another default result set is created, because the name "default" is specified as Result-set-name in a subsequent Search request, or
- it is unilaterally erased or deleted by the target.

Any result set other than the result set named "default" remains available for reference from the time it is created until it is deleted in one of the following ways:

- by a Delete operation
- implicitly, because a result set was specified by the same name in a Search request, and the value of the parameter Replace-indicator was 'on'
- unilaterally by the target (at any time)
- by termination of the Z-association.

3.2.2.1.4 Small-set-element-set-names and Medium-set-element-set-names. These parameters describe the preferred composition of the records expected in the search response. If the query results in a small-set (see 3.2.2.1.6), then Small-set-element-set-names pertains. If the query results in a medium-set, then Medium-set-element-set-names pertains. These two parameters are described in 3.6.2.

3.2.2.1.5 Preferred-record-syntax. The origin may specify a preferred record syntax for retrieval records. If the target cannot supply a particular record according to the Preferred-record-syntax, it supplies the record according to one of the other abstract syntaxes from the set for which Presentation contexts are currently established for this A-association.

If the target cannot supply the record according to either the requested syntax or a syntax corresponding to an established presentation context, it returns a surrogate diagnostic for that record unless the established set of presentation contexts is empty; in that case, this standard does not prescribe the target action.

3.2.2.1.6 Small-set-upper-bound, Large-set-lower-bound, and Medium-set-present-number. The result set is considered a "small-set," "medium-set," or "large-set," depending on the values of parameters Small-set-upper-bound and Large-set-lower-bound of the Search request, and Result-count of the Search response (see 3.2.2.1.8). The result set is a small-set if Result-count is not greater than small-set-upper-bound. The result set is a large-set if Result-count is larger than or equal to Large-set-lower-bound. Otherwise, the result set is a medium-set. If the query results in a small-set, response records corresponding to all database records identified by the result set are to be returned to the origin (subject to possible message size constraints). If the query results in a large-set, no response records are to be returned. If the query results in a medium-set, the maximum number of response records to be returned is specified by Medium-set-present-number.

Notes:

1. The result set may be a medium-set only when Result-count is greater than small-set-upper-bound and less than Large-set-lower-bound, and this can occur only if Large-set-lower-bound is at least 2 greater than Small-set-upper-bound; i.e., the result set cannot be a medium-set if Large-set-lower-bound exceeds Small-set-upper-bound by 1. For example, if Large-set-lower-bound is 11 and Small-set-upper-bound is 10, the intent is "if 10 or fewer database records are found, return response records for them all, otherwise do not return any," and medium-set-present-number would not apply.
2. Small-set-upper-bound may be zero. Large-set-lower-bound must be greater than Small-set-upper-bound.
3. If the origin does not want any response records returned regardless of the value of Result-count, Large-set-lower-bound should be set to 1 and Small-set-upper-bound to zero.

3.2.2.1.7 Response-records. The target processes the search, creating a result set that identifies a set of database records. It cannot be assumed however that search processing requires physical access to the database records. A particular database record might not be accessible but this circumstance might not be recognized until an attempt is made to access the record for the purpose of forming a retrieval record.

After processing the search, the target attempts to create retrieval records to be included in the Search response, correspondi ng to the first N database records identified by the result set (N depends on the request parameters and Result-count, as described in 3.2.2.1.6). For each database record for which a retrieval record cannot be included, a surrogate diagnostic record is substituted.

The term *response record* refers to a retrieval record or a surrogate diagnostic record. The parameter Response-records is one of the following:

•N response records;

•a number of response records, which is less than N because of message size constraints (see 3.3);

•one or more non-surrogate diagnostic records (see note) indicating that the search cannot be processed, and why it cannot be processed; or

•one or more non-surrogate diagnostic records (see note) indicating that records cannot be presented, and why not, e.g., "element set name not valid for database."

Note: If version 2 is in force, the target returns a single non-surrogate diagnostic record. If version 3 is in force, the target returns one or more non-surrogate diagnostic records.

The order of occurrence of response records within the parameter Response-records is according to the order in which they are identified by the result set. Each may optionally be accompanied by the name of the database in which the record resides. However, the database name must accompany the first response record being returned, and must accompany any record from a database different from its immediate predecessor.

3.2.2.1.8 Result-count and Number-of-records-returned. The parameter Result-count is the number of database records identified by the result set. If the result set is empty, result-count is zero. The parameter Number-of-records-returned is the total number of records returned in the Search response.

3.2.2.1.9 Next-result-set-position. The parameter Next-result-set-position takes on the value M+1, where M is the position of the result set item which identifies the database record corresponding to the last response record among those returned; or zero if M = Result-count.

3.2.2.1.10 Search-status. The parameter Search-status, returned in the response, assumes one of the following two values:

success	-	The search completed successfully.
failure	-	The search did not complete successfully.

A value of 'success' does not imply that the expected response records are being returned as part of the response (see Present-status, 3.2.2.1.11). Note also, a value of 'success' does not imply that any database records were located by the search. A value of 'failure' *does* imply that *none* of the expected response records is being returned. In the latter case, the target returns one or more non-surrogate diagnostic records (see note) indicating that the search cannot be processed.

Note: If version 2 is in force, the target returns a single non-surrogate diagnostic record. If version 3 is in force, the target returns one or more non-surrogate diagnostic records.

3.2.2.1.11 Result-set-status and Present-status. These are status descriptors necessary to distinguish potentially ambiguous situations that can occur during search and present operations.

Result-set-status occurs if and only if the value of Search-status is 'failure', and its value is one of the following:

subset	-	Partial, valid results available.
interim	-	Partial results available, not necessarily valid.
none	-	No result set.

Present-status occurs if and only if the value of Search-status is 'success', and its value is one of the following:

success	-	All of the expected response records are available.
partial-1	-	Not all of the expected response records can be returned because the request was terminated by access control.
partial-2	-	Not all of the expected response records can be returned because they will not fit within the preferred message size.
partial-3	-	Not all of the expected response records can be returned because the request was terminated by resource control, at origin request.
partial-4	-	Not all of the expected response records can be returned because the request was terminated by resource control, by the target.
failure	-	None of the expected response records can be returned. One or more non-surrogate diagnostic records is returned (see note in 3.2.2.1.7).

3.2.2.1.12 Additional-search-information. On the response the target may use this parameter to convey information that is a by-product of the search process, including, for example, intermediate result counts, why particular records were returned, or whether a particular attribute was used for searching a database. On the request the origin may use this parameter to indicate the preferred format or content of that information. User Information format SearchResponse-1 is defined in Annex 9, USR. This parameter may be used only when version 3 is in force.

3.2.2.1.13 Other-information. This parameter may be used by either the origin or target for additional information not specified by the standard. This parameter may be used only when version 3 is in force.

3.2.2.1.14 Reference-id. See 3.4.

3.2.3 Retrieval Facility

The Retrieval facility consists of two services: Present and Segment.

The origin sends a Present request to request response records according to position within a result set maintained by the target. The target responds by sending a Present response, containing the requested response records. Alternatively, if segmentation is in effect and the requested response records will not fit within the Present response message, the target may segment the response by sending one or more Segment requests before the Present response. The procedures for segmentation are described in 3.3.

The Segment requests (if any) together with the Present response are referred to as the *aggregate Present response*. Each Segment request as well as the Present response is referred to as a *segment* of the Present response. If an aggregate Present response consists of a single segment (i.e., only a Present response) it is called a *simple Present response*.

3.2.3.1 Present Service

The Present service allows the origin to request response records corresponding to database records represented by a specified result set. Database records are referenced by relative position within the result set. The origin specifies a range and may follow with subsequent requests specifying different ranges. See Table 3.

Table 3: Parameters of the Present Service

Parameter	Origin Request	Target Response
Number-of-records-requested	x	
Result-set-start-position	x	
Additional-ranges	x (opt.)	
Result-set-id	x	
Element-set-names	x (opt.)	
Preferred-record-syntax	x (opt.)	
Comp-spec	x (opt.)	
Max-segment-count	x (opt.)	
Max-segment-size	x (opt.)	
Max-record-size	x (opt.)	
Response-records		x (if appl.)
Number-of-records-returned		x
Next-result-set-position		x
Present-status		x
Other-information	x (opt.)	x (opt.)
Reference-id	x (opt.)	x (if appl.)

Notes:
1. If version 3 is in force, a single request may include more than one range.
   The origin may request, for example, records one through five and follow with a request for records four through six.
2. In this section, “record N” means “the response record corresponding to the database record identified by result set entry N.”

3.2.3.1.1 Number-of-records-requested and Result-set-start-position. The origin requests a range of records: N records beginning at record M. M = Result-set-start-position, N = Number-of-records-requested and N is not greater than (Result-count - M) + 1.

3.2.3.1.2 Additional-ranges. The origin may request additional ranges of records by including this parameter, which consists of one or more pairs (M, N) where M and N are as described in 3.2.3.1.1. For the first pair (M, N) M must be greater than or equal to the sum of Result-set-start-position and Number-of-records-requested. For any consecutive pairs (M1, N1) and (M2, N2), M1 + N1 must be less than M2. This parameter may occur only when version 3 is in force.

3.2.3.1.3 Result-set-id. The origin indicates the name of a transient result set, created during this Z-association, from which records are to be retrieved.

3.2.3.1.4 Element-set-names. The origin may indicate the desired composition of the retrieved records. See 3.6.2.

3.2.3.1.5 Preferred-record-syntax. See 3.2.2.1.5.

3.2.3.1.6 Comp-spec. This parameter may be included only if the parameter Element-set-names is omitted, and only if version 3 is in force. If included, Comp-spec provides an alternative means of specifying the desired composition of retrieved records. See 3.6.

3.2.3.1.7 Max-segment-count, Max-segment-size, and Max-record-size. These three parameters may be used only when version 3 is in force.

Max-segment-count may be included when level-1 or level-2 segmentation is in effect; it specifies the maximum number of segments the target may include in the aggregate Present response. If its value is 1, no segmentation is applied for the operation and Max-record-size should not be included.

Max-segment-size and/or Max-record-size may be included only when level 2 segmentation is in effect. Max-segment-size is the largest allowable segment; if included, it overrides Preferred-message-size (for this Present operation only); if not included it assumes the value of Preferred-message-size. Max-record-size is the largest allowable retrieval record within the aggregate Present response; if included, it must equal or exceed Max-segment-size.

These three parameters are further detailed in 3.3.3.2.

3.2.3.1.8 Response-records. This parameter consists of a sequence of response records, or possibly, if 'level 2 segmentation' is in effect, a final fragment (see 3.3.3) followed by zero or more response records. Alternatively (if the operation included no Segment requests) the parameter consists of one or more non-surrogate diagnostic records indicating that the request cannot be processed, and why not (see note below).

A response record will be returned in the aggregate Present response for each record requested in the request (subject to message size, access-control, and resource-control constraints). Each response record corresponds to a result set entry, and the result set ordinal positions represented by the response records will be ascending and consecutive, unless the request included the parameter Additional-ranges. In this case, the positions will be ascending but may have gaps which will correspond exactly to the gaps in the requested ranges.

Each response record may optionally be accompanied by the name of the database to which it corresponds. However, the database name must accompany the first response record (or starting fragment) within the first segment of the aggregate Present response, and must accompany any response record (or starting fragment of a response record) from a database different from its immediate predecessor within the aggregate Present response.

When the origin has received the aggregate Present response, the result (if all of the segments are reassembled, and segmented response records reassembled from their fragments) will be one of the following:

•N response records, where N = Number-of-records-requested,

•a number of response records that is less than N (reason specified by Present-status), or

•one or more diagnostic records (see note) indicating that the request cannot be processed, and why not.

Note: If version 2 is in force, the target returns a single non-surrogate diagnostic record. If version 3 is in force, the target returns one or more non-surrogate diagnostic records.

3.2.3.1.9 Number-of-records-returned and Next-result-set-position. The parameter Number-of-records-returned is the total number of records in the aggregate Present response. Next-result-set-position is the value M+1, where M is the position of the result set item corresponding to the last record among those included in the response; or zero if M is the position of the last result set item.

3.2.3.1.10 Present-status. Present-status is mandatory in a Present response and its values are the same as those listed for Present-status in 3.2.2.1.11. Present-status refers to the aggregate Present response.

3.2.3.1.11 Other-information. This parameter may be used by the origin or target for additional information not specified by the standard. This parameter may be used only if version 3 is in force.

3.2.3.1.12 Reference-id. See 3.4.

3.2.3.2 Segment Service

If the records requested by a Present request will not fit in a single segment, and if segmentation is in effect, the target returns multiple segments, each of which contains a portion of the records. Each except the last segment is returned as a Segment request (the last segment is returned as a Present response).

Notes:

1. The segment service is modelled as a request, even though, logically, the target is not making a request. The reason is that (for purposes of abstract service definition and resultant protocol specification) any message is a request or a response, a response must be preceded by a request of the same type, and there may be at most one response to a given request. Because of these modelling constraints (a) the Segment service cannot be modelled as a response (because if it were, it would necessarily respond to a segment request, and it is a non-confirmed service); and (b) the present operation cannot be modelled as a Present request followed by multiple Present responses.
2. This service may be used only when version 3 is in force.
3. If segmentation is not in effect, the target does not send any Segment requests and the aggregate Present response consists of a simple Present response. If the records requested will not fit in a segment, the procedures described in 3.3.1 apply.
4. If the records requested will fit in a single segment (whether or not segmentation is in effect) the target does not send any Segment requests and the aggregate Present response consists of a simple Present response.

Table 4: Parameters of the Segment Service

	Target
Parameter	**Request**
Segment-records	x
Number-of-records-returned	x
Other-information	x (optional)
Reference-id	x (if applicable)

3.2.3.2.1 Segment-records. If level 1 segmentation is in effect, the parameter Segment-records consists of a sequence of response records.

If level 2 segmentation is in effect, the parameter Segment-records may include response records as well as fragments (see 3.3.3). It may be composed of a final fragment (except within the first segment of the aggregate Present response), followed by zero or more response records, followed by a starting fragment. Neither fragment need occur; however if neither occurs there must be at least one response record. (Note that fragments pertain only to retrieval records; a diagnostic record may not be segmented.)

The order of occurrence of a response record or fragment of a retrieval record is according to the order in which the record is identified by the result set. Each response record or starting fragment may optionally be accompanied by the name of the database to which it pertains. However, the database name must accompany the first response record (or starting fragment) within the first segment of the aggregate Present response, and must accompany any response record (or starting fragment of a retrieval record) from a database different from its immediate predecessor within the aggregate Present response.

3.2.3.2.2 Number-of-records-returned. This is the total number of response records and starting fragments included within the segment.

3.2.3.2.3 Other-information. This parameter may be used by the target for additional information not specified by the standard.

3.2.3.2.4 Reference-id. See 3.4.

3.2.4 Result-set-delete Facility

The Result-set-delete facility consists of a single service, Delete.

3.2.4.1 Delete Service

The Delete service enables an origin to request that the target delete specified result sets, or all result sets, created during the Z-association. The target responds by reporting information pertaining to the result of the operation. See Table 5.

Table 5: Parameters of the Delete Service

Parameter	Origin Request	Target Response
Delete-function	x	
Result-set-list	x (if appl.)	
Delete-operation-status		x
Delete-list-statuses		x (if appl.)
Number-not-deleted		x (if appl.)
Bulk-statuses		x (if appl.)
Delete-msg		x (opt.)
Other-information	x (opt.)	x (opt.)
Reference-id	x (opt.)	x (if appl.)

3.2.4.1.1 Delete-function. The origin specifies one of the following:

list	-	delete specified result sets (see 3.2.4.1.2), or
bulk-delete	-	delete all result sets currently on the target created during this Z-association.

3.2.4.1.2 Result-set-list. This parameter occurs if and only if Delete-function is ‘list’. It contains a list of result sets (created during this Z-association) to be deleted.

3.2.4.1.3 Delete-operation-status. Delete-operation-status is the status of the delete request. It assumes one of the values ‘success’ or ‘failure-3’ through ‘failure-9’ in Table 6.

Table 6: Delete Statuses

Status	Description
success	Result set(s) deleted.
failure-1	Result set did not exist.
failure-2	Result set previously unilaterally deleted by target.
failure-3	System problem at target (optional text message may be included in the Delete-msg parameter).
failure-4	Access-control failure: the delete request caused the target to issue an Access-control request which the origin failed to satisfy, or the origin could not accept an Access-control request.
failure-5	Operation terminated by resource control at origin request.
failure-6	Operation terminated by target due to resource constraints.
failure-7	Bulk delete of result sets not supported by target.
failure-8	Not all result sets deleted (on a bulk-delete request) (see 3.2.4.1.5).
failure-9	Not all requested result sets deleted (on a list request).
failure-10	Result-set in use.

Notes:

1. failure-7 and failure-8 can occur only if Delete-operation is Bulk-delete.
2. failure-10 may be used only when version 3 is in force.

3.2.4.1.4 Delete-list-statuses. Delete-list-statuses is present in a Delete response if Delete-function in the request was ‘list’. Delete-list-statuses contains the same list of result sets as in the Result-set-list parameter of the Delete request, each paired with a status. Possible status values are ‘success’, ‘failure-1’ through ‘failure-6’, and ‘failure-10’. See Table 6.

3.2.4.1.5 Number-not-deleted and Bulk-statuses. These two parameters occur only if Delete-function is Bulk-delete and if Delete-operation-status = ‘failure-8’. The parameter Number-not-deleted indicates how many result sets were not deleted, and the parameter Bulk-statuses gives individual statuses for those not deleted.

Note, however, that a target is not obligated to provide statuses for each result set not deleted on a bulk delete. For example, a target may abort a bulk delete when the first failure to delete a result set that is part of the bulk delete fails; in this case only a single status might be included in the parameter Bulk-statuses.

If a bulk delete results in more statuses than can fit into a single Delete-response message, the target may discard those that do not fit.

3.2.4.1.6 Delete-msg. Delete-msg, if present, contains an optional text message.

3.2.4.1.7 Other-information. This parameter may be used by either the origin or target for additional information not specified by the standard. This parameter may be used only when version 3 is in force.

3.2.4.1.8 Reference-id. See 3.4.

3.2.5 Access Control Facility

The Access Control facility consists of a single service, Access-control.

3.2.5.1 Access-control Service

The Access-control service allows a target to challenge an origin. The challenge might pertain to a specific operation or to the Z-association. The Access-control request/response mechanism can be used to support access control challenges or authentication, including password challenges, public key cryptosystems, and algorithmic authentication.

An origin must be prepared to accept and respond to Access-control requests from the target if access control is in effect. A target may issue an Access-control request which is either part of a specific (active) operation, or which pertains to the Z-association.

•If concurrent operations is in effect:

— If the Access-control request includes a Reference-id: The supplied Reference-id must correspond to an active operation; the Access-control request is part of that operation. The Access-control response must also include that Reference-id.

— If the Access-control request does not include a Reference-id: The Access-control request and response are not part of any operation, they pertain to the Z-association.

•If serial operations is in effect: The target may issue an Access-control request only when there is an active operation; the Access-control request and subsequent response are part of that operation and must include the Reference-id of the operation (which is assumed 'null' if not present in the initiating request).

The following procedures pertain to access control as it applies to an operation:

1. After sending an initiating request, the origin must be prepared to receive an Access-control request (for that operation), respond with an Access-control response, then later receive another Access-control request, etc., before receiving a terminating response. The target might suspend processing of the operation from the time that it sends the Access-control request until it receives the Access-control response. The challenge does not interrupt any other operation. If the origin response is acceptable to the target, the operation proceeds as if the challenge has never taken place. If the origin fails to respond correctly to the challenge then the target's terminating response to the interrupted operation may indicate that the operation was terminated du e to an Access-control failure.
2. If the origin fails to respond correctly to a challenge during an Init operation, the target may reject the Z-association (by setting the Result parameter to 'reject', and optionally supplying an explanatory message in the User-information-field of the Init response). However, the target need not necessarily reject the Z-association. For example the target might wish to invoke a security challenge during an Init operation to determine whether the origin is authorized to use a capability it has proposed. If the origin fails to respond properly, the target may simply refuse the use of that particular operation (via the Options parameter).
3. During a Search or Present operation, while the target is preparing records for presentation, it might send an Access-control request pertaining to a particular record. If the origin fails to respond correctly to the challenge, the target may simply substitute a surrogate diagnostic: "security challenge failed; record not included."

The following procedures pertain to access control as it applies to the Z-association:

1. If concurrent operations is in effect, following initialization the origin must be prepared at any time during the association, whether or not operations are active, to receive an Access-control request pertaining to the Z-association, to respond with an Access-control response, then later to receive another Access-control request, etc.
2. The target might suspend processing of some or all of the active operations from the time that it sends the Access-control request until it receives the Access-control response. If the origin response is acceptable to the target, the suspended operations proceed as if the challenge had never taken place.
3. If the origin fails to respond correctly to the challenge, the target might decide to terminate one or more operations but to leave open the Z-association. In that case, the target's terminating response to any such operations may indicate that the operation was terminated because of an Access-control failure. Alternatively, the target may close the Z-association. See Table 7.

Table 7: Parameters of the Access-control Service

Parameter	Target Request	Origin Response
Security-challenge	x	
Security-challenge-response		x
Other-information	x (opt.)	x (opt.)
Reference-id	x (if appl.)	x (if appl.)

3.2.5.1.1 Security-challenge and Security-challenge-response. Definitions for format and content of the challenge and response are subject to registration; several definitions are defined and registered in Annex 7, ACC. Alternatively, the contents of these two parameters may be established by prior agreement between a given target/origin pair.

3.2.5.1.2 Other-information. This parameter may be used by either the origin or target for additional information not specified by the standard. This parameter may be used only when version 3 is in force.

3.2.5.1.3 Reference-id. If serial operations is in effect, or if concurrent operations is in effect and the challenge pertains to a particular operation, then the use of Reference-id is governed by section 3.4. If 'concurrent operations' is in effect and the challenge pertains to the Z-association, then the Reference-id is to be omitted from both the request and response.

3.2.6 Accounting/Resource Control Facility

The Accounting/Resource Control facility consists of three services:

1. the Resource-control service, invoked by the target, either as part of an active operation (of any type) or pertaining to the Z-association;
2. the Trigger-resource-control service, invoked by the origin as part of an active operation (of any type except Init), and
3. the Resource-report service, invoked by the origin to initiate a Resource-report operation.

The Resource-control service permits the target to send a Resource-control request, which might include a resource report. The report might notify the origin that either actual or predicted resource consumption will exceed agreed upon limits (or limits built into the target), and request the origin's consent to continue an operation, via the Resource-control response. The target might, for example, inform the origin about the current status of a result set being generated on the target during a Search operation, and indicate information about the progress of the operation.

The Trigger-resource-control service permits the origin to request that the target initiate the Resource-control service, or cancel the operation.

The Resource-report service permits the origin to request that the target send a Resource-report pertaining to a completed operation or to the Z-association.

3.2.6.1 Resource-control Service

An origin must be prepared to accept and respond to Resource-control requests from the target if resource control is in effect. A target may issue a Resource-control request which is either part of a specific (active) operation or which pertains to the Z-association.

- If concurrent operations is in effect:
  - — If the Resource-control request includes a Reference-id: The supplied Reference-id must correspond to an active operation; the Resource-control request is part of that operation. The Resource-control response (if any) must also include that Reference-id.
  - — If the Resource-control request does not include a Reference-id: The Resource-control request and response are not part of any operation, they pertain to the Z-association.
- If serial operations is in effect: The target may issue a Resource-control request only when there is an active operation; the Resource-control request and (possible) subsequent response are part of that operation and must include the Reference-id of the operation (which is assumed 'null' if not present in the initiating request).

The Resource-control request indicates whether a response is required:

- If so, the origin must issue a Resource-control response. If the Resource-control request was part of an operation the response is part of the same operation; the target awaits the Resource-control response, and subsequently issues a terminating response after processing of the operation is concluded.
- If not, the origin must not issue a Resource-control response. If the Resource-control request was part of an operation the target subsequently issues the terminating response, after processing of the operation is concluded.

An origin should be prepared to receive, and (conditionally) respond to, multiple Resource-control requests as part of an operation (while the operation is active), or pertaining to the Z-association.

If the origin responds to a Resource-control request with a Resource-control response saying to terminate an operation, it can expect to receive a terminating response. This response might indicate that the operation was terminated at origin request. However, the response might alternatively indicate that the operation completed, since the operation at the target may continue to execute and subsequently complete before the Resource-control response reaches the target. See Table 8.

Table 8: Parameters of the Resource-control Service

Parameter	Target Request	Origin Response
Resource-report	x (opt.)	
Partial-results-available	x (if appl.)	
Suspended-flag	x (if appl.)	
Response-required	x	
Triggered-request-flag	x (opt.)	
Continue-flag		x
Result-set-wanted		x (if appl.)
Other-information	x (opt.)	x (opt.)
Reference-id	x (if appl.)	x (if appl.)

3.2.6.1.1 Resource-report. This parameter may be used to convey information about the current and estimated resource consumption at the server. The format of Resource-report resource-1 and resource-2 are defined in Annex 6, RSC.

3.2.6.1.2 Partial-results-available. The target indicates the status of the result set via the flag Partial-results-available, whose value is one of the following:

subset	-	Partial, valid results available.
interim	-	Partial results available, not necessarily valid.
none	-	No results available.

This parameter is meaningful only as part of a search operation. If its value is 'subset' or 'interim', then the target will accept subsequent Present requests against the result set if the origin indicates (via the Continue-flag) that the operation is to be terminated and if the value of the parameter Result-set-wanted is 'on'.

If the value of Partial-results-available is 'none' then the target need not accept subsequent Present requests in the event that the origin indicates (via the Continue-flag) that the operation is to be terminated.

Note that if the Suspended-flag is off, the partial results available situation may change because processing of the Search operation may continue. In all cases, the values of Search-status and Result-set-status in the Search response should be treate d as the authoritative information.

3.2.6.1.3 Suspended-flag. This parameter is valid only when the request pertains to an operation. The target indicates whether or not processing of the operation has been suspended pending the Resource-control response. This flag occurs if and only if the value of Response-required is 'yes'.

3.2.6.1.4 Response-required. The target indicates whether or not a response (from the origin) to this request is required.

3.2.6.1.5 Triggered-request-flag. This parameter is valid only when the request pertains to an operation. The target may optionally indicate whether or not this request resulted from a Trigger-resource-control request from the origin.

3.2.6.1.6 Continue-flag. This parameter is valid only when the request pertains to an operation. The origin indicates to the target whether or not to continue processing the operation.

3.2.6.1.7 Result-set-wanted. This flag is valid only

•during a Search operation,

•when the value of Partial-results-available is 'subset' or 'interim', and

•when the value of the parameter Continue-flag is 'do not continue'.

If the value of this flag is 'yes', the target will maintain the (possibly partial) result set for subsequent Present operations. If the value of the flag is 'no', the target may delete the result set. A result set status of 'none' on the subsequent Search response indicates that the target has discarded the result set. In all cases, the values of Search-status and Result-set-status in the Search response describe the actual decisions made by the target and the way in which the search terminated.

3.2.6.1.8 Other-information. This parameter may be used by either the origin or target for additional information, not specified by the standard. This parameter may be used only when version 3 is in force.

3.2.6.1.9 Reference-id See 3.4.

3.2.6.2 Trigger-resource-control Service

An origin may issue Trigger-resource-control requests during an operation (except during an Init operation), as part of that operation. It serves as a signal to the target that the origin wishes the target to:

•simply send a Resource-report, i.e., issue a Resource-control request with Response-required 'off';
•invoke full resource control, i.e., issue a Resource-control request with Response-required 'on'; or
•cancel the operation.

The target is not obliged to take any specific action upon receipt of a Trigger-resource-control request. For the purpose of procedure description, there is no response to the request; if the target wishes to issue a Resource-control request it does so unilaterally. (If the origin issues a Trigger-resource-control request and subsequently receives a Resource-control request as part of the same operation, the origin cannot necessarily determine whether the latter resulted from the Trigger-resource-control request. However, the target may include Triggered-request-flag in the Resource-control-request to so indicate.)

If the origin issues a Trigger-resource-control request saying to cancel the operation, and if the target honors the request, the origin can expect to receive a terminating response indicating that the operation was terminated at origin request.

Although an origin may issue a Trigger-resource-control request as part of an active operation, the target might receive the request after the operation terminates. In that case, the target will ignore the Trigger-resource-control request. Furthermore, the target might receive a Trigger-resource-control request after issuing a Resource-control request for the same operation while awaiting a Resource-control response. In that case, again, the target should ignore the Trigger-resource-control request. (Note that in general, the target may ignore any Trigger-resource-control request.) See Table 9.

Table 9: Parameters of the Trigger-resource-control Service

Parameter	**Origin Request**
Requested-action	x
Preferred-resource-report-format	x (if appl.)
Result-set-wanted	x (if appl.)
Other-information	x (optional)
Reference-id	x (if appl.)

3.2.6.2.1 Requested-action. The origin indicates one of the following:

resource-report	-	issue a Resource-control request and set Response-required to 'off'.
resource-control	-	issue a Resource-control request and set Response-required to 'on'.
cancel	-	terminate the operation.

3.2.6.2.2 Preferred-Resource-report-format. The origin may indicate a resource report format that it prefers.

3.2.6.2.3 Result-set-wanted. This flag is meaningful only for a Search operation, and when the requested action is 'cancel'. If the value of the flag is 'yes', the origin requests that the target maintain the (possibly partial) result set for subsequent Present operations. See 3.2.6.1.7.

3.2.6.2.4 Other-information. This parameter may be used by the origin for additional information, not specified by the standard. This parameter may be used only when version 3 is in force.

3.2.6.2.5 Reference-id. See 3.4.

3.2.6.3 Resource-report Service

The Resource-report service allows an origin to request a Resource-report, pertaining to a specified, completed operation, or to the entire Z-association.

Note: The Resource-report service differs from the Trigger-resource-control service in this respect: Trigger-resource-control is a non-confirmed service; there is a request, but no response. The request is part of, but does not initiate, an operation; it requests a report pertaining to that active operation. Resource-report, in contrast, is a *confirmed* service; there is a request and a response (the target is obliged to respond, although the target is not obliged to include a resource report in the response). The request and response initiate and terminate an operation respectively; the request identifies a particular *completed* operation and solicits a report pertaining to that operation (or it may solicit a report pertaining to the entire Z-association). See Table 10.

Table 10: Parameters of the Resource-report Service

Parameter	Origin Request	Target Response
Preferred-resource-report-format	x (opt.)	
Op-id	x (opt.)	
Resource-report-status		x
Resource-report		x (opt.)
Other-information	x (opt.)	x (opt.)
Reference-id	x (opt.)	x (if appl.)

3.2.6.3.1 Preferred-resource-report-format. The origin may indicate a resource report format that it prefers.

3.2.6.3.2 Op-id. This parameter may be supplied by the origin to identify a completed operation for which the origin requests a resource report. This parameter may be used only when version 3 is in force.

- If Op-id is present, it consists of a Reference-id, and refers to the most recently completed operation that used that Reference-id.

  Notes:

  1. When an operation terminates, if the origin anticipates that it will subsequently issue a Resource-report request pertaining to that operation, it is the origin's responsibility to ensure that the Reference-id is not reused before doing so.
  2. The origin may (but need not) use the same reference-id for the Resource-report operation as that specified in Op-id, and if so, Op-id will nevertheless pertain to a completed operation only. However, it is recommended that the origin not specify a value of Op-id equal to any reference-id being used by any active operation other than this Resource-report operation. If the origin does so, the target may (but need not) consider the request in error (see failure-6 of Resource-report-status).
  3. If the origin wants resource information about an *active* operation, it should not use the Resource-report service, but instead use the Trigger-resource-control service as part of that operation. If the operation terminates before the target receives the Trigger-resource-control request, the origin will receive a terminating response and may then subsequently issue a Resource-report request pertaining to that (completed) operation.

- If Op-id is not present, the origin requests a resource report pertaining to the Z-association.

3.2.6.3.3 Resource-report-status. The target supplies one of following status values:

success	-	A resource report is included (and in the preferred format, if the parameter Preferred-resource-report-format was included in the request).
partial	-	A resource report is included, but not in the preferred format (applies only if the parameter Preferred-resource-report-format was included in the request).
failure-1	-	Target unable to supply resource report.
failure-2	-	Operation terminated by target due to resource constraints.
failure-3	-	Access-control failure.
failure-4	-	Unspecified failure.
failure-5	-	There is no known operation with specified id.
failure-6	-	There is an active operation with specified id.

Note: Failure-5 and failure-6 apply only when version 3 is in force.

3.2.6.3.4 Resource-report. See 3.2.6.1.1.

3.2.6.3.5 Other-information. This parameter may be used by either the origin or target for additional information not specified by the standard. This parameter may be used only when version 3 is in force.

3.2.6.3.6 Reference-id. See 3.4.

3.2.7 Sort Facility

The Sort facility consists of a single service, Sort.

3.2.7.1 Sort Service

The Sort service allows an origin to request that the target sort a result set (or merge multiple result sets and then sort). The origin specifies a sequence of sort elements. The result set is to be ordered according to the specified sequence, and subsequent positional requests against the result set will be construed by the target to apply to the result set as so ordered. See Table 11.

Table 11: Parameters of the Sort Service

Parameter	Origin Request	Target Response
Input-result-sets	x	
Sorted-result-set	x	
Sort-sequence	x	
Sort-status		x
Result-set-status		x (if appl.)
Diagnostics		x (if appl.)
Other-information	x (opt.)	x (opt.)
Reference-id	x (opt.)	x (if appl.)

3.2.7.1.1 Input-result-sets. This parameter is the name of a result set to be sorted, or the names of result sets to be merged and the result sorted.

3.2.7.1.2 Sorted-result-set. This parameter is the name of the sorted result set. It may be the name of an existing result set (including one of the names included in Input-result-set); if so, then if the sort is processed, the existing result set is deleted, and a new result set by that name is created; its content is the sorted results. If Sorted-result-set is not the name of an existing result set and if the sort is processed, a result set by the specified name is created by the target, whose content is the sorted results; the content of the Input-result-sets is left unchanged. In any case, if the sort is not processed, the final content of Sorted-result-set is indicated by the parameter Result-set-status.

3.2.7.1.3 Sort-sequence. The parameter Sort-sequence comprises the elements that are to be used for sorting, together with the direction of the sort (ascending or descending), case sensitivity (if applicable), and target action if an element is missing from a record in the result set to be sorted. Each of the sort elements is a set of attributes, a sort-field-designator, or an element specification, that the target has designated (see note below) for use as a sort key.
Note: The target designates this information either via the Explain facility, or through some mechanism outside of the standard.

3.2.7.1.4 Sort-status. The parameter Sort-status, returned by the target, assumes one of the following values:

success	-	The sort was performed successfully.
partial-1	-	The sort was performed but the target encountered records with missing values in one or more sort elements.
failure	-	The sort was not performed. The target supplies one or more diagnostics in the parameter Diagnostics.

3.2.7.1.5 Result-set-status. The target supplies this parameter if and only if the value of Sort-status is 'failure'. It refers to the contents of Sorted-result-set, and its value is one of the following:

empty	-	The result set is empty.
interim	-	Partial results available, not necessarily valid.
unchanged	-	The content of the result set is unchanged (applies only if Sorted-result-set is one of the input result sets).
none	-	Result set not created (applies only if Sorted-result-set is not one of the input result sets).

3.2.7.1.6 Diagnostics. The target includes this parameter if the value of Sort-status is 'failure'. It includes one or more diagnostic records.

3.2.7.1.7 Other-information. This parameter may be used by the origin or target for additional information not specified by the standard.

3.2.7.1.8 Reference-id. See 3.4.

3.2.8 Browse Facility
The Browse facility consists of a single service, Scan.

3.2.8.1 Scan Service
The Scan service is used to scan an ordered term-list (subject terms, names, titles, etc.). The ordering of the term-list is target defined. The origin specifies a term-list to scan and a starting term (implicitly, by specifying an attribute/term combination and a database-id), the size of the scanning steps, and the desired number of entries and position of the starting term in the response. See Table 12.

Table 12: Parameters of the Scan Service

Parameter	Origin Request	Target Response
Database-names	x	
Term-list-and-start-point	x	
Step-size	x (opt)	x (if appl)
Number-of-entries	x	x
Position-in-response	x (opt)	x (opt)
Scan-status		x
Entries		x (opt)
Other-information	x (opt)	x (opt)
Reference-id	x (opt)	x (if appl.)

3.2.8.1.1 Database-names. The parameter Database-names identifies a set of databases to which the term-list (specified by Term-list-and-start-point) pertains.

3.2.8.1.2 Term-list-and-start-point. The origin supplies an attribute list and term. The attribute list contains attributes indicating which term-list to scan. The term, as qualified by those attributes, indicates where scanning begins; this will be a presumed entry in the term-list. If there is no matching entry, the first entry with higher value is to be the starting point.

As an example, to scan a list of personal names: the attribute list might consist of a single attribute whose type is 'use' and whose value is 'personal name'; the term would specify a personal name; the database-id would identify one or more databases to which the list of personal names pertains.

3.2.8.1.3 Step-size. The origin may specify the desired number of entries in the term-list between two adjacent entries in the response. A value of zero means "do not skip any entries." If the target cannot support the requested step size, it sets Scan-status to 'failure' and includes a non-surrogate diagnostic such as "only step size of zero supported" or "requested step size not supported." If the origin omits this parameter, the step size is selected by the target, and the target includes the selected step size in the response.

3.2.8.1.4 Number-of-entries. The origin indicates the proposed number of entries to be returned. The target indicates the actual number of entries returned. If the actual number is less than the proposed number, the reason is indicated in Scan-status.

3.2.8.1.5 Position-in-response. The origin may optionally indicate the preferred position, within the returned entries, of the specified starting point value. A value of 1 refers to the first of the returned entries. A value of 0 means that the returned entries should begin with the term immediately following the starting point term. A value of Number-of-entries + 1 means that the origin requests terms immediately preceding the starting point term.

The target may indicate the actual position of the chosen starting point within the returned entries.

Example: If the values of the request parameters Number-of-entries and Position-in-response are 10 and 3 respectively, then the origin requests two terms immediately preceding the starting point value, followed by the starting point value, followed by the immediately-following seven terms.

Note: If response parameter Position-in-response is less than the value proposed in the request, the origin may conclude that there were fewer terms than expected in the low end of the term-list. However, if Position-in-response is the same value in the response as proposed in the request, but Number-of-entries in the response is less than the value proposed in the request, the origin may not conclude that there were fewer terms than expected at the high end of the term-list, unless Scan-status is Partial-5. The reason that fewer terms than expected are returned is indicated in the Scan-status.

3.2.8.1.6 Scan-status. The target indicates the result of the operation. The defined values are:

success	-	The response contains the number of entries (term-list-entries or surrogate diagnostics) requested.
partial-1	-	Not all of the expected entries can be returned because the operation was terminated by access-control.
partial-2	-	Not all of the expected entries will fit in the response message.
partial-3	-	Not all of the expected entries can be returned because the operation was terminated by resource-control, at origin request.
partial-4	-	Not all of the expected entries can be returned because the operation was terminated by resource-control, by target.
partial-5	-	Not all of the expected entries can be returned because the term-list contains fewer entries (from either the low end, high end, or both ends of the term-list) than the number of terms requested.
failure	-	None of the expected entries can be returned. One or more non-surrogate diagnostics is returned.

3.2.8.1.7 Entries. The parameter Entries returned by the target consists of one of the following:

- •N entries, where each entry is a term-list-entry or surrogate diagnostic, where N = Number-of-entries in the request.
- •A number of entries which is less than N, and may be zero (reason specified by Scan-status).

It may also include:

- •One or more non-surrogate diagnostic records (possibly indicating that the operation cannot be processed, and why it cannot).

Each term-list-entry includes a term (occurring in one of the databases specified in the parameter Database-names), and optionally the following:

- •A display term (when the actual term is not considered by the target to be suitable for display).
- •A list of suggested attributes for use in subsequent Scan requests (useful for scanning multiple indices, e.g., author and title, at the same time).
- •A suggested alternative term.
- •Occurrence-information: this might include a count of records in which the term occurs. It may also list counts for specific attributes, possibly further broken down by database. Alternatively, a term-list-entry might list databases in which the term occurs, and for associated attributes, but no counts.
- •Other information: additional information concerning the entry.

3.2.8.1.8 Other-information. This parameter may be used by the origin or target for additional information, not specified by the standard.

3.2.8.1.9 Reference-id. See 3.4.

3.2.9 Extended Services Facility

The Extended Services facility consists of a single service, Extended-services.

3.2.9.1 Extended Services Service

The Extended-Services (ES) service allows an origin to create, modify, or delete a *task package* at the target. The target maintains task packages in a special database, described in section 3.2.9.2. A task package pertains to an ES *task*.

An *extended service* is a task *type*, related to information retrieval, but not defined as a Z39.50 service. Execution of a task by the target is outside the scope of ISO 23950. The extended services defined by this standard are listed in section 3.2.9.1.2. Definitions of those services are included in Annex 8, EXT.

The origin sends an ES Request to the target requesting execution of a task. The request includes parameters which the target uses to construct the task package. The target checks the request for validity, for consistency with the user's access privileges, and possibly for other target-dependent limitations. The target sends an ES response indicating that the request was accepted or supplying an indication of the reason the request was rejected.

The ES service is a confirmed service, initiated by the origin. The ES operation consists of a request from the origin and a response from the target, possibly with intervening Access-control or Resource-control messages. However, although the request may result in the initiation of a task, the task is not considered part of the Z39.50 ES operation. The target response, which completes the ES operation, does not necessarily signal completion of the task. A task may have a lifetime that exceeds a single Z-association.

Execution of the ES Operation results in the creation of a task package, represented by a database record in the ES database.

For example, when a target creates a task package of type PersistentResultSet, a (persistent) result set is created, represented by the created task package, in the form of a record in the extended services database. When that package is subsequently retrieved by an origin, in either the same or a different Z-association, a copy of that persistent result set is made available to that Z-association, as a Z39.50 result set (i.e., as a transient result set; a result set name for use during the Z-association is included within the task package). When an origin deletes the task package, the persistent result set is deleted.

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 particular extended service. Among the common parameters (indicated in Table 13, listed under "task package parameter" in the right column), some are supplied by the origin as parameters in the ES request, and are used by the target to form the task package; some of those supplied by the origin may be overridden by the target. Others are supplied by the target. The specific parameters are derived from the parameter Task-specific-parameters of the ES request (see Annex 8, EXT).
Note: The response parameter Task-package below refers to the actual task package, and if it occurs (see 3.2.9.1.13), it includes some or all (depending on the parameter Elements) of the parameters listed under "task package parameter."

3.2.9.1.1 Function. The origin specifies Create, Delete, or Modify. If the function is Create, the target is to create a task package, and assign to it the name specified by the parameter Package-name, if supplied.

Table 13: Parameters of the Extended Services Service

Parameter	Origin request	Target response	Task package parameter
Function	x		
Package-type	x		
Package-name	x (opt)		x (opt)
User-id	x (opt)		x (opt)
Retention-time	x (opt)		x (opt)
Permissions	x (opt)		x (opt)
Description	x (opt)		x (opt)
Target-reference			x (opt)
Creation-date-time			x (opt)
Task-status			x
Package-diagnostics			x (opt)
Task-specific-parameters	x		(see note)
Wait-action	x		
Elements	x (if appl)		
Operation-status		x	
Operation-diagnostics		x (if appl)	
Task-package		x (if appl)	
Other-information	x (opt)	x (opt)	
Reference-id	x (opt)	x (if appl)	

Note: Task-specific-parameters are defined for each extended service. For each task-specific parameter, the definition states whether or not the parameter occurs in the task package.

If the function is Delete or Modify, the target is to delete or modify the task package specified by the parameter Package-name. A target that supports deletion or modification may nonetheless deny the request, for example, because the task is already in progress or the package is in use.

If the function is Delete, the origin requests that if the specified task has not been acted on, it should not be started; if the task is active, the target should either terminate the task or refuse the request.

If the function is Modify, the origin requests that parameter values in the request (as well as those within parameter Task-specific-parameters) replace the corresponding values in the task package. If an optional parameter is omitted, the target does not modify that parameter within the task package (thus to return a parameter to its default value, an origin must explicitly provide the default value).

3.2.9.1.2 Package-type. The Package-type identifies the extended service requested. The extended services defined by this standard (see Annex 8, EXT) are:

- •Save a result set for later use
- •Save a Query for later use
- •Define a periodic search schedule
- •Order an item
- •Update a database
- •Create an export specification
- •Invoke a previously created export specification.

3.2.9.1.3 Package-name. The origin may optionally supply a name for the task package to be created. If so, the triple (Package-type, User-id, Package-name) must be unique (i.e., there must be no other task package of that type, for that user with the same name, otherwise the request is in error), and that triple identifies the task package for subsequent reference. Package-name should be included if the origin intends to reference the task package.

3.2.9.1.4 User-id. The User-id identifies the user to be associated with the task package. If not supplied, this parameter may default to the Id of the current user. A target may or may not allow an origin to supply a user id different from its own.

3.2.9.1.5 Retention-time. The origin may optionally specify a retention period (e.g., 2 hours, 3 days, 1 week), which may be overridden by the target. When the retention time has passed, the target may delete the retained task package. A retention time of zero means the task package is not to be retained after the task is completed.

3.2.9.1.6 Permissions. The origin may indicate who may access the task package. If the origin does not supply this parameter, only the creating user may do so. See 3.2.9.3.

3.2.9.1.7 Description. The origin may include a description. It might describe, for example, the result set, for a Persistent Result Set task; or the query, for a Persistent Query task.

3.2.9.1.8 Target-reference. The target may supply a unique identifier for the task package.

3.2.9.1.9 Creation-date-time. The target supplies the date and time that the task package was created.

3.2.9.1.10 Task-status. The target indicates the status of the task. Values are 'pending', 'active', 'complete', and 'aborted'.

3.2.9.1.11 Package-diagnostics. The target may include one or more diagnostics in the task package.

3.2.9.1.12 Task-specific-parameters. These are additional parameters, defined by the specific extended service.

3.2.9.1.13 Wait-action. The origin indicates whether the target should (or may) include the task package in the ES response. This immediate response mechanism may avoid the need for follow-up Search and Present operations, or in general, for making the task package available through the extended services database (see section 3.2.9.2).

This parameter has four possible values:

- *wait:* the target must perform the task before issuing the ES response (unless the operation aborts; see section 3.2.9.4). If the target is not willing to perform the task before issuing the response it must refuse the request by responding with a status of 'failure' and an appropriate diagnostic. If the target accepts the request, it includes the parameter Task-package in the response.
- *wait-if-possible:* the origin requests that, if possible, the target perform the task before issuing the ES response and include the task package in the response. If not possible, the target should proceed as though the value were 'do not wait'.
- *do-not-wait:* The origin does not request that the target attempt to perform the task before issuing the ES response. However, if the target does perform the task before issuing the response, then the response may include the task package.
- *do-not-send-task-package:* The target may perform the task when it chooses, but is not to include the task package in the response under any circumstance.

3.2.9.1.14 Elements. The origin may optionally include this parameter if Wait-action is other than 'do-not-send-task-package'. It is an element set name for the task package in the event that it is returned in the response parameter Task-package.

3.2.9.1.15 Operation-status. This is the status of the ES operation. It is one of the following:

done	-	The request was accepted, the task is complete and results are included in Task-package.
accepted	-	The request was accepted and the task is queued for processing, or is in process.
failure	-	The request was refused. One or more diagnostics are supplied (in parameter Operation-diagnostics).

3.2.9.1.16 Operation-diagnostics. The target may supply additional diagnostic information if Operation-status is 'failure'.

3.2.9.1.17 Task-package. If Operation-status is 'done', the target includes the task package. The portion of the actual task package included depends on the parameter Elements.

3.2.9.1.18 Other-information. This parameter may be used by the origin or target for additional information, not specified by the standard.

3.2.9.1.19 Reference-id. See section 3.4.

3.2.9.2 The Extended Services Database

Targets that support the Extended Services facility provide access to a database with the name IR-Extend-1 (referred to as the "extended services database" or "ES database"). Records in the extended services database are task packages constructed from the Request-parameter-package parameter in ES requests (the target may begin execution of the task at any time after it accepts the request, which may be before the task package has been stored in the database). The target may (but need not) retain a task package until the requested task has completed; it may retain the task package until the origin requests that it be deleted. A target may unilaterally delete a task package from the ES database at any time.

Note: This means, as a practical matter, the target need not actually create a task package for a given task, particularly when the task is executed immediately. However, it is recommended that a task package exist when the status of the task is pending, active, or aborted.

When the target receives an ES request it may immediately create a task package, with status 'pending', before completely validating the request. The origin may thus search the database anytime after submitting a request (during the same or a subsequent Z-association), for a resulting task package. In particular, if an ES operation is aborted (see 3.2.9.4) the origin may be able to determine that the request for that operation was received.

An ES database may be listed in the target Explain database, with a list of extended services the target supports, allowable export destinations, options that an origin may supply for an export task, etc.

An extended services database will appear to the origin as any other database supported by the target (records may be searched and retrieved by the Z39.50 Search and Retrieval facilities; search processing is defined locally by the target; the target may impose access control or exclude records to which the origin is not authorized access). However, certain search terms are predefined in order to allow a semantic level of interoperability. The attribute set used to search the database is defined and registered in Annex 3, ATR. The task package structures are defined and registered in Annex 8, EXT.

The ES database may provide the following special element sets (in addition to "F"):

•Identification: includes the creating user's identification, the origin-supplied name of the task package, and possible permissions for other users to access the request. Other identifying information such as time of creation may be included.

•UniqueName: the creating user's identification and the name of the task package.

•Permissions: the contents of the UniqueName element set, and in addition, the granted permissions for the task package. A target might present the full permissions list only to the task package creator, presenting to other users only the permissions applicable to them.

•Status: a short summary of the current status of the request, perhaps including cost and other resource usage.

•Brief: Identification element set plus the most important elements of the Status element set.

3.2.9.3 Owners and Permissions

The creating user of a task package may apply any extended service function to the package, as well as retrieve the full package (via the Retrieval facility) and invoke the package via other extended services. (Invocation occurs, for example, when a Periodic Query task references a saved Query.)

Using the Modify function of the ES request, an origin can change the access permissions of a task package by supplying a new permissions list, which is a sequence of user ids and for each, a sequence of allowed operations, from the following set:

•Delete

•Modify-Contents

•Modify-Permissions

•Present

•Invoke.

As an example of the use of the 'invoke' permission, a target might create a task package, on behalf of a client user, of type PersistentQuery; a persistent query is created, represented by the created task package. The target may subsequently be requested to create a PeriodicQuerySchedule task package, on behalf of a different user, which refers to (i.e., "invokes") that persistent query task package. The target would do so only if that user has 'invoke' privilege for that persistent query. As another example, a target may create an ExportSpecification (package) on behalf of one user, and a different user may subsequently 'invoke' that ExportSpecification by creating an InvokeExportSpecification package, if that user has 'invoke' privilege for the ExportSpecification.

Targets may provide group names for use in permission lists, but a group name would be syntactically the same as a user Id. (The target might report the composition of groups, but the mechanism for doing so is not described by this standard.)

3.2.9.4 Aborted Operations

An origin may receive a response to an ES request only during the Z-association in which it issues the request (as for any other Z39.50 operation). If an ES operation is aborted (explicitly, or because the Z-association is closed or the A-association terminated), the origin will not receive a terminating response. This has no effect on the disposition or processing of the task, regardless of the value of Wait-action that was specified on the request. If an ES operation aborts, Wait-action automatically assumes the value 'do-not-send-task-package'.

If an ES operation is aborted, the origin may search the ES database (possibly in a subsequent Z-association) for information that would otherwise have been returned in the response.

3.2.10 Explain Facility

The Explain facility allows an origin to obtain details of the implementation of a target, including databases available for searching, attribute sets and diagnostic sets used by the target, and schema, record syntax and element specification definitions supported for retrieval. Targets that support the Explain facility:

•provide access (via the Z39.50 Search and Present services) to a database with the name IR-Explain-1 (referred to as the "Explain database");
•support the explain attribute set, exp-1, defined in Annex 3, ATR (which defines a set of Use attributes and imports bib-1 non-Use attributes); and
•support the Explain syntax, which is defined and registered in Annex 5, REC.

A record (or result set item representing a record) within the Explain database, is referred to as an "Explain record."

3.2.10.1 Searching the Explain Database

The Explain database appears to the origin as any other database supported by the target. However, certain search terms, corresponding to information categories, are predefined in order to allow a semantic level of interoperability. Terms are searched case-insensitive.

The exp-1 attribute set is used to search the Explain database. Combinations of Use attributes and terms allow searching upon information category; well-defined combinations of Use attributes may be used to allow additional specification by the origin to limit the records to those of immediate interest. Combinations of exp-1 Use attributes to perform a common set of searches are listed in 3.2.10.1.1 and 3.2.10.1.4. Since the Explain database may be searched as any other database using attributes from one or more attribute sets, this list is not exhaustive. However, it is recommended that a target supporting the Explain facility support this list of common searches. As described in 3.2.10.1.2 and 3.2.10.1.3, the HumanStringLanguage, DateAdded, DateChanged, and DateExpires attributes can be used in combination with any of the combinations listed in 3.2.10.1.1 and 3.2.10.1.4.

The exp-1 attribute set consists of a set of Use attributes and imports the non-Use bib-1 attributes. It is recommended that a target supporting the Explain facility support the bib-1 relation attribute 'equal' (see note), position attribute 'any position in field', and structure attribute 'key'.

Note: If the target intends to support searching based on date ranges (e.g., to limit a search to records created before or after a particular date or between two dates), the target should also support one or more of the following relation attributes: 'less than', 'less than or equal', 'greater than', and 'greater or equal'.

Origins should not in general expect that the explain database is searchable using the bib-1 truncation attribute, completeness attribute, or any of the alternative values of the relation, position, and structure attributes defined in bib-1. However, targets are free to provide access to the Explain database using those and other alternative attributes and attribute values.

3.2.10.1.1 Searching for Predefined Information Categories. Records corresponding to a particular explain information category are searched by an operand where the term is the name of that category; for example, all records corresponding to TargetInfo are searched using the term "TargetInfo." For each category one or more *key elements* are defined, and may be provided as search terms (using the appropriate attribute). A search with an operand where the Use attribute = 'ExplainCategory' and the term is a category, and with additional operands corresponding to each key for that category where the value of the Use attribute is the key, should result in (at most) a single record.

The primary mechanism for search and retrieval of information from the Explain database is for the origin to select the records in a category using the Use attribute 'ExplainCategory' and to extract desired information from those records to formulate a subsequent search. For example the origin may search records with ExplainCategory = 'DatabaseInfo', and retrieve summary information (see 3.2.10.2.2) from those records. Each summary record will include a database name, which serves as a key for a possible subsequent search.

A list and brief description of the Explain information categories (and thus search terms) as well as the keys for each category are given in Table 14. In 3.2.10.3 each category is described in detail.

An origin should adhere to the following rules when searching an Explain database by the predefined information categories:
•To search for information about the target, use ExplainCategory='TargetInfo'.
•To search for information about a specific database, use ExplainCategory='DatabaseInfo' in combination with the DatabaseName attribute to specify the key of the desired databaseInfo record.
•To search for information about a specific schema, use ExplainCategory='SchemaInfo' in combination with the SchemaOID attribute to specify the desired schema.
•To search for information about a specific tag set, use ExplainCategory='TagSetInfo' in combination with the TagSetOID attribute to specify the desired tag set.
•To search for information about a specific record syntax, use ExplainCategory= 'RecordSyntaxInfo' in combination with the RecordSyntaxOID attribute to specify the desired record syntax.
•To search for information about a specific attribute set, use ExplainCategory='AttributeSetInfo' in combination with the AttributeSetOID attribute to specify the desired attribute set.
•To search for information about term lists for a database, use ExplainCategory='TermList-Info' in combination with the DatabaseName attribute to specify the desired database.
•To search for information about a specific extended service, use ExplainCategory = 'ExtendedServicesInfo' in combination with the oid for that extended service.

- •To search for the attributes and combination of attributes which may be used in searching a database, use ExplainCategory=’AttributeDetails’ in combination with the DatabaseName attribute to specify the database for which attribute information is desired.
- •To search for information about a specific term list, use ExplainCategory = ‘TermListDetails’ in combination with the name for the term list.
- •To search for the element set names defined for a record syntax for a particular database, use ExplainCategory=’ElementSetDetails’ in combination with the RecordSyntaxOID attribute to specify the desired record syntax and the DatabaseName attribute to specify the desired database.
- •To search for the definition of a specific element set name, use ExplainCategory = ‘ElementSetDetails’ in combination with the ElementSetName attribute to specify the desired element set name. There may be multiple records located since the explain database contains one record for each element set name for each record syntax for each database.
- •To search for a particular element set name defined for a record syntax, for a particular database, use ExplainCategory = ‘ElementSetDetails’ in combination with the ElementSetName attribute to specify the desired element set name, the RecordSyntaxOID attribute to specify the desired record syntax and the DatabaseName attribute to specify the desired database.
- •To search for the description of the elements of a retrieval record, for a particular record syntax, in a specific schema, for a particular database, use ExplainCategory=’RetrievalRecordDetails’ in combination with the RecordSyntaxOID attribute to specify the desired record syntax, the SchemaOID attribute to specify the desired schema, and the DatabaseName attribute to specify the desired database.

3.2.10.1.2 Searching for Information in a Particular Language. Elements intended to be presented to the user by the origin are said to consist of “human-readable text.” Each record includes a language element indicating the language of the human-readable text within the record. The explain database might contain several records with identical information in different languages. To search for records in a certain language, the HumanStringLanguage attribute may be used (in conjunction with the three-character language code as the term; see Z39.53-1994).

For example, to search for a list of databases that have descriptive records in English, the query might be of the form:

(Category = ‘DatabaseInfo’) AND (HumanStringLanguage = ‘eng’).

The HumanStringLanguage attribute is intended primarily for use in Version 2. When version 3 is in force, the use of variants is recommended.

3.2.10.1.3 Searching for Information by Control Dates. To search for new records in an Explain database, use the DateAdded attribute; for updated records use the DateChanged attribute, for records based on their date of expiration use the DateExpires attribute. Any of these three may be used in combination with the searches described above.

3.2.10.1.4 Searching for Information Using Content Values. Some of the Explain records are searchable using attributes which take values from elements within the pertinent Explain records. These Use attributes can be used to select subsets of records of specific information category. For instance, the Availability Use attribute can be used to select those database information records for databases which are currently available. The use of these attributes by an origin should conform to the following rules:

- •To locate databases currently available, use the ExplainCategory attribute with term ‘DatabaseInfo’, in combination with the Availability attribute with term ‘yes’.
- •To locate the databases provided by a specific supplier, use the ExplainCategory attribute with term ‘DatabaseInfo’, in combination with the Supplier attribute with the supplier’s name as term.
- •To locate databases provided by a specific producer, use the ExplainCategory attribute with term ‘DatabaseInfo’ in combination with the Producer attribute with the producer’s name as term.
- •To locate databases that are not proprietary, use the ExplainCategory attribute with term ‘DatabaseInfo’, in combination with the Proprietary attribute with term ‘no’.
- •To locate databases that have no user fee, use the ExplainCategory attribute with term ‘DatabaseInfo’, in combination with the UserFee attribute with term ‘no’.

3.2.10.2 Retrieval of Explain Records

A Present request for Explain records should specify the Explain syntax as the Preferred-record-syntax. Each explain information category has its own record layout, and all are described in the Explain syntax definition (see Annex 5, REC.1).

Explain records include key elements that serve to uniquely identify each record. Each Explain category is defined in terms of key elements, non-key “brief” elements (see 3.2.10.2.2), “non-brief” elements, and possibly other categories. Key elements are always part of the brief elements.

3.2.10.2.1 Retrieval and Human-Readable Text. The Explain database might provide alternative variations of human-readable information (however, for language variations, see note below). For example, a text element might be retrievable in ASCII, SGML, or PostScript. To request a particular format, use the variant facilities of Version 3.
Note: For language variation, see 3.2.10.1.2. The Explain database logically includes different records for different languages, and therefore selection based on language occurs during the search.

Table 14: Explain Categories and Keys

Category	An Explain record in this category describes:	Key(s)
TargetInfo	*The target, including search constraints imposed by the target.*	target name
DatabaseInfo	*A database. Information about supported query types, attribute sets, record syntaxes, schemas, diagnostic sets, resource control formats and access control formats. A group of databases offering a common set of characteristics may be described as a single, logical, database. In this case, a list of databases subsumed within this logical database is provided.*	database name
SchemaInfo	*A Schema.*	schema oid
TagSetInfo	*A tag Set.*	tagSet oid
RecordSyntaxInfo	*A record syntax.*	record syntax oid
AttributeSetInfo	*An attribute set, including the attributes supported within the set.*	attribute set oid
TermListInfo	*Term lists supported for a database.*	database name
ExtendedServices Info	*An extended service.*	extended service oid
AttributeDetails	*Attributes that can be used to search a database including the other attributes with which it may be combined.*	database name
TermListDetails	*A term list.*	term list name
ElementSetDetails	*An element set (for a particular record syntax, for a particular database).*	database name, element set name, record syntax oid
RetrievalRecord Details	*The elements of a retrieval record (for a particular record syntax, defined by a particular schema).*	databaseName,
SortDetails	*Sort specification for a database.*	database name
Processing	*Processing instructions for a database, for a particular processing context, name of instructions, and object identifier for the abstract syntax of the externally defined Instructions.*	database name, name, oid
VariantSetInfo	*A variant set definition; classes, types, and values, for a specific variant set definition supported by the target. Support by the target of a particular variant set definition does not imply that the definition is supported for any specific database or element.*	variantSet oid
UnitInfo	*Unit definitions supported by the target.*	unit system name
CategoryList	*Explain categories that the target supports.*	(no key)

3.2.10.2.2 Retrieving Summary and Descriptive Information. The Explain facility provides for the retrieval of summary, or "brief" information. For example the origin may request summary information about all of the databases supported by a target without retrieving the full databaseInfo records. Within each category's definition, elements are designated as "brief" or "non-brief." Elements designated "brief" are obtained when using the element set name 'B'. Elements designated "non-brief" are obtained (along with brief-elements) when using the element set name 'F'.

The Explain facility also provides for the retrieval of descriptive information, for certain categories, via the element set name 'description' (for details, refer to the ASN.1 definition for the Explain syntax). For example, a Database-info record includes an element that contains a description (in human-readable text) of the database; to retrieve only the brief elements and the description element, the element set name 'description' may be used.

Individual categories defined in the Explain syntax may designate other element set names for specific subsets of information within that category.

3.2.10.3 Detailed Descriptions of the Information Categories

This section includes complete descriptions of each information category. In addition to the information enumerated, each record:

•contains information about the record itself, e.g., date of creation and expiration date of the record; and

•includes an element indicating the language of the "human-readable text" elements of the record.

These are logical descriptions that do not reflect that there might be language variants of a record or syntax variants of an element.

Many of the Explain elements are optional, but are not so indicated in the description below. For specific information, refer to the ASN.1 definition.

3.2.10.3.1 Target-Info. Information about the target. There is one such Explain record in the Explain database.

Brief elements:

•A name for the target (only one), in human-readable text
•Recent news of interest to people using this target, in human-readable text
•An icon used to represent this target (in machine-presentable form)
•Whether named results sets are supported
•Whether multiple databases can be searched in one search request
•The maximum number of concurrent result sets supported
•The maximum size (in records) of a result set
•The maximum number of terms allowed in one search request
•A timeout interval after which the target will trigger an event if no activity has occurred
•A "welcome" message from the target to be displayed by the origin.

Non-brief elements:

•Contact information for the organization supporting this target
•A description of the target, in human-readable text
•A set of nicknames or alternate names by which the target is known
•Restrictions pertaining to this target, in human-readable text
•A payment address (e.g., business office) for the organization supporting this target
•Hours of operation
•A list of supported database combinations
•Internet address and Port number
•OSI addresses
•Languages supported for message strings
•The following elements, where each object listed is supported for one or more databases (to determine which are supported for a particular database, retrieve the record for that database):

— Which query-types are supported, and details for each supported type
— Diagnostic sets supported
— Attribute sets supported
— Schemas supported
— Record syntaxes supported
— Resource challenges supported
— Access challenges supported
— Cost information
— Variant sets supported
— element set names supported
— Unit systems supported.

3.2.10.3.2 Database-Info. Detailed description of a database and database-related restrictions and parameters. There is one such Explain record for each database supported.

Brief elements:

•Full database name (only one)
•Whether this is an Explain database (possibly for a different server)
•A list of short (or alternate) names for the database
•An icon used to represent this database (in machine-presentable form)
•Whether there is a charge to access this database
•Whether this database is currently available for access
•A human-readable name or title for the database (as opposed to the database name, which is typically a short string not meant to be human-readable, and not variable by language.)

Non-brief elements:

•A list of keywords for the database
•A description of the database in human-readable text

- Associated databases: those that the target allows (and possibly encourages) to be searched in combination with this database
- Sub-databases that make up this conceptual single database
- Any disclaimers concerning this database in human-readable text
- News about this database in human-readable text
- A record count for the database (and whether the count is accurate or an estimate)
- A description of the default order in which records are presented in human-readable text.
- An estimate of the average record size (in bytes)
- A maximum record size (in bytes)
- Hours of operation that this database is available
- Best time to access this database, in human-readable text
- Time of last update of this database
- Update cycle/interval for this database
- Coverage dates of this database in human-readable text
- Whether this database contains proprietary information
- A description of copyright issues relating to this database in human-readable text
- A notice concerning copyright which the target expects the origin to display to the user if possible, in human-readable text
- Description and contact information for the database producer and database supplier, and how to submit material for inclusion in this database, in human-readable text
- Which query-types are supported for this database, and details for each supported type
- Diagnostic sets supported for this database
- Attribute sets supported for this database
- Schemas defined for this database
- Record syntaxes supported by this database
- Resource reports supported for this database
- Text describing access control for this database in human-readable text
- Costing information related to this database, in both machine-readable format, and in human-readable text, for connect, present, and search
- Variant sets supported for this database
- Element set names supported for this database, with names and descriptions given in human-readable text
- Unit systems supported for this database.

3.2.10.3.3 Schema-Info. Descriptive information about a database schema. There is one Explain record for each schema supported by the target.
Note: this is not specific to a database.
Brief elements:

- The object identifier of the schema definition
- The name of this schema.

Non-brief elements:

- A description of this schema in human-readable text
- TagSets used by this schema, and for each, a designated tagType
- The abstract record structure defined by this schema.

3.2.10.3.4 Tag-Set-Info. Descriptive information about a given tagSet. There is one such Explain record for each supported tagSet.
Brief elements:

- The object identifier for the tagSet
- The name of this tagSet.

Non-brief elements:

- A description of this tagSet, in human-readable text
- For each element defined in the tagSet:
  - — The name of the element
  - — Nicknames for the element
  - — The tag assigned to the element
  - — A description of the element
  - — Its datatype.

3.2.10.3.5 Record-Syntax-Info. Descriptive information about a record syntax. There is one Explain record for each abstract record syntax supported by the target. Note: this is not specific to a database.
Brief elements:
- •The object identifier of the abstract record syntax
- •A name by which this syntax is known.

Non-brief elements:
- •Transfer syntaxes supported for this abstract syntax (object identifiers)
- •A description of this abstract record syntax in human-readable text
- •An ASN.1 module describing the syntax
- •The record structure defined by this syntax.

3.2.10.3.6 Attribute-Set-Info. Descriptive information about an attribute set. There is one record for each supported attribute set.
Brief elements:
- •The attribute set Id (object identifier) for this attribute set
- •Its name.

Non-brief elements:
- • For each attribute type, its name, description, and integer value of the type, and a list of attributes. For each attribute:
  - — Its name
  - — Description
  - — Its value
  - — Names of equivalent attributes. Equivalences are derived from the attribute set definition (not from the targets behavior)
- •Description of the attribute set.

3.2.10.3.7 TermList-Info. Descriptive information about term-lists. There is one Explain record for each database.
Brief elements:
- •Full database name (one only)
- •Summary information about each term-list associated with this database (for each term-list described, there is a TermList-details record):
  - — Name of the term-list. Must be unique for the database. This is the name to be used to search for the TermList-details record for this term list
  - — Its title. For users to see; need not be unique
  - — An indication of how expensive it is to search, using the associated attributes. The target indicates one of the following:
    - – The attribute (combination) associated with this list will do fast searches
      Note: To obtain the attribute combination, retrieve the associated TermList-details record.
    - – The attribute (combination) will work as expected, so there is probably an index for the attribute (combination) or some similar mechanism
    - – Can use the attribute (combination), but it might not provide satisfactory results. Probably there is no index, or post-processing of records is required
    - – cannot search with this attribute (combination) alone
  - — Whether the term-list may be scanned
  - — A list of names of alternative, broader term-lists
  - — A list of names of alternative, narrower term-lists.

(No non-brief elements.)

3.2.10.3.8 Extended-Services-Info. Descriptive information about an extended service. There is one Explain record for each extended service supported.
Brief elements:
- •The object identifier of the extended service
- •A name by which this extended service is known
- •Boolean flags, indicating:
  - — Whether it is a private extended service
  - — Whether restrictions apply
  - — Whether a fee applies
  - — Whether the service is available
  - — Whether retention is supported
- •What level of wait-action is supported.

Non-brief elements:
- •A description in human-readable text
- •Explain elements specific to this extended service (defined within the specific extended service definition)
- •An ASN.1 module for the Explain definition.

3.2.10.3.9 Attribute-Details. Information for each attribute. There is one Explain record for each supported database.
Brief elements:
- •Name of the database to which this attribute information applies.

Non-brief elements:
- •For each attribute set supported for the database, the object identifier of the attribute set, and for each attribute within the set
  - — The attribute type
  - — A default value which applies if the attribute is omitted, and a description of default behavior in human readable form
  - — For each value of the attribute:
    - – The attribute value
    - – A description of that value in human-readable text
    - – Sub-attributes (for Use attributes): a list of alternative values that allow access to the same aspect of the record, but in greater detail
    - – Super-attributes (for Use attributes): a list of alternative values that allow access to the same aspect of the record at a coarser level
    - – Whether the value is only “partially supported”: i.e., the value is accepted but may not provide expected results
- •A list of all attributes combinations supported for the database.

3.2.10.3.10 Term-list-Details. Descriptive information for a term-list. There is one record for each term-list listed by TermList-info records.
Brief elements:
- •Name of the term-list.

Non-brief elements:
- •A description
- •Attribute combination corresponding to this list. If list may be scanned, this is the attribute combination to be used by scan
- •Maximum step-size supported
- •Collating sequence (e.g., ASCII, EBCDIC) in human-readable text
- •Order (ascending or descending)
- •Estimated number of terms
- •A list of sample terms (not guaranteed to be valid; optimally would represent a uniformly distributed sampling of the list).

3.2.10.3.11 Element-Set-Details. Descriptive information about an element set. There is one Explain record for each element set for each record syntax for each database.
Brief elements:
- •The database to which this record pertains
- •The element set name for the element set described by this record
- •The record syntax to which this record pertains
- •The schema for which this element set is defined.

Non-brief elements:
- •A description, in human-readable text, of the element set
- •For each element in the element set, the information provided for each element by the Retrieval-Record-Details category.

3.2.10.3.12 Retrieval-Record-Details. Descriptive information about the elements of a retrieval record. Note that the elements are relative to a database schema. There is one such Explain record for each database for each schema for each record syntax.
Brief elements:
- •The database, schema, and record syntax to which this Explain record pertains.

Non-brief elements (for each element described by the syntax):
- •The name of the element
- •The tag of the element, if any
- •A list of schema elements that comprise this element within the record syntax
- •The maximum size of the element
- •The minimum size of the element

- •The average size of the element
- •The size of the element, if fixed length
- •Whether or not the element is repeatable
- •Whether or not the element is required
- •A description of the element in human-readable text
- •A description of its contents in human-readable text
- •Charging/billing issues related to this element in human-readable text
- •Restrictions (e.g., copyright, proprietary) pertaining to use and access to this element in human-readable text
- •Alternate names for this element
- •Generic names for this element text (e.g., a "geographicSubject" element might also be under the generic name "subject")
- •Attribute combinations corresponding to this element.

3.2.10.3.13 Sort-Details. Description of the sorting capabilities supported by the target. There is one record for each database.
Brief elements:
- •Database to which this sort description pertains.

Non-brief elements:
- •For each sort key:

  — A description
  — If the key is a record element, a specification of the element
  — If the key is an attribute combination, a specification of that combination
  — The type of key: character, numeric, structure
  — Whether the key is case-sensitive.

3.2.10.3.14 Processing-Info. Instructions, representing how the target believes the data should be processed by the origin for presentation to the user. Instructions are defined externally. For a given database and processing context (access, search, retrieval, record-presentation, and record-handling) for which the target offers processing information, there may be more than one set of instructions; these are distinguished by name. Each set of instructions may be available in more than one abstract syntax; these are distinguished by object identifier. Thus an Explain record of this type is distinguished by database, processing context, name, and object identifier.
Brief elements:
- •Full name of the database to which this record pertains
- •The context for which this processing information is pertinent
- •A name for this processing information
- •An object identifier, for the abstract syntax of the externally defined instructions.

Non-brief elements:
- •A description of the instructions, in human-readable form
- •The machine-processable instructions, externally defined (whose abstract syntax is identified by the object identifier referenced above).

3.2.10.3.15 Variant-set-info. Descriptive information about a variant set definition supported by the target; classes, types, and values supported for a particular variant set. Support of a particular variant set definition does not imply that the definition is supported for any specific database or element.
Brief elements:
- •The object identifier of the variant set definition
- •Its name.

Non-brief elements:
- •A list of supported classes, including name and description; and for each, a list of supported types, including name and description; and for each, a list of supported values.

3.2.10.3.16 Unit-info. Descriptive information about a unit system definition supported by the target.
Brief elements:
- •The name of the unit system.

Non-brief elements:
- •A description
- •A list of unit types, including name and description, and for each, a list of units, including name and description.

3.2.10.3.17 Category-list. A list of the Explain categories supported by the target. There is one such record for the Explain database. It consists of the information below, for each supported category.
Brief elements:

- The search term used in conjunction with Use attribute of ExplainCategory to search for records of this category.

Note: the following need occur only if the target is supporting a category not defined in this standard:

- The original search term (This is for information categories where the target is supporting a revision of the original definition of a category.)
- A description
- An ASN.1 definition of the record for this category.

3.2.11 Termination Facility

The Termination Facility consists of the single service, Close.

3.2.11.1 Close Service

The Close service allows either an origin or target to abruptly terminate all active operations and to initiate termination of the Z-association. See Table 15.

The Close service may be used only when version 3 is in force. If so, following initialization, at any time until a Close request is either issued or received, either the origin or target:

- may issue a Close request, consider all active operations to be abruptly terminated, await a Close response (discarding any intervening messages), and consider the Z-association closed; and
- should be prepared to receive a Close request, consider all active operations to be abruptly terminated, issue a Close response, and consider the Z-association closed.

Table 15: Parameters of the Close Service

Parameter	Request	Response	Note
Close-reason	x	x	
Diagnostic-information	x (opt)	x (opt)	
Resource-report-format	x (opt)		Origin only
Resource-report	x (opt)	x (opt)	Target only
Other-information	x (opt)	x (opt)	
Reference-id	x (if appl)	x (if appl)	

3.2.11.1.1 Close-reason. This parameter indicates the reason the origin or target is closing the Z-association. Its values are:

- finished
- shutdown
- system problem
- cost limits
- resources
- security violation
- protocol error
- lack of activity
- unspecified
- response to Close request.

Note: Both the Close request and Close response map to the same protocol message (Close APDU). If both systems issue a Close request at the same time, each will receive the peer message as a Close response (even though the message was not sent as such). This potential ambiguity will not affect the correct operation of the protocol. However, for the case where the message is indeed sent as a Close response, the last of the above listed statuses, “response to Close request,” is provided and may optionally be used.

3.2.11.1.2 Diagnostic-information. The target may include an optional text message, providing additional diagnostic information.

3.2.11.1.3 Resource-report-format and Resource-report. When the origin issues a Close request: the origin may include the parameter resource-report-format to request that the target include a resource report (see 3.2.6.1.1) in the response. The target’s decision to include a resource report in the response (and the format) is unilateral: it may include or omit a report regardless of whether the origin included the parameter resource-report-format.

When the target issues a Close request: the target may unilaterally include a resource report.

3.2.11.1.4 Other-information. This parameter may be used by the origin or target for additional information, not specified by the standard.

3.2.11.1.5 Reference-id. The parameter Reference-id may be included or omitted on a Close request or response from the origin.

The target should omit Reference-id on a Close request. On a Close response, if the target is responding to a Close request that included Reference-id, the target may either include Reference-id using the identical value, or it may omit the parameter. If the target is responding to a Close request that did not include a Reference-id, the target should omit the parameter.

3.3 Message/Record Size and Segmentation

A "segment" is a message that is sent (or is in preparation for transmission) by the target as part of an aggregate Present response, i.e., a Segment request or Present response.

Throughout 3.3, "record" is used as follows:

- •Unless otherwise qualified, it means "response record," i.e., retrieval record or surrogate diagnostic.
- •Except within 3.3.3, it means "surrogate diagnostic record" if the record size exceeds preferred-message-size.
- •"Record N" means "the response record corresponding to the database record identified by result set entry N."
- •A record is considered to be a string of bytes (for the purpose of describing segmentation procedures).
- •"Record size" refers to the size of a record in bytes.

Except within 3.3.3, a set of records is said to "fit in a segment" if the sum of their sizes, not including protocol control information, does not exceed Preferred-message-size. For the Present operation, the target might be unable to fit the requested records in a single segment, because of record or message size limitations. In that case, the target may perform segmentation of the Present response (if segmentation is in effect) by sending multiple segments (Segment requests followed by Present response).

Two levels of segmentation, level 1 and level 2, are subject to negotiation. If neither level is in effect, the target response to a Present request consists of a simple Present response (a single segment) which contains an integral number of records. If level 1 segmentation is in effect, the target response to a Present request may consist of multiple segments (Segment requests followed by a Present response), and each segment must contain an integral number of records, i.e., records may not span segments. If level 2 segmentation is in effect, the target response to a Present request may consist of multiple segments, and records may span segments.

3.3.1 Procedures When No Segmentation Is in Effect

The procedures in this section (3.3.1) apply when no segmentation is in effect. (They apply not only to a Present operation when no segmentation is in effect, but they also apply in general to a Search operation, whether or not segmentation is in effect; a Search response is not subject to segmentation.)

The target responds to a Present request with a simple Present response (or to a Search request with a Search response), which contains an integral number of records. If the target is not able to return all of the records requested because of message size limitations, the target should fit as many records as possible.

Procedure

Assume that the target is attempting to return records M through N. If records M through N fit in the response, then the target returns those records. Otherwise, the target returns records M through P, where P is chosen so that records M through P fit in the response, but records M through P+1 do not.

Illustration

Assume that the target is attempting to return records 1 through 10; records 1 through 6 fit in the response, but retrieval records 1 through 7 will not fit.

The size of retrieval record 7 itself:

(a) does not exceed Preferred-message-size, or

(b) exceeds Preferred-message-size, but does not exceed Exceptional-record-size, or

(c) exceeds Exceptional-record-size.

In case (a), the target returns records 1 through 6. In case (b), except as noted below (see "Exception"), the target substitutes a diagnostic record for retrieval record 7, indicating that the record exceeds Preferred-message-size. In case (c) the target substitutes a diagnostic record for retrieval record 7, indicating that the record exceeds Exceptional-record-size. (If Exceptional-record-size equals Preferred-message-size then there is no distinction between the meaning of the two diagnostics.)

In case (b) or (c):

- •If the diagnostic record will not fit along with records 1 through 6, the target returns records 1 through 6. (Preferred-message-size must always be large enough to contain any diagnostic record; thus a subsequent present request beginning with record 7 will retrieve the diagnostic.)
- •Otherwise, the target inserts the diagnostic record and proceeds to attempt to fit records 8 through 10.

Exception
If a Present request specifies a single record (i.e., Number-of-records-requested equals 1) then if the size of that retrieval record exceeds Preferred-message-size, but does not exceed Exceptional-record-size, the target will return that single retrieval record. Note that this exception applies only to a Present operation and not to a Search operation.

Thus in case (b), the origin may subsequently retrieve retrieval record 7, by issuing a Present request in which that record is the only record requested.

Note that the purpose of this distinction between Preferred-message-size and Exceptional-record-size is to allow the transfer of normal length records to proceed in a routine fashion with convenient buffer sizes, while also providing for the transfer of an occasional exceptionally large retrieval record without requiring the origin to continually allocate and hold local buffer space for worst-case records. Note also that this intended purpose is defeated if the origin routinely requests a single record.

3.3.2 Level 1 Segmentation

When level 1 segmentation is in effect, the target may segment the aggregate Present response into multiple segments (zero or more Segment requests followed by a Present response), each consisting of integral records (i.e., records may not span segments). The procedures described in this section (3.3.2) apply if level 1 segmentation is in effect.

Beginning with the first record requested and continuing with adjacent higher number records, the target forms segments to contain the requested records. Each segment is sent as a Segment request, except the last, which is sent as a Present response.

The number of segments must not exceed the value of the (optional) Present request parameter Max-segment-count, if supplied.

If Max-segment-count is supplied, and its value is 1, then the procedures of 3.3.1 apply. Also, the same exception as cited in 3.3.1 applies if a Present request has requested a single record.

Procedure
Assume that the origin requests result set records M through N.

Case A: M<N (i.e., more than one record requested).

1. Set P=M.
2. If records P through N fit in a segment:
    - Fit records P through N in the segment.
    - Go to step 3.

   Otherwise,
    - Fit records P through Q, where Q (which is less than N) is such that records P through Q fit in a segment, but records P through Q+1 do not.
    - If Max-segment-count is reached, go to step 3.
    - Send the segment as a Segment request.
    - Set P=Q+1.
    - Repeat step 2.
3. Send the segment as a Present response.

Case B: M=N (i.e., a single record requested).
The target sends a simple Present response (a single segment). The size of the segment may exceed Preferred-message-size. The segment contains the single requested retrieval record, or a surrogate diagnostic record if the size of the record exceeds Exceptional-record-size.

Illustration
Assume the origin has requested records 1 through 10.

1. If all ten records fit in a segment, the aggregate Present response consists of a Present response including the requested records. Present-status is ‘success’ (all expected response records available).
2. Suppose records 1 through 4 fit in a segment, but records 1 through 5 do not; records 5 through 9 fit in a segment but records 5 through 10 do not. (Assume the Present request has specified a value of 3 or greater for the parameter Max-segment-count.) Then the aggregate Present response consists of:

    - a segment request including records 1 through 4,
    - a segment request including records 5 through 9, and
    - a Present response including record 10.

   Present-status is ‘success’ (all expected response records available).

   Note that the target is expected to pack as many records into a segment as will fit; thus for example, the first segment would not consist of records 1 through 3, because records 1 through 4 will fit.

3. Assume the conditions in (2) are true, except that the Present request has specified a value of 2 for the parameter Max-segment-count. Then the aggregate Present response consists of:

•a Segment request including records 1 through 4, and
•a Present response including records 5 through 9.

Present-status is 'partial-2' (not all expected response records available, because they will not all fit within the preferred message size).

3.3.3 Level 2 Segmentation

When level 2 segmentation is in effect, the target may segment the aggregate Present response into multiple segments (as is the case for level 1 segmentation) and in addition, records may span segments. The procedures described in this section (3.3.3) app ly if level 2 segmentation is in effect.

If a retrieval record will not fit in a segment (along with records already packed into the segment) it may be segmented into multiple contiguous fragments (see 3.3.3.1) to be packed into consecutive segments according to the procedures detailed in 3.3.3.2 and 3.3.3.3.

3.3.3.1 Fragments

A *fragment* is a proper substring of a record (as noted above, within section 3.3.3 a record is treated as a string of bytes). A particular instance of segmentation of a record results in a sequence of two or more fragments whose concatenation (not including protocol control information) is identical to the record. However, there may be different instances of segmentation of a particular record, and the origin cannot necessarily predict how a record will be segmented into fragments by the target in a particular instance.

For the purpose of procedure description (3.3.3.3) a *starting fragment* is defined to be a fragment that starts at the beginning of a record. An *intermediate fragment* is a fragment that neither starts at the beginning nor ends at the end of a record. A *final fragment* is a fragment that ends at the end of a record. An integral record (not segmented) is not a fragment.

The sum of the sizes of the records and record fragments in a segment, not including protocol control information, must not exceed Max-segment-size (see 3.3.3.2).

3.3.3.2 Segment Size, Record Size, and Segment Count

If level 2 segmentation is in effect, the Present request may optionally include these three parameters:

Max-segment-size — The largest allowable segment. If included, overrides Preferred-message-size (for this Present operation only). If not included, Max-segment-size assumes the value Preferred-message-size.

Max-record-size — The largest allowable retrieval record within the aggregate Present response. If included, it must equal or exceed Max-segment-size. (If level 2 segmentation is in effect, the parameter Exceptional-record-size that was negotiated during initialization does not apply, whether or not Max-record-size is included, unless the value of Max-segment-count is 1.)

Max-segment-count — The maximum number of segments the target may include in the aggregate Present response. If its value is 1, no segmentation is applied for the operation, the procedures of section 3.3.1 apply, and Max-record-size should not be included.

If the latter two parameters are both included, Max-record-size must not exceed the product Max-segment-size times Max-segment-count.

If Max-record-size but not Max-segment-count is included, the origin should be prepared to receive as many segments as necessary to retrieve the requested records.

If Max-segment-count is included (and its value is greater than 1), but Max-record-size is not, the product Max-segment-size times Max-segment-count is the maximum record size for the operation.

If the latter two parameters are both omitted the origin should be prepared to receive arbitrarily large records and an arbitrary number of segments.

3.3.3.3 Segmentation Procedures

The following procedures apply for level 2 segmentation. The target fits as many integral records as possible into the first segment. If all of the requested records will fit, the segment is sent as a simple Present response. Otherwise, in the space remaining within that segment the target fits a starting fragment of the following record (if possible), and the segment is sent as a Segment request. The target then fits the remainder of that record into the next segment if possible; if not possible, the target sends Segment requests as necessary with intermediate fragments, and fits the final fragment, if any, into the beginning of the next segment and fills as many integral records as possible within the space remaining within that segment. If the last of the requested records is placed in the segment (or Max-segment-count is reached) the segment is sent as a Present response. Otherwise the target continues to fill segments in this manner until the last of the requested records is placed in a segment or Max-segment-count is reached, and sends each segment as a Segment request except the last, which it sends as a Present response. These procedures are reiterated more formally as follows:

Procedure
Assume that the origin requests records M through N. (Note that "Record" means "surrogate diagnostic record" if the size of the record exceeds Max-record-size, or if the target is unable to segment the record so that each fragment fits within a segment.)

1. Set R=M (begin preparation of first segment).
2. If record R fits in the current segment:

   •Fit integral records R through P, where P is the largest number (not exceeding N) so that records R through P fit.
   •If P equals N, or if Max-segment-count is reached, go to step 8 below.
   •R=P+1

3. Note that having reached this step, record R will not fit in the current segment. If the Present request has included Max-segment-count and the target is unable to determine whether record R will fit in the remainder of the aggregate response:

   •Insert a surrogate diagnostic record, which in effect suggests that the origin might again attempt to retrieve the record, but without specifying a Max-segment-count.
   •Go to step 7.

4. If record R will not fit in the remainder of the aggregate response, go to step 8.
5. If record R will fit in the remainder of the aggregate response, but no starting fragment will fit in the current segment:

   •Note that this condition precludes the possibility that the segment is empty; see note preceding step 1.
   •Transmit a Segment request (begin preparation of the next segment).
   •go to step 2.

6. Note that having reached this step, Record R will fit in the remaining segments; it will not fit within the current segment, but a starting fragment will fit in the current segment.

   •Fit the largest possible starting fragment of record R and transmit a Segment request.
   •Fill as many complete segments as necessary (which may be zero) with intermediate fragments of record R and send Segment requests.
   •Begin preparation of the next segment, first inserting the final fragment of record R.

7. Set R = R+1.
   •If R is less than or equal to N, go to step 2.

8. Send a Present response.

Illustration
Assume the origin has requested records 1 through 12. All records are 500 bytes, except record 5, which is 10,000 bytes. Max-segment-size is 3200.

1. Suppose record 5 consists of 10 elements, each 1000 bytes. The target is able to segment record 5, but only at element boundaries; the target will not let the elements span fragments.
   Note: this means that the target may segment the record so that a fragment consists of bytes M*1000+1 through (M+N)*1000, M= 0,1, ...9; N = 1, 2, ..., 10-M; e.g., bytes 1-1000, 1-2000, 1-3000, 1000-2000, 1000-3000, 1000-4000, etc.
   Suppose further that the target cannot segment any other records. The aggregate Present response is as follows:
   segment 1: Segment request consisting of records 1 through 4, and the first 1000 bytes of record 5 as a starting fragment.
   Note: the size of the segment is 3000 bytes which is less than the Max-segment-size of 3200, but the target cannot fit another fragment in the segment because that would cause the segment size to exceed the 3200-byte maximum (the minimum fragment size is 1000 bytes).
   segment 2: Segment request consisting of bytes 1001 through 4000 of record 5 as an intermediate fragment.
   segment 3: Segment request consisting of bytes 4001 through 7000 of record 5 as an intermediate fragment.
   segment 4: Segment request consisting of bytes 7001 through 10,000 of record 5 as a final fragment.
   segment 5: Segment request consisting of records 6 through 11.
   segment 6: Present response consisting of record 12.

2. Suppose further that the target can segment the smaller records into 100 byte fragments (or multiples).
   segments 1 through 3 are as in illustration 1.
   segment 4: Segment request consisting of bytes 7001 through 10,000 of record 5 as a final fragment, and bytes 1 through 200 of record 6 as a starting fragment.
   segment 5: Segment request consisting of bytes 201 through 500 of record 6 as a final fragment, records 7 through 11, and the first 400 bytes of record 12 as a starting fragment.
   segment 6: Present response consisting of bytes 401 through 500 of record 12 as a final fragment.
3. Suppose the target can segment any of the records at arbitrary byte boundaries.
   segment 1: Segment request consisting of records 1 through 4 and the first 1200 bytes of record 5 as a starting fragment.
   segment 2: Segment request consisting of bytes 1201 through 4200 of record 5 as an intermediate fragment.
   segment 3: Segment request consisting of bytes 4201 through 7400 of record 5 as an intermediate fragment.
   segment 4: Segment request consisting of bytes 7401 through 10,000 of record 5 as a final fragment, record 6, and the first 100 bytes of record 7 as a starting fragment.
   segment 5: Present response consisting of bytes 101 through 500 of record 7 as a final fragment, and records 8 through 12.

3.4 Operations and Reference-id

A request from the origin of a particular operation type initiates an *operation*, which is terminated by the respective response from the target. The following operation types are defined: Init, Search, Present, Delete, Resource-report, Sort, Scan, and Extended-services. (Thus each origin request type corresponds to an operation type with the exception of the following request types: Trigger-resource-control and Close.) An operation consists of the initiating request and the terminating response, along with any intervening Access-control and Resource-control requests and responses, Trigger-resource-control requests, and Segment requests. An operation is assigned a Reference-id by the origin, the origin includes the Reference-id within the initiating request, and it must be included within each message of the operation. If 'serial operations' is in effect, the Reference-id parameter may be omitted in the initiating request; in that case the reference-id is considered null for that operation, and all other messages of that operation must also omit the Reference-id parameter.

Any message sent from origin to target or vice versa (i.e., any request or response defined by this service definition) is part of an operation (identified by its Reference-id) with these exceptions:

- A Close request or response is not part of any operation.
  Note: A Close request or response may include a reference-id, according to the procedures specified in 3.2.11.1.5.
- If 'concurrent operations' is in effect any Resource-control or Access-control request or response which does not include a Reference-id is not part of an operation.

This standard does not assume any relationship between a given operation and any subsequent operation even if the latter operation uses the same reference-id. This standard does not specify the contents of the Reference-id parameter, nor its meaning, except to the extent that it is used to refer to an operation. Reference-ids are always assigned by the origin and have meaning only within the origin system. Since no semantics are attributed to the Reference-id, it has no implied data type and can only be described as transparent binary data. (Its ASN.1 type is therefore OCTET STRING.)

3.5 Concurrent Operations

If 'concurrent operations' is in effect, the Reference-id parameter is mandatory in an initiating request (however, see note), and the origin may initiate multiple concurrent operations, each identified by a different reference-id.
Note: The Reference-id parameter is always optional in an Init request; 'concurrent operations' does not take effect until negotiation is complete, and is thus not in effect during an Init operation.

Once an operation is initiated, until that operation is terminated, another operation may not be initiated with the same reference-id. This standard does not specify the order in which concurrent operations are processed at the target; the target may process concurrent operations in any manner it chooses.

Example:
The origin may issue a Search request using Reference-id "100," and then issue a second Search request using Reference-id "101" before receiving the Search response from the first Search request. There would then be two concurrent operations. Receipt by the origin of the response corresponding to the second Search request (identified by Reference-id "101") would terminate the second operation, and that might occur before termination of the first operation (identified by Reference-id "100"). The origin might then issue a Present request (against the result set created by the second operation), initiating another operation. In that case, the origin must supply a Reference-id other than "100" (because there is an active operation with that Reference-id). The new Reference-id could (but need not) be "101"; if it is, the target may not assume any implied relationship between this new operation and the previous operation which used Reference-id "101."

No operation may be initiated while an Init operation is in progress. No operation may be initiated within a Z-association after a Close request has been sent or received.

All result sets are, in principle, available to any operation. It is possible that two or more concurrent operations will attempt to reference the same result set. This standard does not specify what happens in that circumstance. The origin should not initiate concurrent Search operations with the same value of Result-set-id.

Other than the restriction cited above (that when the origin uses a Reference-id to initiate an operation, until that operation is terminated it may not use that Reference-id to initiate another operation) there are no restrictions on the reuse or management of Reference-ids by the origin. The origin might recycle Reference-ids randomly among users, or it may manage local threads by assigning different Reference-ids to end-users. The target is not required to know how the origin manages Reference-ids, or in particular, that the origin is using Reference-ids to distinguish different users. The target is not required to have any knowledge of multiple end users at the origin, the target interacts only with the (single) origin.

3.6 Composition Specification

For each database supported the target defines one or more schemas (see 3.1.5), and designates one as the default schema. For each schema, the target designates one or more element specification identifiers.

An *element specification identifier* is the object identifier of an *element specification format* (a structure used to express an element specification) or an *element set name*. The latter is a primitive name. An *element specification* is an instance of an element specification format, or an element set name.

For the default schema, at least one of the element specification identifiers must be an element set name, and the target designates one as the default element set name for the database.

Note: the target designates this information either via the Explain facility, or through some mechanism outside of the standard.

For each record to be returned in a Search or aggregate Present response, the target applies an abstract record structure (defined by a schema for the database to which that record belongs) to form an abstract database record. The target applies an element specification to the abstract database record to form another instance of the abstract database record (the latter might be a null transformation), to which the target applies a record syntax to form a retrieval record.

If the origin includes the parameter Comp-spec (in a Present request) the procedures of 3.6.1 apply. For a Search operation, or a Present operation when the parameter Comp-spec is omitted, the default schema is assumed for each record, and the procedures of 3.6.2 apply.

3.6.1 Comp-spec Specified

The Present request parameter Comp-spec includes a set of one or more pairs of a database name and associated composition specification. Each composition specification may include a schema identifier (or if not, the default schema for the database is assumed) and an element specification. For each record to be returned in the aggregate Present response:

- If the database to which the record belongs is specified (as a component of one of the pairs) then the target forms an abstract database record by applying the corresponding composition specification (i.e., by first applying the abstract record structure, defined by the schema, to the database record to form an abstract database record, and then applying the element specification where the schema and element specification are from the composition specification), if it is able to do so.
- Otherwise, the target forms an abstract database record by applying the abstract record structure defined by the default schema and default element set name for the database to which the record belongs.

The parameter Comp-spec may alternatively consist of a single composition specification with no database specified. In that case, for each record to be returned, if the target is able to form an abstract database record according to that composition specification, it does so. If not, an abstract database record is composed according to the default schema and default element set name for the database to which the record belongs.

The target applies a record syntax (which may be included in the composition specification or within the parameter Preferred-record-syntax) to the resulting abstract database record to form a retrieval record.

3.6.2 Comp-spec Omitted

When requesting the retrieval of a set of records from a result set, if the parameter Comp-spec is omitted, the procedures of this section apply.

Notes:

1. This is always the case on a Search request because the parameter Comp-spec is not included in the definition of the Search request.
2. This is always the case when version 2 is in force because the parameter Comp-spec is not defined in version 2.

The Search request parameters Small-set-element-set-names and Medium-set-element-set-names, and the Present request parameter Element-set-names, take the form of a set of one or more pairs of a database name and associated element set name. For each record to be returned in the Search or aggregate Present response, the target first applies the abstract record structure defined by the default schema for the database to which the record belongs, to form an abstract database record, and then applies an element set name, as follows:

- If the database to which the record belongs is specified (as a component of one of the pairs), and if the corresponding element set name is valid for the default schema for the database, then the target applies that element set name.
- If not, the target applies the default element set name for the database.

Each of these parameters may alternatively consist of a single element set name with no database specified. In that case, for each record to be returned, if the element set name is valid for the default schema for the database to which the record belongs, the target applies that element set name; if not, the target applies the default element set name for the database.

A target must always recognize the character string "F" as an element set name to mean "full"; when it is applied to an abstract database record, it results in the same abstract database record (i.e., a null transformation).

A target must always recognize the character string "B" as an element set name to mean "brief" record. This standard does not define the meaning of "brief." Unless the origin knows the target's definition of "brief" for a given schema, it should not assume that any particular elements are included.

The origin may specify a "preferred-record-syntax," which the target applies (to the abstract database record formed by the application of the element set name) to form a retrieval record. If the origin does not specify a preferred-record-syntax the target may select one (see 3.2.2.1.5).

3.6.3 Record Syntax

For each record to be returned in a Search or aggregate Present response, the element set name, or the schema and element specification from the composition specification, results in an abstract database record, as described above. To that abstract database record, the target applies a record syntax, indicated as described above. The term *record syntax* has the following meaning:

- When specified by the origin (either as the value of Preferred-record-syntax or within a composition specification), it takes the form of an OID and refers to an abstract syntax (paired, or to be paired by the target, with a transfer syntax) that the origin requests the target use for retrieval records.
- When specified by the target, it takes the form of an OID or p-context accompanying a retrieval record in a Search or Present response, and it refers to an abstract syntax paired with a transfer syntax.

3.7 Type-1 and Type-101 Queries

This section specifies procedures when Query-type is 1 (or 101; see Note 2 below). Type-1 is the "Reverse Polish Notation" (RPN) query. It has the following structure:

RPN-Query ::= Argument | Argument + Argument + Operator

```
Argument      ::=        Operand | RPN-Query
operand       ::=        AttributeList + Term | ResultSetId | Restriction
Restriction ::=  ResultSetId + AttributeList
operator    ::=  AND | OR | AND-NOT | Prox
```

The notation above is used as follows:

::= means "is defined as"

| means "or"

+ means "followed by," and + has precedence over | (i.e., + is evaluated before |).

Notes:

1. For type-1, the Prox operator and the Restriction operand are defined for version 3 only. When version 2 is in effect, it is a protocol error to include either the Prox operator or Restriction operand in a type-1 query.
2. The type-101 query is defined as identical to the type-1 query, with the exception that the Prox operator and Restriction operand are defined not only for version 3, but for version 2 as well. Thus the definition of the type-101 query is independent of version.

A Z39.50-conforming target must support the type-1 query, but support of the type-1 query does not imply support of any of the defined operators or operands.

The target designates what query types it supports, and which operators and operands.

Note: The target designates this information either through the Explain facility or through some mechanism outside of the standard.

If the target claims support for the Prox operator, the target should also designate whether it supports the extended result set model for proximity (the extended result set model for searching as described in 3.1.6 and its specialization for proximity as described in 3.7.2.2). If the target claims support for the Restriction operand, then it must also support the extended result set model for restriction (the extended result set model for searching and its specialization for restriction as described in 3.7.3).

Note: Only in certain circumstances (detailed below) does support of the Prox operator require support of the extended result set model for proximity. However, support of the Restriction operand always requires support of the extended result set model for Restriction.

3.7.1 Representation and Evaluation of the Type-1 and Type-101 Queries

At the origin, the query is represented by a tree. Each subtree represents an operand, either a simple operand or a complex operand. Each leaf node represents a simple operand: Result-set-id, AttributeList+ Term, or Restriction. Each non-leaf node represents a complex operand: a subtree whose root is an operator, and which contains two subtrees, a left operand and a right operand.

The origin traverses the tree according to a left post-order traversal, to produce a sequence of (simple) operands and operators which is transmitted to the target.

At the target, evaluation of the sequence of operands and operators is illustrated by the use of a stack. Whenever an operand is encountered, it is put on the stack. Whenever an operator is encountered, the last two objects that have been put on the stack are pulled off and the operator is applied as follows:

Each operand represents a set of database records, and each is one of the following:

(a) AttributeList+term — the set of database records obtained by evaluating the specified attribute-set and term against the collection of databases specified in the Search request.

(b) ResultSetId — the set of database records represented by the transient result set identified by ResultSetId.

(c) Restriction operand (ResultSetId+AttributeList): the set of database records represented by the result set identified by ResultSetId, restricted by the specified attribute set (see 3.7.3).

Note: if the Restriction operand occurs the target must support the extended result set model for restriction; otherwise the query is in error.

(d) An intermediate result set (resulting from a previous evaluation placed on the stack) — representing the records identified by that result set.

Illustration

Let S1 and S2 be the sets represented by the left and right operand respectively. Let S be defined as follows:

•If the operator is AND, S is the intersection of S1 and S2.

•If the operator is OR, S is the union of S1 and S2.

•If the operator is AND-NOT, S is the set of elements in S1 which are not in S2.

•If the operator is Prox:

—If both operands are of form (a) S is the subset of records in the set (S1 AND S2) for which A ProxTest B is true (see 3.7.2.1) where A and B are the two operands.

—Otherwise:

—The target must support the extended result set model for proximity; or else the query is in error.

—let R1 and R2 be result sets representing the sets S1 and S2 (i.e., each is either:

– the result set specified by the corresponding operand, if it was of form (b),

or

– the hypothetical result set representing the set of records represented by that operand, otherwise.

In either case, both R1 and R2 are assumed to conform to the extended result set model for proximity.)

Each entry in R1 and R2 contain positional information, in the form of position vectors. For each record represented by both R1 and R2, consider every ordered pair consisting of a position vector associated with the record as represented in R1 and a position vector associated with the record as represented in R2. For each pair that qualifies according to the ProxTest:

– the record is qualified into the set S;

and

– a position vector is created for that record as represented in the resultant set, composed from that ordered pair.

An intermediate result set is created which represents the records in the set S, and is put on the stack. When evaluation of the query is complete (i.e., all query-terms have been processed) one object will remain on the stack (otherwise the query is in error), representing a set of database records, which is the result of the query.

3.7.2 Proximity

3.7.2.1 The Proximity Test

The proximity test, ProxTest, includes a Distance, Relation, Unit, and two Boolean flags: Ordered and Exclusion.

•Distance: Difference between the ordinal positional values of the two operands (e.g., if unit is ‘paragraph’, distance of zero means “same paragraph”). Distance is never negative.

•Relation: LessThan, LessThanOrEqual, Equal, GreaterThanOrEqual, GreaterThan, or NotEqual.

•Unit: Character, Word, Sentence, Paragraph, Section, Chapter, Document, Element, Subelement, ElementType, Byte, or a privately defined unit.

- Ordered flag: if set, the test is for "right" proximity only (the left ordinal must not exceed the right ordinal and Distance is compared with the difference between the right and left ordinals); otherwise, the test is for "right" or "left" proximity (Distance is compared with the absolute value of the difference between the left and right ordinals).
- Exclusion flag: if set, "not" is to be applied to the operation (for example if the test with Exclusion flag 'off' is "'cat' within 5 words of 'hat'," then the same test with Exclusion flag 'on' is "'cat' not within 5 words of 'hat'").

Example
Suppose A and B respectively specify "personal name = 'McGraw, J.' " and "personal name= 'Stengel, C.' ," and:

- Distance is 0
- Relation is 'equal'
- Proximity-unit is 'paragraph'
- Ordered flag is 'false'
- Exclusion flag is 'false'.

Then the result is the set of records in which both of the personal names occur within the same paragraph. Using the same example, if the Exclusion flag is set to 'true', the result is the set of records in which the two personal names never both occur within the same paragraph.

If the Ordered flag is set to 'true' (and Exclusion flag to 'false') then the result is the set of records in which the personal name 'McGraw, J.' occurs within the same paragraph as, but before, the personal name 'Stengel, C.'

If distance is instead 1 ('ordered' and 'exclusion' flag 'false') the result is the set of records in which the two personal names occur in adjacent paragraphs. If, in addition, Relation-type is 'less-than-or-equal' the result is the set of records in which the two names occur within the same or adjacent paragraphs.

3.7.2.2 Extended Result Set Model for Proximity
In the extended result set model for proximity, the target maintains positional information in the form of one or more position vectors associated with each record represented by the result set, which 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 here informally to refer to a proximity test).

The manner in which the target maintains this information is not prescribed by this standard. Annex 13, ERS (non-normative) provides examples.

3.7.3 Restriction and the Extended Result Set Model
The Restriction operand specifies a result-set-id and a set of attributes, and it represents the set of database records identified by the specified result set, restricted by the specified attributes.

Example
Let R be the result set produced by a search on the term 'cat', representing three records:

1 where 'cat' occurs in the title,
2 where 'cat' occurs in the title and as an author, and
3 where 'cat' occurs in the title, as an author, and as a 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 associated with each record represented by the result set which may be used in the evaluation of a restriction operand as a surrogate for the search that created the result set. The manner in which the target maintains this information is not prescribed by this standard. Annex 13, ERS (non-normative) provides examples.

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. Sections 4.1 and 4.2 respectively describe the formats and rules for exchange of Z39.50 *application protocol data unit*s (APDUs). An APDU is a unit of information, transferred between origin and target, whose format is specified by the Z39.50 protocol, consisting of application-protocol-information and possibly application-user-data. 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 are part of the standard.

Z39-50-APDU-1995 -- OID for this definition, assigned in OID.3.1, is {Z39-50 2 1}

```
DEFINITIONS ::=
BEGIN   -- Z39.50 Maintenance Agency Official Text for ISO 23950.
--
EXPORTS OtherInformation, Term, AttributeSetId, AttributeList, AttributeElement, ElementSetName, SortElement,
DatabaseName, CompSpec, Specification, Permissions, InternationalString, IntUnit, Unit, StringOrNumeric, Query, Records,
ResultSetId, DefaultDiagFormat, DiagRec;
--
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}

-- Initialize APDUs
--

 InitializeRequest ::= SEQUENCE{
      referenceId                        ReferenceId OPTIONAL,
      protocolVersion                    ProtocolVersion,
      options                            Options,
      preferredMessageSize     [5]       IMPLICIT INTEGER,
      exceptionalRecordSize    [6]       IMPLICIT INTEGER,
      idAuthentication         [7]       ANY OPTIONAL, -- see note below
      implementationId         [110]     IMPLICIT InternationalString OPTIONAL,
      implementationName       [111]     IMPLICIT InternationalString OPTIONAL,
      implementationVersion    [112]     IMPLICIT InternationalString OPTIONAL,
      userInformationField     [11]      EXTERNAL 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 InternationalString OPTIONAL,
--                      userId       [1]    IMPLICIT InternationalString OPTIONAL,
--                      password     [2]    IMPLICIT InternationalString OPTIONAL },
--     anonymous        NULL,
--     other            EXTERNAL
-- May use access control formats for 'other'.  See Annex 7 ACC.
--
 InitializeResponse ::= SEQUENCE{
 referenceId                          ReferenceId OPTIONAL,
 protocolVersion                      ProtocolVersion,
 options                                    Options,
 preferredMessageSize        [5]      IMPLICIT INTEGER,
 exceptionalRecordSize       [6]      IMPLICIT INTEGER,
 result                               [12]    IMPLICIT BOOLEAN,    -- reject = FALSE; Accept = TRUE
 implementationId            [110]    IMPLICIT InternationalString OPTIONAL,
 implementationName          [111]    IMPLICIT InternationalString OPTIONAL,
 implementationVersion       [112]    IMPLICIT InternationalString OPTIONAL,
 userInformationField        [11]     EXTERNAL OPTIONAL,
 otherInfo                            OtherInformation OPTIONAL}
-- Begin auxiliary definitions for Init PDUs
 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 request 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 10163-1:1993
-- 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)}
-- end auxiliary definitions for Init PDUs

--Search APDUs
 SearchRequest ::= SEQUENCE{
 referenceId                                ReferenceId OPTIONAL,
 smallSetUpperBound             [13]        IMPLICIT INTEGER,
 largeSetLowerBound             [14]        IMPLICIT INTEGER,
 mediumSetPresentNumber         [15]        IMPLICIT INTEGER,
 replaceIndicator               [16]        IMPLICIT BOOLEAN,
 resultSetName                  [17]        IMPLICIT InternationalString,
 databaseNames                  [18]        IMPLICIT SEQUENCE OF DatabaseName,
 smallSetElementSetNames        [100]       ElementSetNames OPTIONAL,
  mediumSetElementSetNames          [101]   ElementSetNames OPTIONAL,
  preferredRecordSyntax             [104]   IMPLICIT OBJECT IDENTIFIER OPTIONAL,
  query                             [21]    Query,
-- Following two parameters may be used only if version 3 is in force.
  additionalSearchInfo              [203]   IMPLICIT OtherInformation OPTIONAL,
  otherInfo                         OtherInformation OPTIONAL}

-- Query Definitions
  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,
                     type-102           [102]   OCTET STRING}
--
-- Definitions for RPN query
               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 force:
                                    --   - 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 force
               numeric                  [215]   IMPLICIT INTEGER,
               characterString          [216]   IMPLICIT InternationalString,
               oid                      [217]   IMPLICIT OBJECT IDENTIFIER,
               dateTime                 [218]   IMPLICIT GeneralizedTime,
               external                 [219]   IMPLICIT EXTERNAL,
               integerAndUnit           [220] IMPLICIT IntUnit,
               null                     [221] IMPLICIT NULL}

  Operator ::= [46] CHOICE{
                              and         [0] IMPLICIT NULL,
                              or          [1] IMPLICIT NULL,
                              and-not     [2] IMPLICIT NULL,
                                        -- If version 2 is in force:
                                        --  - 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 force.
                                                   -- If included, overrides value of attributeSet
                                                   -- in RPNQuery above, but only for this attribute.
          attributeType  [120]  IMPLICIT INTEGER,
          attributeValue        CHOICE{
                                numeric       [121]  IMPLICIT INTEGER,
                                                   -- If version 2 is in force,
                                                   -- Must select 'numeric' for attributeValue.

                                complex [224] IMPLICIT SEQUENCE{
                                list       [1] IMPLICIT SEQUENCE OF StringOrNumeric,
                                semanticAction       [2] IMPLICIT SEQUENCE OF INTEGER OPTIONAL}}}

 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
                                }
-- End definitions for RPN Query

 SearchResponse ::= SEQUENCE{
 referenceId                        ReferenceId OPTIONAL,
 resultCount                 [23]   IMPLICIT INTEGER,
 numberOfRecordsReturned [24]       IMPLICIT INTEGER,
 nextResultSetPosition       [25]   IMPLICIT INTEGER,
 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 force.
 additionalSearchInfo       [203] IMPLICIT OtherInformation OPTIONAL,
 otherInfo                        OtherInformation OPTIONAL}
--Retrieval APDUs
 PresentRequest ::= SEQUENCE{
 referenceId                             ReferenceId OPTIONAL,
 resultSetId                             ResultSetId,
 resultSetStartPoint              [30]   IMPLICIT INTEGER,
 numberOfRecordsRequested         [29]   IMPLICIT INTEGER,
 additionalRanges                 [212]  IMPLICIT SEQUENCE OF Range OPTIONAL,
-- additionalRanges may be included only if version 3 is in force.
 recordComposition                CHOICE{
 simple                           [19]   ElementSetNames,
                                  -- must choose 'simple' if version 2 is in force
 complex                          [209]  IMPLICIT CompSpec} OPTIONAL,
 preferredRecordSyntax            [104]  IMPLICIT OBJECT IDENTIFIER 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 ::= SEQUENCE{
                     -- Segment PDU may only be used when version 3 is in force,
                     -- and only when segmentation is in effect.
 referenceId                             ReferenceId OPTIONAL,
 numberOfRecordsReturned          [24]   IMPLICIT INTEGER,
 segmentRecords                   [0]    IMPLICIT SEQUENCE OF NamePlusRecord,
 otherInfo                               OtherInformation OPTIONAL}
--
 PresentResponse ::= SEQUENCE{
 referenceId                             ReferenceId OPTIONAL,
 numberOfRecordsReturned          [24]   IMPLICIT INTEGER,
 nextResultSetPosition            [25]   IMPLICIT INTEGER,
 presentStatus                           PresentStatus,
 records                                 Records OPTIONAL,
 otherInfo                               OtherInformation OPTIONAL}
-- begin auxiliary definitions for Search and Present 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,
                                         -- Must select one of the above two, retrievalRecord or
                                         -- surrogateDiagnostic, unless 'level 2 segmentation' is in effect.
                                  startingFragment          [3] FragmentSyntax,
                                  intermediateFragment      [4] FragmentSyntax,
                                  finalFragment             [5] FragmentSyntax}}
 FragmentSyntax ::= CHOICE{
            externallyTagged             EXTERNAL,
            notExternallyTagged          OCTET STRING}
 DiagRec ::= CHOICE{
                       defaultFormat          DefaultDiagFormat,
                                                     -- Must choose defaultFormat if version 2 is in effect.
                       externallyDefined      EXTERNAL}

 DefaultDiagFormat::= SEQUENCE{
 diagnosticSetId       OBJECT IDENTIFIER,
 condition             INTEGER,
 addinfo               CHOICE{
                                   v2Addinfo    VisibleString, -- version 2
                                   v3Addinfo    InternationalString    -- version 3
                                    }}
 -- end definition of records
 Range  ::= SEQUENCE{
             startingPosition            [1] IMPLICIT INTEGER,
             numberOfRecords             [2] IMPLICIT INTEGER}
--
 ElementSetNames ::= CHOICE {
             genericElementSetName       [0] IMPLICIT InternationalString,
             databaseSpecific            [1] IMPLICIT SEQUENCE OF SEQUENCE{
                                                 dbName        DatabaseName,
                                                 esn           ElementSetName}}

 PresentStatus         ::=      [27]   IMPLICIT INTEGER{
                                             success         (0),
                                             partial-1       (1),
                                             partial-2       (2),
                                             partial-3       (3),
                                             partial-4       (4),
                                             failure         (5)}

-- begin definition of composition specification
 CompSpec ::= SEQUENCE{
 selectAlternativeSyntax    [1] IMPLICIT BOOLEAN,
                                             -- See comment for recordSyntax, below.
 generic                    [2] IMPLICIT Specification OPTIONAL,
 dbSpecific                 [3] IMPLICIT SEQUENCE OF SEQUENCE{
                                                   db      [1] DatabaseName,
                                                   spec    [2] IMPLICIT Specification} OPTIONAL,
            -- At least one of generic and dbSpecific must occur, and both may occur. If both, then for
            -- any record not in the list of databases within dbSpecific, generic applies.
 recordSyntax               [4] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL
                                             -- For each record, the target selects the first record syntax
                                             -- in this list that it can support.  If the list is exhausted, the
                                             -- target may select an alternative syntax if
                                             -- selectAlternativeSyntax is 'true'.
                            }
 Specification ::= SEQUENCE{
 schema                [1] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
 elementSpec           [2] CHOICE{
                                   elementSetName       [1] IMPLICIT InternationalString,
                                   externalEspec        [2] IMPLICIT EXTERNAL} OPTIONAL}
 -- end definition of composition specification
 -- end auxiliary definitions for search and response APDUs

-- Delete 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 InternationalString 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)}
--

--Access- and Resource-control APDUs
--
 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} OPTIONAL,
                                               -- Optional only in version 3; mandatory in version 2. If
                                               -- omitted (in version 3) then diagnostic must occur.
 diagnostic                     [223]          DiagRec OPTIONAL, -- Version 3 only.
 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 APDUs
 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]   IMPLICIT ListEntries  OPTIONAL,
  attributeSet                      [8]   IMPLICIT AttributeSetId OPTIONAL,
  otherInfo                               OtherInformation OPTIONAL}

-- begin auxiliary definitions for Scan
  ListEntries ::= SEQUENCE{
  entries                           [1]   IMPLICIT SEQUENCE OF Entry OPTIONAL,
  nonsurrogateDiagnostics           [2]   IMPLICIT SEQUENCE OF DiagRec OPTIONAL
-- At least one of entries and nonsurrogateDiagnostics must occur
                                    }

  Entry  ::= CHOICE {
  termInfo                          [1]   IMPLICIT TermInfo,
  surrogateDiagnostic               [2]   DiagRec}
--
  TermInfo ::= SEQUENCE {
  term                                    Term,
  displayTerm                       [0]   IMPLICIT InternationalString OPTIONAL,
                                    -- Presence of displayTerm means that term is not considered by
                                    -- the target to be suitable for display, and displayTerm should
                                    -- instead be displayed. 'term' is the actual term in the term list;
                                    -- 'displayTerm' is for display purposes only, and is not an actual
                                    -- term in the term list.
  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}}
  otherOccurInfo                    OtherInformation OPTIONAL}
-- end auxiliary definitions for Scan

-- Sort APDUs
  SortRequest  ::= SEQUENCE{
  referenceId                   ReferenceId OPTIONAL,
  inputResultSetNames     [3]   IMPLICIT SEQUENCE OF InternationalString,
  sortedResultSetName     [4]   IMPLICIT InternationalString,
  sortSequence            [5]   IMPLICIT SEQUENCE OF SortKeySpec,
                                            -- order of occurrence is from major to minor
  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
  SortKeySpec ::= SEQUENCE{
  sortElement                          SortElement,
  sortRelation              [1]        IMPLICIT INTEGER{
                                             ascending                (0),
                                             descending               (1),
                                             ascendingByFrequency     (3),
                                             descendingByfrequency    (4)},
  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{
  generic                   [1]        SortKey,
  datbaseSpecific           [2]        IMPLICIT SEQUENCE OF SEQUENCE{
                            databaseName         DatabaseName,
                            dbSort               SortKey}}

  SortKey ::= CHOICE{
  sortfield                 [0]        IMPLICIT InternationalString,
                            -- 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

-- Extended Service APDUs
  ExtendedServicesRequest  ::= SEQUENCE{
  referenceId                                ReferenceId OPTIONAL,
  function                        [3]        IMPLICIT INTEGER {
                                                   create        (1),
                                                   delete        (2),
                                                   modify        (3)},
  packageType                     [4]        IMPLICIT OBJECT IDENTIFIER,
  packageName                     [5]        IMPLICIT InternationalString OPTIONAL,
                                             -- PackageName mandatory for 'modify' or 'delete'; optional for
                                             -- 'create'. Following four parameters mandatory for 'create'; should
                                             -- be included on 'modify' if being modified; not needed on 'delete'.
  userId                        [6]    IMPLICIT InternationalString OPTIONAL,
  retentionTime                 [7]    IMPLICIT IntUnit OPTIONAL,
  permissions                   [8]    IMPLICIT Permissions OPTIONAL,
  description                   [9]    IMPLICIT InternationalString OPTIONAL,

-- (ExtendedServiceRequest APDU continued)
  taskSpecificParameters        [10]   IMPLICIT EXTERNAL OPTIONAL,
                                              -- Mandatory for 'create'; included on 'modify' if specific
                                              -- parameters being modified; not necessary on 'delete'. For the
                                              -- 'EXTERNAL,' use OID of specific ES definition and select
                                              --  CHOICE [1]: 'esRequest'.
  waitAction                    [11]   IMPLICIT INTEGER{
                                              wait              (1),
                                              waitIfPossible (2),
                                              dontWait          (3),
                                              dontReturnPackage (4)},
  elements                             ElementSetName OPTIONAL,
  otherInfo                            OtherInformation OPTIONAL}
--

ExtendedServicesResponse ::= SEQUENCE{
  referenceId                          ReferenceId OPTIONAL,
  operationStatus               [3]    IMPLICIT INTEGER{
                                              done              (1),
                                              accepted          (2),
                                              failure      (3)},
  diagnostics                   [4]    IMPLICIT SEQUENCE OF DiagRec OPTIONAL,
  taskPackage                   [5]    IMPLICIT EXTERNAL OPTIONAL,
                                              -- Use OID: {Z39-50-recordSyntax (106)} and corresponding
                                              -- syntax. For the EXTERNAL, 'taskSpecific,' within that
                                              -- definition, use OID of the specific es, and choose [2],
                                              -- 'taskPackage'.
  otherInfo                            OtherInformation OPTIONAL}

Permissions ::= SEQUENCE OF SEQUENCE{
  userId                        [1] IMPLICIT InternationalString,
  allowableFunctions            [2] IMPLICIT SEQUENCE OF INTEGER{
                                              delete            (1),
                                              modifyContents    (2),
                                              modifyPermissions (3),
                                              present      (4),
                                              invoke       (5)}}

Close ::= SEQUENCE{
  referenceId                          ReferenceId OPTIONAL,  -- See 3.2.11.1.5.
  closeReason                          CloseReason,
  diagnosticInformation         [3]    IMPLICIT InternationalString OPTIONAL,
  resourceReportFormat          [4]    IMPLICIT ResourceReportId OPTIONAL,
                                              -- For use by origin only, and only on Close request;
                                              -- origin requests target to include report in response.
  resourceReport                [5]    ResourceReport OPTIONAL,
                                              -- For use by target only, unilaterally on Close request;
                                              -- on Close response may be unilateral or in response
                                              -- to origin request.
  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
 ResultSetId         ::=          [31]    IMPLICIT InternationalString
 ElementSetName      ::=          [103]   IMPLICIT InternationalString
 DatabaseName        ::=          [105]   IMPLICIT InternationalString
 AttributeSetId      ::=                  OBJECT IDENTIFIER

-- OtherInformation
 OtherInformation     ::=         [201]   IMPLICIT SEQUENCE OF SEQUENCE{
 category                         [1]     IMPLICIT InfoCategory OPTIONAL,
 information                              CHOICE{
 characterInfo                    [2]     IMPLICIT InternationalString,
 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}

-- Units
 -- 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.

 IntUnit ::= SEQUENCE{
 value                      [1] IMPLICIT INTEGER,
 unitUsed                   [2] IMPLICIT Unit}
--
 Unit ::= SEQUENCE{
 unitSystem                 [1] InternationalString 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
 InternationalString ::= GeneralString
                                  -- When version 2 is in force, this collapses to VisibleString. That is, only characters in the
                                  -- visibleString repertoire may be used. (Datatype compatibility with version 2 is not affected,
                                  -- because references are IMPLICIT.)  When version 3 is in force, the semantics of the
                                  -- GeneralString content may be altered by negotiation during initialization. If no such
                                  -- negotiation is in effect, then GeneralString semantics are in force.

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

END -- IR DEFINITIONS
```

4.2 Protocol Procedures

Protocol procedures are described in this section. Rules for extensibility and conformance requirements are specified in sections 4.3 and 4.4 respectively.

4.2.1 Presentation and Association Control Services

The Information Retrieval protocol may be used in conjunction with the presentation layer and the association control service element (ACSE).

4.2.1.1 Service Provided by the Presentation Layer

Z39.50 may use the presentation service as defined in ISO 8822 to provide a presentation connection for communication between a Z39.50 origin/target pair. The communication service that supports this protocol is a connection-oriented service defined in ISO 8822 in an established application association, in combination with ACSE, ISO 8649.

A Z39.50 origin establishes application-associations as necessary with the target. 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.

4.2.1.2 Association Control Services

The complete application service may include 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. The life of an A-association has three distinct phases: 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 that are valid during the information transfer phase. Prior to the exchange of Z39.50 APDUs, the Information Retrieval service user invokes 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 Annex 2, CTX.

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.

4.2.2 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 is modeled 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:

- Request — A primitive issued by the origin to the service-provider in order to invoke some procedure
- Indication — A primitive issued by the service-provider to the target service-user to indicate that a procedure has been invoked by its peer
- Response — A primitive issued by the target service-user to the service-provider at the completion of the procedure previously invoked by an indication
- 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 drawn between the portion of the service-provider residing on the client and the portion of the service-provider residing on server (respectively, the origin and the target). 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 to target.
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 to origin.
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 that occur for a Search operation:

1. Search request from origin service-user to service-provider.
2. Search APDU from origin to target.
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 to origin.
6. Search confirm from service-provider to origin service-user.

The interactions 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 the means by which they are implemented are not constrained by this specification. For example, in an actual implementation the target service-user and service-provider might be combined in a single program, and 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 (Tables 16 through 22). 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-machines"). One O-machine is for a Present operation and one O-machine is 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 interrelationship 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 semicolons (;). The last such action specified is always a transition to the resulting state, in parentheses.

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 semicolons. The following predicates and variables 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.

Variable	Meaning
<op>	An operation type (other than Init): search, present, delete, scan, sort, resource-report, Extended-services.
opCnt	Number of active operations.
retSt	Return state. An integer; 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 that pertain to an operation are reflected in the operation state table, except for those that occur during initialization (those are shown in part 1 of the Z-association table) and those that pertain to an aborted operation (those apply to part 3 of the Z-association table). Those that 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 successfully 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: The origin is awaiting an Init request from the service-user.
1. Init Sent: The origin is 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 service-user.
3. Rsc Recvd: During initialization the origin has received a resource-control PDU and is awaiting a Resource-control response from the service-user.
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 service-user.

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 service-user.

10. Close sent: The origin is awaiting a Close PDU from the target.
11. Close Received: The origin is awaiting a Close response from the service-user.

Origin States for Operation:

1. *For Present operation:* Present sent: The origin is awaiting a Present-response PDU from the target. *For operation other than Present:* <Op> sent: The origin is awaiting an <Op>-response PDU from the target.

2. Rsc recvd: The origin has received a Resource-control-request PDU pertaining to the operation and is awaiting a Resource-control response from the service-user.
3. Acc recvd: The origin has received an Access-control-request PDU pertaining to the operation and is awaiting an Access-control response from the service-user.

Target States

Target States for Z-association:

0. Closed: The target is awaiting an Init PDU from the origin.
1. Init recvd: The target is awaiting an Init Response from the service-user.
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: The target is awaiting a Close PDU from the origin.
11. Close Received: The target is awaiting a Close response from the service-user.

Target States for Operation:

1. *For Present operation:* Present sent: The target is awaiting a Present response from the service-user. *For operation other than Present:* <Op> sent: The target is awaiting an <Op>-response PDU from the service-user.
2. Rsc sent: 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 sent: 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

Table 16 lists the events and actions that appear in the state tables (Tables 17-22). 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.

Table 16: Abbreviations of Events and Actions in State Tables

Abbreviation	**Meaning**
<op> PDU	<operation type> PDU
<op> req	<operation type> request
<op> resp	<operation type> response
<op> conf	<operation type> confirm
<op> resp PDU	<operation type> Response PDU
Acc conf	Access-control confirm
Acc ind	Access-control indication
Acc PDU	Access-control PDU
Acc req	Access-control request
Acc resp	Access-control response
Acc Resp PDU	Access-control-response PDU
AnyOpPdu	Any PDU belonging to an operation
AnyPdu	Any PDU except Close
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 conf	Present confirm
Prsnt resp PDU	Present-response PDU
Prsnt resp	Present response
Rsc conf	Resource-control confirm
Rsc ind	Resource-control indication
Rsc PDU	Resource-control PDU
Rsc req	Resource-control request
Rsc resp	Resource-control response
Rsc resp PDU	Resource-control-response PDU
Seg ind	Segment Indication
Seg PDU	Segment PDU

Abbreviation	**Meaning**
Seg req	Segment request
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 conf	Resource-control confirm (Z-association)
Z-Rsc PDU	Resource-control PDU (Z-association)
Z-Rsc req	Resource-control request (Z-association)
Z-Rsc req noResp	Resource-control request, “no response” (Z-association)
Z-Rsc resp	Resource-control response (Z-association)
Z-Rsc resp PDU	Resource-control-response PDU (Z-association)

Miscellaneous actions

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 17: State Table 1 (part 1 -- 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 17: State Table 1 (part 2 -- Origin Z39.50 Association: Processing Phase)

State Event	Serial Idle 4	Concurrent 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 17: State Table 1 (part 3 -- 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 18: State Table 2 -- 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 conf; EndOp ind; exit		

Table 19: State Table 3 -- 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> conf; EndOp ind; exit		

Table 20: State Table 4 (part 1 -- 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; (1)	
Rsc req		Rsc PDU; :[resp] (3) else (1):		
Rsc resp PDU				Rsc conf; (1)

Table 20: State Table 4 (part 2 -- Target Z39.50 Association: Processing Phase)

State Event	Serial Idle (4)	Concurrent 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 conf; (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 20: State Table 4 (part 3 -- 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 21: State Table 5 -- 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 22: State Table 6 -- Target Operation Other Than Present			
State **Event**	**<op> 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)		
<op> resp	<op> resp PDU; EndOp ind; exit		

4.2.4 Protocol Errors
Any event not listed in the tables of section 4.2.3 is not valid and is considered to be a protocol error. 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.

Additional conditions that may be treated as protocol errors are described in 4.4.2.2.

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 conformance requirements in 4.4.1. These requirements are elaborated in 4.4.2.

4.4.1 General Conformance Requirements
The system shall:

(a) Act in the role of origin or target.
(b) Support the Init, Search, and Present services. See 4.4.2.2.1.
(c) Support the syntax in 4.1.
(d) Support the Type-1 Query. See 4.4.2.2.2.
(e) Support (at minimum) version 2 of the protocol.
(f) Follow the procedures specified in sections 3, 4.1, 4.2, and 4.3.
(g) Assign values to APDU data elements according to the procedures of sections 3 and 4.1.

4.4.2 Specific Conformance Requirements
Section 4.4.2.1 provides a table of Z39.50 features for which 4.4.2.2 specifies conformance requirements. In particular, conformance requirements are described as they pertain to version 2 and version 3 respectively.

4.4.2.1 Z39.50 Features
Table 23, Z39.50 features, indicates the applicable protocol version (2 or 3), a reference to a description of the feature, and a reference to the section within 4.4.2.2 that describes conformance requirements for the feature. The "item" column is used by the sections within 4.4.2.2 to refer back to the table.

Table 23: Z39.50 Features, Protocol Version, and Conformance

Item	Feature	Version	Reference	Conformance
1	**Init Service**	V2 and V3	3.2.1.1	4.4.2.2.1
2	**Search Service**	V2 and V3	3.2.2.1	4.4.2.2.1
3	Query type-1	V2 and V3	3.7	4.4.2.2.2
4	Multiple attribute sets	V3	Note 1	4.4.2.2.3
5	Multiple data types for search term	V3	Note 2	4.4.2.2.3
6	Complex attribute values	V3	Note 3	4.4.2.2.3
7	Result set restriction	V3	3.7	4.4.2.2.3
8	Proximity	V3	3.7.2	4.4.2.2.4
9	Query type-101	V2 and V3	3.7	4.4.2.2.4
10	Query types 0, 2, 100	V2 and V3	3.2.2.1.1	4.4.2.2.4
11	Query type 102	V3	3.2.2.1.1	4.4.2.2.5

Item	Feature	Version	Reference	Conformance
12	Additional-search-information parameter in Search request and response	V3	3.2.2.1.12	4.4.2.2.6
13	Named result sets	V2 and V3	3.2.2.1.3	4.4.2.2.23
14	**Present Service**	V2 and V3	3.2.3.1	4.4.2.2.1
15	Additional-ranges and Comp-spec parameters on Present request	V3	3.2.3.1.2, 3.2.3.1.6	4.4.2.2.7
16	Max- segment-count, -segment-size, -record-size parameters on Present request	V3	3.2.3.1.7	4.4.2.2.8
17	Diagnostic format -- default form	V2 and V3	Note 4	4.4.2.2.9
18	Diagnostic format -- external form	V3	Note 4	4.4.2.2.9
19	addinfo type VisibleString	V2, V3	Note 5	4.4.2.2.10
20	addinfo type InternationalString	V3	Note 5	4.4.2.2.10
21	Multiple non-surrogates in Search or Present response	V3	Note 6	4.4.2.2.11
22	**Segment Service**	V3	3.2.3.2	4.4.2.2.12
23	Level-1 segmentation	V3	3.3.2	4.4.2.2.12
24	Level-2 segmentation	V3	3.3.3	4.4.2.2.12
25	**Delete Service**	V2 and V3	3.2.4.1	4.4.2.2.13
26	failure-10 value of Delete-list-status on Delete response	V3	3.2.4.1.4	4.4.2.2.15
27	**Access-control Service**	V2 and V3	3.2.5.1	4.4.2.2.14
28	Security-challenge-response and diagnostic in Access-control response	V3	Note 7	4.4.2.2.16
29	**Resource-control Service**	V2 and V3	3.2.6.1	4.4.2.2.14
30	**Trigger-resource-control Service**	V2 and V3	3.2.6.2	4.4.2.2.13
31	**Resource-report Service**	V2 and V3	3.2.6.3	4.4.2.2.13
32	Op-id parameter of Resource-report-request	V3	3.2.6.3.2	4.4.2.2.17
33	failure-5 and failure-6 values of Resource-report-status in Resource-report response	V3	3.2.6.3.3	4.4.2.2.18
34	**Sort Service**	V2 and V3	3.2.7.1	4.4.2.2.13
35	**Scan Service**	V2 and V3	3.2.8.1	4.4.2.2.13
36	**Extended-Services Service**	V2 and V3	3.2.9.1	4.4.2.2.13
37	**Close Service**	V3	3.2.11.1	4.4.2.2.19
38	Explain facility	V2 and V3	3.2.10	4.4.2.2.20
39	Other-information (in a request or response other than Scan, Sort, or Extended Services)	V3	Note 8	4.4.2.2.6
40	Other-information in Scan, Sort, and ES	V2 and V3	Note 8	4.4.2.2.21
41	Concurrent Operations	V3	3.5	4.4.2.2.22
42	InternationalString full use of GeneralString repertoire	V3	Note 9	4.4.2.2.24
43	Reference Id	V2 and V3	3.4	4.4.2.2.25

Notes:

(1) In version 2 a type-1 query includes a single, global attribute set id, which identifies an attribute set definition that pertains to all of the attributes within the query. In version 3 a type-1 query also includes a global attribute set id, but in addition, each attribute within the query may also be qualified with an attribute set id (which, If included, overrides the global attribute set id).

(2) In version 2 a search term must be of ASN.1 type OCTET STRING. In version 3 it may be any of the following: OCTET STRING, INTEGER, InternationalString, OBJECT IDENTIFIER, GeneralizedTime, EXTERNAL, IntUnit, or NULL.

(3) In version 2, in a type-1 query, an attribute value must be numeric (i.e. ASN.1 type INTEGER). In version 3, an attribute value may be numeric or 'complex'. The complex form may include multiple values, each either numeric or character string, and a semantic action indicator (corresponding to some semantic action defined within the attribute set definition).

(4) See introductory text of Annex ERR.

(5) In version 2, when using default diagnostic format, the addInfo parameter must be ASN.1 type VisibleString. In version 3 it may be type InternationalString.

(6) In version 2, a Search or Present response may include at most a single non-surrogate diagnostic record. In version 3 a Search or Present response may include multiple non-surrogate diagnostic records. (Responses other than Search or Present that include diagnostics may include multiple non-surrogate diagnostics regardless of version.)
(7) In version 2, in the Access control response, securityChallengeResponse must occur, and no diagnostic may occur. In version 3, securityChallengeResponse may be omitted, if the parameter 'diagnostic' is present.
(8) In version 2, the parameter otherInformation may be used only in Scan, Sort, and Extended Services requests and responses. In version 3 it may be used in any request or response.
(9) See definition of InternationalString in ASN.1 for APDUs.

4.4.2.2 Detailed Requirements

4.4.2.2.1 Init, Search, and Present Services *(See items 1, 2 and 14 in Table 23).* A system must support the Init, Search, and Present services.

This means that an origin must be capable of sending Init, Search, and Present requests and receiving the respective responses. A target must respond properly to Init, Search, and Present requests with respective responses.

An origin may indicate (via option bits) during initialization that it does not intend to utilize the Present service during the Z-association; this does not constitute non-conformance. If, however, an origin indicates that it does intend to utilize the Present service, and the target refuses, this does constitute non-conformance on the part of the target.

This requirement is independent of version.

4.4.2.2.2 Type-1 Query *(See item 3 in Table 23).* An origin must be capable of formulating a type-1 query within a Search request, and a target should expect to receive a type-1 query.

An origin or target may support other query types. If the origin fails to send a type-1 query during a Z-association, this does not constitute non-conformance on the part of the origin. If, however, the origin does send a type-1 query and the target responds with a diagnostic indicating “query type not supported” this does constitute non-conformance on the part of the target.

This requirement does not mean that any specific feature of the type-1 query must be supported. A target that receives a type-1 query that conforms to the type-1 query syntax but which includes a feature that it does not support must not treat this condit ion as a protocol error (but instead should return an appropriate diagnostic, however, that diagnostic must not indicate “query type not supported”).

This requirement is independent of version.

4.4.2.2.3 Multiple attribute sets, Multiple data types for search term, Complex attribute values, Result set restriction, and Proximity *(See items 4, 5, 6, 7, and 8 in Table 23).* For version 2, the origin may not use any of these features in a type-1 query. If target receives a type-1 query with any of these features, it may treat this condition as a protocol error.

For version 3, the origin may, but is not required, to use any of these features in a type-1 query. The target should expect type-1 queries to include any or all of these features, but is not required to support any of these features. If the target receives a type-1 query which includes any of these features that it does not support, it must not treat this condition as a protocol error (but rather should return an appropriate diagnostic).

4.4.2.2.4 Query types 0, 2, 100, and 101 *(See items 9 and 10 in Table 23).* An origin is not required to support queries of any of these types. A target should expect to receive, but need not support queries of these types. If a target receives a query of one of these types that it does not support it must not treat this condition as a protocol error but instead should return a diagnostic indicating that the query type is not supported.

This requirement is independent of version.

4.4.2.2.5 Query Type-102 *(See item 11 in Table 23).* For version 2, an origin may not use the type-102 query. If a target receives a type-102 query it may treat this condition as a protocol error.

For version 3, an origin may, but need not support the type-102 query. A target should expect to receive, but need not support, type-102 queries; if it receives a type-102 query it must not treat this condition as a protocol error.
Note: ISO 23950 lists type-102 as a valid query type (for version 3) but does not include a definition.

4.4.2.2.6 Additional-search-information parameter in Search request or response; Other-information parameter in any request or response other than Scan, Sort, or Extended Services *(See items 12 and 39 in Table 23).* For version 2, a system may not use these parameters; if a system receives one of these parameters it may treat this condition as a protocol error.

For version 3, a system is never required to use any of these parameters. However, a system should expect to receive these parameters, but is not required to interpret or process the information contained within any of these parameters.

4.4.2.2.7 Additional-ranges and Comp-spec parameters on Present request *(See item 15 in Table 23).* For version 2, the origin may not use these parameters. If the target receives one of these parameters it may treat this condition as a protocol error.

For version 3, the origin is not required to, but may use either of these parameters. The target should expect to receive, but need not support either of these parameters. If the target receives but does not support one of these parameters, it should not treat this condition as a protocol error (but instead should return an appropriate status value and/or diagnostic).

4.4.2.2.8 Max-segment-count, Max-segment-size, and Max-record-size parameters on Present request *(See item 16 in Table 23).* For version 2, as well as for version 3 when segmentation is not in effect, the origin may not use these parameters; if the target receives any of these parameters it may treat this condition as a protocol error.

For version 3:

- If level-1 segmentation is in effect:

  — The origin may but is not required to support Max-segment-count. The target should expect to receive, but need not support Max-segment-count. If the target receives but does not support Max-segment-count, it must not treat this condition as a protocol error (but instead should return an appropriate status value and/or diagnostic).
  — The origin may not use Max-segment-size or Max-record-size. If target receives either it may treat this condition as a protocol error.

- If level-2 segmentation is in effect:

  — The origin may but is not required to support any of these three parameters. The target should expect to receive, but need not support any of these parameters. If the target receives but does not support a parameter, it must not treat this condition as a protocol error (but instead should return an appropriate status value and/or diagnostic).

4.4.2.2.9 Diagnostic format *(See items 17 and 18 in Table 23).* For version 2, the target may send diagnostics in a Search or Present response using the default form only. If the origin receives a diagnostic which does not conform to the default form, it may treat this condition as a protocol error.
Note: This rule applies to Search and Present responses only. Responses other than Search or Present that include diagnostics are not affected.

For version 3, the target may send diagnostics using the default or external form. The origin should expect to receive diagnostics in either form.

4.4.2.2.10 Addinfo of default diagnostic format *(See items 19 and 20 in Table 23).* For version 2, when the target sends a diagnostic in a Search or Present response using the default form, the addinfo parameter must be of ASN.1 type VisibleString. If the origin receives a diagnostic that violates this rule, it may treat this condition as a protocol error.

For version 3 the addinfo parameter may be of either type VisibleString or InternationalString.

4.4.2.2.11 Multiple non-surrogates in Search or Present response *(See item 21 in Table 23).* For version 2, the target must not include multiple non-surrogate diagnostics in a Search or Present response; if it does so, the origin may treat this condition as a protocol error.
Note: This rule applies to Search and Present responses only. There are responses other than Search or Present that include diagnostics, and these are not affected.

For version 3, the target may (but is not required to) include multiple non-surrogate diagnostics in a Search or Present response and if it does, the origin must not treat this condition as a protocol error.

4.4.2.2.12 Segmentation *(See items 22, 23, and 24 in Table 23).* For version 2, as well as for version 3 when segmentation is not in effect, the target may not send a Segment request, and if it does, the origin may treat this condition as a protocol error.

For version 3, level-1 or level-2 segmentation may be negotiated, however neither the target not the origin is required to support segmentation.

4.4.2.2.13 Delete service, Trigger-resource-control service, Resource-report service, Sort service, Scan service, and Extended-Services service *(See items 25, 30, 31, 34, 35, and 36 in Table 23).* A system is not required to support any of these services. They are independently negotiable. If the target receives a request of one of these types and the respective service is not in effect, it may treat this condition as a protocol error.

This requirement is independent of version.

4.4.2.2.14 Access-control and Resource-control services *(See items 27 and 29 in Table 23).* A system is not required to support either of these services. They are independently negotiable. If the origin receives an Access-control or Resource-control request and the respective service is not in effect (or if the request occurs while the origin is awaiting an Init response and the origin has not proposed the respective option in the Init request), it may treat this condition as a protocol error.

This requirement is independent of version.

4.4.2.2.15 'failure-10' value of Delete-list-status on Delete response *(See item 26 in Table 23).* For version 2, the target may not return this value; if it does the origin may treat this condition as a protocol error.

For version 3, the target may return this value.

4.4.2.2.16 Security-challenge-response and Diagnostic in Access-control response *(See item 28 in Table 23).* For version 2, the origin must include in the Access-control response the parameter Security-challenge-response, and may not include a diagnostic. If the target receives an Access-control response that violates this rule it may treat this condition as a protocol error.

For version 3, the origin may include a diagnostic, and if so, the parameter securityChallengeResponse may be omitted.

4.4.2.2.17 Op-id parameter of Resource-report request *(See item 32 in Table 23).* For version 2, the origin may not use this parameter; if the target receives this parameter it may treat this condition as a protocol error.

For version 3, the origin may, but is not required to include this parameter. The target should expect to receive, but need not support the parameter. If the target receives but does not support this parameter, it should not treat this condition as a protocol error but instead should return an appropriate status.

4.4.2.2.18 failure-5 and failure-6 Resource-report-status in Resource-report response *(See item 33 in Table 23).* For version 2, the target may not return either value for this status; if it does the origin may treat this condition as a protocol error.

For version 3, the target may return either value.

4.4.2.2.19 Close service *(See item 37 in Table 23).* For version 2, the Close service may not be used. If a system receives a Close request, it may treat this condition as a protocol error.

For version 3, a system must expect to receive a Close request, and must be capable of responding with a Close response. A system is not required to send a Close request.

4.4.2.2.20 Explain facility *(See item 38 in Table 23).* There are no conformance requirements pertaining to the Explain facility, either for version 2 or version 3. A system may choose to support or not support Explain.

Note that implementation of Explain requires, at minimum, support for searching the Explain database and for the Explain record syntax. This standard does not require support for searching any particular database or support for any particular record syntax.

4.4.2.2.21 Other-information parameter in Scan, Sort, and Extended Services request *(See item 40 in Table 23).* The parameter Other-information may occur in a Scan, Sort, or Extended Services request or response. A system should expect to receive this parameter, but is not required to interpret or process the information contained within the parameter.

This requirement is independent of version.

4.4.2.2.22 Concurrent Operations *(See item 41 in Table 23).* For version 2, as well as for version 3 when concurrent operations is not in effect, if an origin attempts to initiate concurrent operations (i.e., attempts to initiate an operation when an operation is already active), the target may treat this as a protocol error.

For version 3, a system may choose to support or not to support concurrent operations.

4.4.2.2.23 Named Result sets *(See item 13 in Table 23).* A system may choose to support or not support named result sets. If the target receives a Search request where the value of the parameter Result-set-id is other than 'default' and the target does not support named result sets, the target should not treat this condition as a protocol error but should instead return an appropriate diagnostic.

This requirement is independent of version.

4.4.2.2.24 InternationalString Definition *(See item 42 in Table 23).* For version 2, a value of a parameter of ASN.1 type InternationalString must conform to the VisibleString definition. A system that receives a value that violates this rule may treat this condition as a protocol error.

For version 3, a value of a parameter of ASN.1 type InternationalString must conform to the GeneralString definition. A system that receives a value that does not conform to the VisibleString definition (but does conform to the GeneralString definition) must not treat this condition as a protocol error.

4.4.2.2.25 Reference-id *(See item 43 in Table 23).* For both version 2 and version 3, an origin may choose to support or not support the Reference-id parameter; a target must support the Reference-id parameter. Note, however, for version 3, origin support of concurrent operations (see 4.4.2.2.22) implies support for the reference-id parameter.

Annex 1
(Normative)
OID: Z39.50 Object Identifiers

OID.1 Object Identifier Assigned to This Standard

ANSI has assigned the following object identifier to this standard:
{iso (1) member-body (2) US (840) ANSI-standard-Z39.50 (10003)}

Note:This OID was originally assigned to Z39.50-1992; it applied also to Z39.50-1995; it now applies also to ISO 23950. Z39.50 object identifiers are maintained subordinate to this object identifier. No object identifiers are maintained subordinate to the nominal ISO 23950 object identifier { ISO Standard 23950}.

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 appendixes that follow.

```
ANSI-Z39-50-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 Annex 2, CTX.
Z39-50-APDU               OBJECT IDENTIFIER ::= {Z39-50 2}     -- See Annex 1, OID.3.1.
Z39-50-attributeSet       OBJECT IDENTIFIER ::= {Z39-50 3}     -- See Annex 3, ATR.
Z39-50-diagnostic         OBJECT IDENTIFIER ::= {Z39-50 4}     -- See Annex 4, ERR.
Z39-50-recordSyntax       OBJECT IDENTIFIER ::= {Z39-50 5}     -- See Annex 5, REC.
Z39-50-transferSyntax     OBJECT IDENTIFIER ::= {Z39-50 6}     -- See note below.
Z39-50-resourceReport     OBJECT IDENTIFIER ::= {Z39-50 7}     -- See Annex 6, RSC.
Z39-50-accessControl      OBJECT IDENTIFIER ::= {Z39-50 8}     -- See Annex 7, ACC.
Z39-50-extendedService    OBJECT IDENTIFIER ::= {Z39-50 9}     -- See Annex 8, EXT.
Z39-50-userInfoFormat     OBJECT IDENTIFIER ::= {Z39-50 10}    -- See Annex 9, USR.
Z39-50-elementSpec        OBJECT IDENTIFIER ::= {Z39-50 11}    -- See Annex 10, ESP.
Z39-50-variantSet         OBJECT IDENTIFIER ::= {Z39-50 12}    -- See Annex 11, VAR.
Z39-50-schema             OBJECT IDENTIFIER ::= {Z39-50 13}    -- See Annex 12, TAG.
Z39-50-tagSet             OBJECT IDENTIFIER ::= {Z39-50 14}    -- See Annex 12, TAG.
END
```

No object identifier is assigned by this standard for any transfer syntax. For the purpose of presentation context negotiation for an abstract syntax (including the abstract syntax for the APDUs, defined in 4.1), the abstract syntax is paired with a transfer syntax. This pairing is represented by a pair of object identifiers, one for the abstract-syntax (e.g., Z39.50-APDU) and one for the encoding rules. For abstract syntaxes described using ASN.1 (e.g., Z39.50-APDU), a set of basic encoding rules are specified by "ASN.1 Basic Encoding Rules," ISO 8825, identified by the following object identifier:

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

OID.3 Object Identifiers Assigned by This Standard

All object identifiers assigned by this standard (with the exception of the OID for Z39.50 APDUs, assigned in OID.3.1) are explicitly assigned in the appendixes that follow.

OID.3.1 Object Identifiers for Z39.50 APDUs

This standard assigns the following object identifier for the ASN.1 definition of APDUs in 4.1.

Z39-50-APDU {Z39-50-APDU 1}

Note: the same OID is used for APDUs both for Z39.50-1992 and Z39.50-1995 (as well as ISO 23950), because of the interworking capability between the two versions.

OID.4 Object Identifiers Used by This Standard

Z39.50 object identifiers are either *public* or *locally defined*. Public Z39.50 object identifiers are officially registered by this standard or by the Z39.50 Maintenance Agency (see OID.5). Locally defined Z39.50 object identifiers are registered by a registered Z39.50 implementor (see OID.6 and OID.7).

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

Additional object identifiers (official Z39.50 object identifiers not registered by this standard) may be assigned by the Z39.50 Maintenance Agency (see note), in 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 take 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 *published* or *private*. Local published objects are those whose definitions are coordinated with and published by the Z39.50 Maintenance Agency. Local private objects are those whose definitions are not published by the Z39.50 Maintenance Agency.

OID.7 Experimental Objects

Experimental objects take the form:

{Z39-50 n 2000 p m}

where {z39-50 n} is as described in OID.5, and 'p' is the OID index of a registered Z39.50 Implementor.

Annex 2
(Normative)
CTX: Application Context basic-Z39.50-ac

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 1 }

Definition of application context basic-Z39.50-ac

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

1. the association control service element (ACSE, ISO 8650), and

2. the Z39.50 service element.
   Z39.50 and ACSE are used according to the procedures in section 4.2.1.

The presentation services required are those contained in the presentation kernel functional unit and the session duplex functional unit. All Information Retrieval protocol data units will be mapped onto the P-Data service.

In the event of protocol errors, the system detecting the error shall abort the association.

Only the origin may invoke the A-RELEASE service (to initiate orderly release of an A-association).

Annex 3
(Normative)
ATR: Attribute Sets

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}**	
STAS	**{Z39-50-attributeSet 6}**	

Each attribute set defines a set of types and for each type a set of values. An attribute list (see AttributeList in the ASN.1 for APDUs, 4.1), constructed from an attribute set definition, is a list of attribute pairs. An attribute pair (AttributeElement in the ASN.1 for APDUs) consists of an attribute type and a value list (attributeValue within AttributeElement), where each value in the list is defined for that type.

When version 2 is in force, each value list is a single value and is an integer. When version 3 is in force, attributeValue (within AttributeElement) may select 'complex', allowing the value list to include multiple values (each may be integer or string) and also to specify a 'semanticAction', indicating how the target is to treat the multiple attributes.

When an attribute list contains any attribute pair where attributeValue selects 'complex', there must not be any attribute type within the attribute list for which there is more than a single attribute pair.

ATR.1 Attribute Set bib-1

This section defines the attribute set bib-1.

ATR.1.1 Bib-1 Types and Values

This section lists attribute types and values for attribute-set bib-1 (Tables A3-1 through A3-6).

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

Table A3-1: Bib-1 Use Attributes

Use	Value	Use	Value	Use	Value
Personal name	1	NAL call number	18	Title parallel	35
Corporate name	2	MOS call number	19	Title cover	36
Conference name	3	Local classification	20	Title added title page	37
Title	4	Subject heading	21	Title caption	38
Title series	5	Subject Rameau	22	Title running	39
Title uniform	6	BDI index subject	23	Title spine	40
ISBN	7	INSPEC subject	24	Title other variant	41
ISSN	8	MESH subject	25	Title former	42
LC card number	9	PA subject	26	Title abbreviated	43
BNB card no.	10	LC subject heading	27	Title expanded	44
BGF number	11	RVM subject heading	28	Subject precis	45
Local number	12	Local subject index	29	Subject rswk	46
Dewey classification	13	Date	30	Subject subdivision	47
UDC classification	14	Date of publication	31	No. nat'l biblio.	48
Bliss classification	15	Date of acquisition	32	No. legal deposit	49
LC call number	16	Title key	33	No. govt pub.	50
NLM call number	17	Title collective	34	No. music publisher	51

Table A3-1: Bib-1 Use Attributes (continued)

Use	Value	Use	Value	Use	Value
Number db	52	Author-name personal	1004	Editor	1020
Number local call	53	Author-name corporate	1005	Bib-level	1021
Code--language	54	Author-name conference	1006	Geographic-class	1022
Code--geographic area	55	Identifier--standard	1007	Indexed-by	1023
Code--institution	56	Subject--LC children's	1008	Map-scale	1024
Name and title	57	Subject name -- personal	1009	Music-key	1025
Name geographic	58	Body of text	1010	Related-periodical	1026
Place publication	59	Date/time added to db	1011	Report-number	1027
CODEN	60	Date/time last modified	1012	Stock-number	1028
Microform generation	61	Authority/format id	1013	Thematic-number	1030
Abstract	62	Concept-text	1014	Material-type	1031
Note	63	Concept-reference	1015	Doc-id	1032
Author-title	1000	Any	1016	Host-item	1033
Record type	1001	Server-choice	1017	Content-type	1034
Name	1002	Publisher	1018	Anywhere	1035
Author	1003	Record-source	1019	Author-Title-Subject	1036

Table A3-2: Bib-1 Relation Attributes

Relation	Value	Relation	Value	Relation	Value
less than	1	greater than	5	relevance	102
less than or equal	2	not equal	6	AlwaysMatches	103
equal	3	phonetic	100		
greater or equal	4	stem	101		

Table A3-3: Bib-1 Position Attributes

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

Table A3-4: Bib-1 Structure Attributes

Structure	Value	Structure	Value	Structure	Value
phrase	1	date (un-normalized)	100	document-text	106
word	2	name (normalized)	101	local number	107
key	3	name (un-normalized)	102	string	108
year	4	structure	103	numeric string	109
date (normalized)	5	urx	104		
word list	6	free-form-text	105		

Table A3-5: Bib-1 Truncation Attributes

Truncation	Value	Truncation	Value	Truncation	Value
right Truncation	1	do not truncate	100	regExpr-2	103
left truncation	2	process # in search term	101		
left and right	3	regExpr-1	102		

Table A3-6: Bib-1 Completeness Attributes

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

ATR.1.2 Bib-1 Attribute Combinations

If a target does not support a given attribute list, it should fail the search and supply an appropriate diagnostic.

A given attribute type may appear zero times, one time, or more than one time, in an attribute list.

- If an attribute type does not occur in an attribute list, then (in the absence of any prior understanding, either outside of the standard or via the Explain facility) the origin should not expect any particular default target behavior.
- If an attribute type occurs exactly once in an attribute list, then the attribute value specifies the preferred target behavior with respect to that attribute type.
- It is recommended that an attribute type not occur more than once in an attribute list, unless an associated "semantic action" is included (i.e. attributeValue selects 'complex').

When attributeValue selects 'complex', 'semanticAction' may be included. SemanticAction is a sequence of integers; for bib-1, it is either a single integer or a sequence of two integers.

For the first integer in the sequence, values are:

1 May not substitute another attribute. If none in the list is supported, fail the search.
2 May substitute another attribute, but only if none in the list is supported.
3 May substitute another attribute at target discretion (even if one or more in the list is supported).

The second integer in the sequence is to be supplied if and only if there are multiple attributes in the list.
Values are:

1 Select the first supported attribute in the list.
2 Select the best attribute in the list.

ATR.2 Attribute Set exp-1

This section defines the attribute-set exp-1, for use with an Explain database. The attribute set exp-1 defines a single attribute type, 'Use'. In addition, this attribute set definition *imports* non-Use bib-1 attributes, i.e. those of type Relation, Position, Structure, Truncation, and Completeness (see tables A-3-2 through A-3-6). The types and values defined within the bib-1 attribute set for these attributes may be used within the exp-1 attribute set, using the object identifier for this attribute set. It is recommended that a target supporting the Explain facility support the Relation attribute 'equal', Position attribute 'any' position in field', and Structure attribute 'key'.

Note: If the target supports searching based on date ranges (e.g. to limit a search to records created before or after a particular date or between two dates), the target should also support one or more of the following relation attributes: 'less than', 'less than or equal', 'greater than', and 'greater or equal'.

Table A3-7: Exp-1 Use Attributes

Use	Value	Use	Value	Use	Value
ExplainCategory	1	RecordSyntaxOID	6	DateExpires	11
HumanStringLanguage	2	TagSetOID	7	ElementSetName	12
DatabaseName	3	ExtendedServiceOID	8	ProcessingContext	13
TargetName	4	DateAdded	9	ProcessingName	14
AttributeSetOID	5	DateChanged	10	TermListName	15

Table A3-7: Exp-1 Use Attributes (continued)

Use	Value	Use	Value	Use	Value
SchemaOID	16	Proprietary	20	Keyword	25
Producer	17	UserFee	21	ExplainDatabase	26
Supplier	18	VariantSetOID	22	ProcessingOID	27
Availability	19	UnitSystem	23		

Notes:

(1) The search terms for Use attribute ExplainCategory are listed in table A-3-8.

(2) The search term when the Use attribute is HumanStringLanguage is the three-character language code from ANSI/NISO Z39.53-1994 -- Codes for the Representation of Languages for Information Interchange.

(3) The search terms when the Use attribute is ProcessingContext are listed in table A-3-9.

(4) Where the search term is an object identifier (where the name of the Use attribute ends with "OID"): for version 2, it is recommended that the term be a character string representing a sequence of integers (each represented by a character string) separated by periods. For version 3, it is recommended that the term be represented as ASN.1 type OBJECT IDENTIFIER.

(5) Use attribute Keyword is used when searching for DatabaseInfo records (i.e. in combination with an operand where Use is ExplainCategory and term is DatabaseInfo). Its use is to search in the keyword element, for terms that match one of the query terms.

(6) Use attribute ExplainDatabase is used when searching for DatabaseInfo records (i.e. in combination with an operand where Use is ExplainCategory and term is DatabaseInfo). The term should be NULL, for version 3, or otherwise ignored by the target. The Relation attribute either should be omitted or should be AlwaysMatches.

Table A3-8: Search terms associated with use attribute ExplainCategory

TargetInfo	TermListInfo	SortDetails
DatabaseInfo	extendedServicesInfo	Processing
SchemaInfo	AttributeDetails	CategoryList
TagSetInfo	TermListDetails	VariantSetInfo
RecordSyntaxInfo	ElementSetDetails	UnitInfo
AttributeSetInfo	RetrievalRecordDetails	

Table A3-9: Search terms associated with use attribute ProcessingContext

Access	RecordPresentation	RecordHandling
SearchRetrieval		

ATR.3 Attribute Set ext-1

This section defines the attribute-set ext-1, for use with an Extended Services database (see Tables A-3-10 and A-3-11). Two types are defined:

Attribute Type	Value
Use	1
Permissions	2

Additional attributes (types and/or values) may be defined within a specific Extended Service definition. The attribute set id to be used to identify those attributes is the ObjectIdentifier that identifies the specific Extended Service.

Table A3-10: Ext-1 Use Attributes

Use	Value	Use	Value	Use	Value
UserId	1	TaskStatus	4	TargetReference	7
PackageName	2	PackageType	5		
CreationDatetime	3	RetentionTime	6		

Table A3-11: Ext-1 Permission Attributes

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

Note: 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.

Annex 4
(Normative)
ERR: Error Diagnostics

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 force, a diagnostic record must conform to the following format:

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

The diagnostic record includes an integer corresponding to a condition or error, and an (object) identifier of the diagnostic set definition that lists the condition corresponding to that integer.

When version 3 is in force, a diagnostic record may assume the form above, or alternatively, may be defined as EXTERNAL, identified by an OBJECT IDENTIFIER (which identifies the diagnostic *format*, rather than the diagnostic *set*).

Bib-1 is a diagnostic *set*. It was originally defined and registered in Z39.50-1992. This standard includes an extended definition; the conditions listed below for bib-1 include all those that were listed in the Z39.50-1992 bib-1 definition, as well as several additional diagnostics that 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

Table A4-1 below is for use when DiagnosticSetId (within DefaultDiagFormat) equals the object identifier for diagnostic set bib-1, in which case Condition takes values from the "Code" column below.

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

AddInfo is ASN.1 type VisibleString. However, for several of the diagnostics below, AddInfo is used to express the value of a parameter that has an ASN.1 type other than VisibleString. Where Addinfo is used to express a numeric value, it should be a character string representation of that value. Where Addinfo is used to express an object identifier, it should take the form of a sequence of integers (each represented by a character string) separated by periods.

Table A4-1: Diagnostic Conditions

Code	Meaning	Addinfo
1	permanent system error	(unspecified)
2	temporary system error	(unspecified)
3	unsupported search	(unspecified)
4	Terms only exclusion (stop) words	(unspecified)
5	Too many argument words	(unspecified)
6	Too many Boolean operators	(unspecified)
7	Too many truncated words	(unspecified)
8	Too many incomplete subfields	(unspecified)
9	Truncated words too short	(unspecified)
10	Invalid format for record number (search term)	(unspecified)
11	Too many characters in search statement	(unspecified)
12	Too many records retrieved	(unspecified)

13	Present request out-of-range	(unspecified)
14	System error in presenting records	(unspecified)
15	Record not authorized to be sent intersystem	(unspecified)
16	Record exceeds Preferred-message-size	(unspecified)
17	Record exceeds Exceptional-record-size	(unspecified)
18	Result set not supported as a search term	(unspecified)
19	Only *single* result set as search term supported	(unspecified)
20	Only *AND*ing of a *single* result set as search term	(unspecified)
21	Result set exists and replace indicator off	(unspecified)
22	Result set naming not supported	(unspecified)
23	Specified combination of databases not supported	(unspecified)
24	Element set names not supported	(unspecified)
25	Specified element set name not valid for specified database	(unspecified)
26	Only generic form of element set name supported	(unspecified)
27	Result set no longer exists - unilaterally deleted by target	(unspecified)
28	Result set is in use	(unspecified)
29	One of the specified databases is locked	(unspecified)
30	Specified result set does not exist	(unspecified)
31	Resources exhausted - no results available	(unspecified)
32	Resources exhausted - unpredictable partial results available	(unspecified)
33	Resources exhausted - valid subset of results available	(unspecified)
100	(unspecified) error	(unspecified)
101	Access-control failure	(unspecified)
102	Challenge required, could not be issued - operation terminated	(unspecified)
103	Challenge required, could not be issued - record not included	(unspecified)
104	Challenge failed - record not included	(unspecified)
105	Terminated at origin request	(unspecified)
106	No abstract syntaxes agreed to for this record	(unspecified)
107	Query type not supported	(unspecified)
108	Malformed query	(unspecified)
109	Database unavailable	database name
110	Operator unsupported	operator
111	Too many databases specified	maximum
112	Too many result sets created	maximum
113	Unsupported attribute type	type
114	Unsupported Use attribute	value
115	Unsupported *term* value for Use attribute	term
116	Use attribute required but not supplied	(unspecified)
117	Unsupported Relation attribute	value
118	Unsupported Structure attribute	value
119	Unsupported Position attribute	value
120	Unsupported Truncation attribute	value
121	Unsupported Attribute Set	oid
122	Unsupported Completeness attribute	value
123	Unsupported attribute combination	(unspecified)
124	Unsupported coded value for term	value
125	Malformed search term	(unspecified)
126	Illegal term value for attribute	term
127	Unparsable format for unnormalized value	value
128	Illegal result set name	name
129	Proximity search of sets not supported	(unspecified)
130	Illegal result set in proximity search	result set name
131	Unsupported proximity relation	value
132	Unsupported proximity unit code	value
201	Proximity not supported with this attribute combination	attribute list
202	Unsupported distance for proximity	distance
203	Ordered flag not supported for proximity	(unspecified)
205	Only zero step size supported for Scan	(unspecified)

206	Specified step size not supported for Scan	step size
207	Cannot sort according to sequence	sequence
208	No result set name supplied on Sort	(unspecified)
209	Generic sort not supported (database-specific sort only supported)	(unspecified)
210	Database specific sort not supported	(unspecified)
211	Too many sort keys	number
212	Duplicate sort keys	key
213	Unsupported missing data action	value
214	Illegal sort relation	relation
215	Illegal case value	value
216	Illegal missing data action	value
217	Segmentation: Cannot guarantee records will fit in specified segments	(unspecified)
218	ES: Package name already in use	name
219	ES: no such package, on modify/delete	name
220	ES: quota exceeded	(unspecified)
221	ES: extended service type not supported	type
222	ES: permission denied on ES - id not authorized	(unspecified)
223	ES: permission denied on ES - cannot modify or delete	(unspecified)
224	ES: immediate execution failed	(unspecified)
225	ES: immediate execution not supported for this service	(unspecified)
226	ES: immediate execution not supported for these parameters	(unspecified)
227	No data available in requested record syntax	(unspecified)
228	Scan: malformed scan	(unspecified)
229	Term type not supported	type
230	Sort: too many input results	max
231	Sort: incompatible record formats	(unspecified)
232	Scan: term list not supported	alternative term list
233	Scan: unsupported value of position-in-response	value
234	Too many index terms processed	number of terms
235	Database does not exist	database name
236	Access to specified database denied	database name
237	Sort: illegal sort	(unspecified)
238	Record not available in requested syntax	alternative suggested syntax(es)
239	Record syntax not supported	syntax
240	Scan: Resources exhausted looking for satisfying terms	(unspecified)
241	Scan: Beginning or end of term list	(unspecified)
242	Segmentation: max-segment-size too small to segment record	smallest acceptable size
243	Present: additional-ranges parameter not supported	(unspecified)
244	Present: comp-spec parameter not supported	(unspecified)
245	Type-1 query: restriction ('resultAttr') operand not supported	(unspecified)
246	Type-1 query: 'complex' attributeValue not supported	(unspecified)
247	Type-1 query: 'attributeSet' as part of AttributeElement not supported	(unspecified)

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,
DefaultDiagFormat, InternationalString FROM Z39-50-APDU-1995;

DiagnosticFormat ::=  SEQUENCE OF SEQUENCE{
                  diagnostic  [1] CHOICE{
                                    defaultDiagRec        [1] IMPLICIT DefaultDiagFormat,
                                    explicitDiagnostic    [2] DiagFormat} OPTIONAL,
                  message     [2] IMPLICIT InternationalString 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 InternationalString 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 OPTIONAL
                              -- If occurs, term is illegal or not supported, for attribute value,
                              -- if value occurs; otherwise, for type.
                                                                 },

attCombo           [1005]   IMPLICIT SEQUENCE{ -- attribute combination not supported
                              unsupportedCombination   [1] IMPLICIT AttributeList,
                              recommendedAlternatives  [2] IMPLICIT SEQUENCE OF AttributeList OPTIONAL},

term               [1006]   IMPLICIT SEQUENCE{
                            problem        [1] IMPLICIT INTEGER{
                                                  codedValue    (1),
                                                  unparsable    (2),
                                                  tooShort      (3),
                                   type           (4)} OPTIONAL,
                  term         [2] Term},

proximity        [1007] CHOICE{       -- proximity diagnostics:
                  resultSets   [1]    IMPLICIT NULL,               -- proximity between sets not supported
                  badSet       [2]    IMPLICIT InternationalString, -- 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
                  termList1          [3] IMPLICIT NULL,   -- term list not supported (no alternative supplied)
                  termList2          [4] IMPLICIT SEQUENCE OF AttributeList,
                                                         -- term list not supported (alternatives supplied)
                  posInResponse      [5] IMPLICIT INTEGER{  --value of positionInResponse not supported
                                         mustBeOne            (1),
                                         mustBePositive       (2),
                                         mustBeNonNegative    (3),
                                         other                (4)},
                  resources        [6] IMPLICIT NULL,        -- resources exhausted looking for satisfying terms
                  endOfList        [7] IMPLICIT NULL         -- beginning or end of term list
                               },
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.
                    incompatible    [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),     -- too many sort keys
                                           duplicate  (2)},    -- duplicate sort keys
                    action          [8] IMPLICIT NULL,         -- unsupported 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 InternationalString,
                                                               -- one or more of the input result sets too large to sort
              aggregateTooLarge     [11] IMPLICIT NULL         --  aggregate result set too large
                                              },

segmentation     [1010] CHOICE{
                  segmentCount      [0]    IMPLICIT NULL,
                                              -- Cannot guarantee record will fit within max segments. Target
                                              -- suggests that origin try again to retrieve record, without
                                              -- including max-segment-count.
                  segmentSize       [1]    IMPLICIT INTEGER
                                              -- record cannot be segmented into fragments such that the
                                              -- largest will fit within max segment size specified. Target
                                              -- supplies (in bytes) the smallest acceptable value of Max-
```

```
                                        -- segment-size to retrieve the record.
                                    },

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, because:
                                  id             (1),     -- id not authorized, or
                                  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.
                                       }},

accessCtrl       [1012] CHOICE{
                      noUser       [1] IMPLICIT NULL,      -- no user to display challenge to
                      refused      [2] IMPLICIT NULL,      -- access control information refused by user
                      simple       [3] IMPLICIT NULL,      -- only simple form supported (target used
                                                           -- externally defined)
                      oid          [4] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER,
                                                           -- oid not supported (origin supplies alternative
                                                           -- suggested oids)
                      alternative  [5] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER,
                                                           -- origin insists that target use an alternative
                                                           -- challenge for this data (e.g., stronger
                                                           -- authentication or stronger Access control). The
                                                           -- origin supplies suggested alternative oids.
                      pwdInv       [6] IMPLICIT NULL,      -- password invalid
                      pwdExp       [7] IMPLICIT NULL       -- password expired
                           },

recordSyntax   [1013] IMPLICIT SEQUENCE{ -- record cannot be transferred in requested syntax
                      unsupportedSyntax        [1] IMPLICIT OBJECT IDENTIFIER,
                      suggestedAlternatives    [2] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL}
}

END
```

Annex 5
REC: Record Syntaxes
(Normative)

This standard registers the following object identifiers for record syntaxes:

Object identifiers assigned for bibliographic syntaxes, not *described via 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 }**
Ibermarc	**{Z39-50-recordSyntax**	**21 }**

Note: The following transfer syntax (see ISO 2709) 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 via 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-1	**{Z39-50-recordSyntax 105}**	(See REC.5)
Extended Services	**{Z39-50-recordSyntax 106}**	(See REC.6)

Note: The following transfer syntax (see ISO 8825) 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 AttributeSetId, Term, OtherInformation, DatabaseName, ElementSetName, IntUnit, Unit,
StringOrNumeric, Specification, InternationalString, AttributeList, AttributeElement FROM Z39-50-APDU-1995;

Explain-Record ::= CHOICE{
                    -- Each of these may be used as search term  when Use attribute is ‘explain-category’.
  targetInfo                    [0]     IMPLICIT TargetInfo,
  databaseInfo                  [1]     IMPLICIT DatabaseInfo,
  schemaInfo                    [2]     IMPLICIT SchemaInfo,
  tagSetInfo                    [3]     IMPLICIT TagSetInfo,
  recordSyntaxInfo              [4]     IMPLICIT RecordSyntaxInfo,
  attributeSetInfo              [5]     IMPLICIT AttributeSetInfo,
  termListInfo                  [6]     IMPLICIT TermListInfo,
  extendedServicesInfo          [7]     IMPLICIT ExtendedServicesInfo,
  attributeDetails              [8]     IMPLICIT AttributeDetails,
  termListDetails               [9]     IMPLICIT TermListDetails,
  elementSetDetails             [10]    IMPLICIT ElementSetDetails,
  retrievalRecordDetails        [11]    IMPLICIT RetrievalRecordDetails,
  sortDetails                   [12]    IMPLICIT SortDetails,
  processing                    [13]    IMPLICIT ProcessingInformation,
  variants                      [14]    IMPLICIT VariantSetInfo,
  units                         [15]    IMPLICIT UnitInfo,
  categoryList                  [100]   IMPLICIT CategoryList}

-- Element set name ‘B’ (brief)  retrieves:
--           -  ‘commonInfo’ (except for otherInfo within commonInfo)
--           -  key elements
--           -  other elements designated as ‘non-key brief elements’
-- Esn ‘description’ retrieves brief elements as well as ‘description’, and specific additional descriptive
-- elements if designated.
-- Element set name ‘F’ (full) retrieves all of the above, as well as those designated as “non-brief elements.” Some
-- elements designated as OPTIONAL may be mandatory in full records, and are so identified. (Note that all
-- elements that are not part of the brief element set must be designated as OPTIONAL in the ASN.1, otherwise
-- it would be illegal to omit them.)
-- Other esns are defined (below) as needed.

— - - - - - - - - - - -  Info Records
  —   Info records are mainly for software consumption
  —   They describe individual entities within the target system:
  —   The target itself
  —   Individual databases
  —   Schemas
  —   Tag sets
  —   Record syntaxes
  —   Attribute sets
  —   Term lists
  —   Extended services
 —           The information about each Schema, Tag Set, Record Syntax and Attribute Set should
  -- match the universal definitions of these items. The only exception is that a target may omit any
  -- items it doesn’t support, for example the description of the bib-1 attribute set may omit attributes
  -- that the target does not support under any circumstances.
  —           Databases that may be searched together can be listed in the dbCombinations element of the TargetInfo record.
TargetInfo ::= SEQUENCE {
  commonInfo        [0]       IMPLICIT CommonInfo OPTIONAL,
    — Key elements follow:
  name              [1]       IMPLICIT InternationalString,
    — Non-key brief elements follow:
  recent-news       [2]       IMPLICIT HumanString OPTIONAL,
  icon                  [3]    IMPLICIT IconObject OPTIONAL,
  namedResultSets       [4]    IMPLICIT BOOLEAN,
  multipleDBsearch      [5]    IMPLICIT BOOLEAN,
  maxResultSets         [6]    IMPLICIT INTEGER OPTIONAL,
  maxResultSize         [7]    IMPLICIT INTEGER OPTIONAL,
  maxTerms              [8]    IMPLICIT INTEGER OPTIONAL,
  timeoutInterval       [9]    IMPLICIT IntUnit OPTIONAL,
  welcomeMessage        [10]   IMPLICIT HumanString OPTIONAL,
     — non-brief elements follow:
     —  'description' esn retrieves the following two (as well as brief):
  contactInfo           [11]   IMPLICIT ContactInfo OPTIONAL,
  description           [12]   IMPLICIT HumanString OPTIONAL,
  nicknames             [13]   IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
  usage-restrictions    [14]   IMPLICIT HumanString OPTIONAL,
  paymentAddr           [15]   IMPLICIT HumanString OPTIONAL,
  hours                 [16]   IMPLICIT HumanString OPTIONAL,
  dbCombinations        [17]   IMPLICIT SEQUENCE OF DatabaseList OPTIONAL,
  addresses             [18]   IMPLICIT SEQUENCE OF NetworkAddress OPTIONAL,
  languages             [101]  IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
                                    -- Languages supported for message strings.  Each is a three-character
                                    -- language code from Z39.53-1994.
-- characterSets        [102]  this tag reserved for "character sets supported for name and message strings."
  — commonAccessInfo elements list objects the target supports. All objects listed in
  — AccessInfo for any individual database should also be listed here.
  commonAccessInfo  [19]    IMPLICIT AccessInfo OPTIONAL}

DatabaseInfo ::= SEQUENCE {
                    -- A target may provide "virtual databases" that are combinations of individual databases. These
                    -- databases are indicated by the presence of subDbs in the combination database's DatabaseDescription.
  commonInfo                 [0]       IMPLICIT CommonInfo OPTIONAL,
    — Key elements follow:
  name                       [1]       IMPLICIT DatabaseName,
    — Non-key brief elements follow:
  explainDatabase            [2]       IMPLICIT NULL OPTIONAL,
                             -- If present, this database is the Explain database, or an Explain database
                             -- for a different server, possibly on a different host. The means by which
                             -- that server may be accessed is not addressed by this standard. One
                             -- suggested possibility is an implementor agreement whereby the
                             -- database name is a url which may be used to connect to the server.
  nicknames                  [3]       IMPLICIT SEQUENCE OF DatabaseName OPTIONAL,
  icon                       [4]       IMPLICIT IconObject OPTIONAL,
  user-fee                   [5]       IMPLICIT BOOLEAN,
  available                  [6]       IMPLICIT BOOLEAN,
  titleString                [7]       IMPLICIT HumanString OPTIONAL,
    — Non-brief elements follow:
  keywords                   [8]       IMPLICIT SEQUENCE OF HumanString OPTIONAL,
  description                [9]       IMPLICIT HumanString OPTIONAL,
  associatedDbs              [10]      IMPLICIT DatabaseList OPTIONAL,
                             -- databases that may be searched in combination with this one

  subDbs                     [11]      IMPLICIT DatabaseList OPTIONAL,
                             -- When present, this database is a composite representing the combined
                             -- databases 'subDbs'. The individual subDbs are also available.
  disclaimers                [12]      IMPLICIT HumanString OPTIONAL,
  news                       [13]      IMPLICIT HumanString OPTIONAL,
  recordCount                [14]      CHOICE {
                                       actualNumber       [0] IMPLICIT INTEGER,
                                       approxNumber       [1] IMPLICIT INTEGER} OPTIONAL,
  defaultOrder              [15]     IMPLICIT HumanString OPTIONAL,
  avRecordSize              [16]     IMPLICIT INTEGER OPTIONAL,
  maxRecordSize             [17]     IMPLICIT INTEGER OPTIONAL,
  hours                     [18]     IMPLICIT HumanString OPTIONAL,
  bestTime                  [19]     IMPLICIT HumanString OPTIONAL,
  lastUpdate                [20]     IMPLICIT GeneralizedTime OPTIONAL,
  updateInterval            [21]     IMPLICIT IntUnit OPTIONAL,
  coverage                  [22]     IMPLICIT HumanString OPTIONAL,
  proprietary               [23]     IMPLICIT BOOLEAN OPTIONAL, -- mandatory in full record
  copyrightText             [24]     IMPLICIT HumanString OPTIONAL,
  copyrightNotice           [25]     IMPLICIT HumanString OPTIONAL,
  producerContactInfo       [26]     IMPLICIT ContactInfo OPTIONAL,
  supplierContactInfo       [27]     IMPLICIT ContactInfo OPTIONAL,
  submissionContactInfo     [28]     IMPLICIT ContactInfo OPTIONAL,
    — accessInfo lists items connected with the database. All listed items should be in the target's AccessInfo.
  accessInfo                [29]     IMPLICIT AccessInfo OPTIONAL}

SchemaInfo ::= SEQUENCE {
  commonInfo              [0]    IMPLICIT CommonInfo OPTIONAL,
    — Key elements follow:
  schema                  [1]    IMPLICIT OBJECT IDENTIFIER,
    — Non-key brief elements follow:
  name                    [2]    IMPLICIT InternationalString,
    — Non-brief elements follow:
  description             [3]    IMPLICIT HumanString OPTIONAL,
  tagTypeMapping          [4]    IMPLICIT SEQUENCE OF SEQUENCE {
                                 tagType              [0] IMPLICIT INTEGER,
                                 tagSet               [1] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
                                 -- If tagSet is omitted, then this tagType is for a tagSet locally defined
                                 -- within the schema that cannot be referenced by another schema.
                                    defaultTagType    [2] IMPLICIT NULL OPTIONAL
                                                          } OPTIONAL,
  recordStructure         [5]    IMPLICIT SEQUENCE OF ElementInfo OPTIONAL}

     —  ElementInfo referenced in SchemaInfo and RecordSyntaxInfo
            ElementInfo ::= SEQUENCE {
               elementName        [1] IMPLICIT InternationalString,
               elementTagPath[2] IMPLICIT Path,
               dataType           [3] ElementDataType OPTIONAL, — If omitted, not specified.
               required           [4] IMPLICIT BOOLEAN,
               repeatable         [5] IMPLICIT BOOLEAN,
               description        [6] IMPLICIT HumanString OPTIONAL}

     — Path is referenced by ElementInfo as well as PerElementDetails
                  Path ::= SEQUENCE OF SEQUENCE{
                  tagType      [1] IMPLICIT INTEGER,
                  tagValue     [2] StringOrNumeric}
               ElementDataType ::= CHOICE{
                  primitive    [0] IMPLICIT PrimitiveDataType,
                  structured   [1] IMPLICIT SEQUENCE OF ElementInfo}
               PrimitiveDataType ::= INTEGER{
                  octetString          (0),
                  numeric              (1),
                  date                 (2),
                  external             (3),
                  string               (4),
                  trueOrFalse          (5),
                  oid                  (6),
                        intUnit              (7),
                           empty             (8),
                        noneOfTheAbove       (100) — see 'description'
                              }

TagSetInfo ::= SEQUENCE {
 commonInfo             [0]    IMPLICIT CommonInfo OPTIONAL,
      — Key elements follow:
 tagSet                 [1]    IMPLICIT OBJECT IDENTIFIER,
      — non-key brief elements follow:
 name                   [2]    IMPLICIT InternationalString,
      — non-brief elements follow:
 description            [3]    IMPLICIT HumanString OPTIONAL,
 elements               [4]    IMPLICIT SEQUENCE OF SEQUENCE {
                                  elementname   [1] I   MPLICIT InternationalString,
                                  nicknames     [2]     IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
                                  elementTag    [3]     StringOrNumeric,
                                  description   [4]     IMPLICIT HumanString OPTIONAL,
                                  dataType      [5]     PrimitiveDataType OPTIONAL,
                               -- If the data type is expected to be structured, that is described in the schema info,
                               -- and datatype is omitted here.
                                  otherTagInfo        OtherInformation OPTIONAL} OPTIONAL}


RecordSyntaxInfo ::= SEQUENCE {
 commonInfo             [0]    IMPLICIT CommonInfo OPTIONAL,
      — Key elements follow:
 recordSyntax           [1]    IMPLICIT OBJECT IDENTIFIER,
      — Non-key brief elements follow:
 name                   [2]    IMPLICIT InternationalString,
      — non-brief elements follow:
 transferSyntaxes       [3]    IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
 description            [4]    IMPLICIT HumanString OPTIONAL,
 asn1Module             [5]    IMPLICIT InternationalString OPTIONAL,
 abstractStructure      [6]    IMPLICIT SEQUENCE OF ElementInfo OPTIONAL
                                     — Omitting abstractStructure only means target isn't using
                                     — Explain to describe the structure, not that there is no structure.
                                  }

AttributeSetInfo ::= SEQUENCE {
 commonInfo             [0]    IMPLICIT CommonInfo OPTIONAL,
      — Key elements follow:
 attributeSet           [1]    IMPLICIT AttributeSetId,
      — non-key brief elements follow:
 name                   [2]    IMPLICIT InternationalString,
      — non-brief elements follow:
 attributes             [3]    IMPLICIT SEQUENCE OF AttributeType OPTIONAL,
                                     -- mandatory in full record
 description            [4]    IMPLICIT HumanString OPTIONAL}
-- AttributeType referenced in AttributeSetInfo
      AttributeType ::= SEQUENCE {
                        name               [0] IMPLICIT InternationalString OPTIONAL,
                        description        [1] IMPLICIT HumanString OPTIONAL,
                        attributeType      [2] IMPLICIT INTEGER,
                        attributeValues    [3] IMPLICIT SEQUENCE OF AttributeDescription}
              AttributeDescription ::= SEQUENCE {
                        name                  [0] IMPLICIT InternationalString OPTIONAL,
                        description           [1] IMPLICIT HumanString OPTIONAL,
                        attributeValue          [2] StringOrNumeric,
                        equivalentAttributes    [3] IMPLICIT SEQUENCE OF StringOrNumeric OPTIONAL
                                   -- each is an occurrence of 'attributeValue' from AttributeDescription for a
                                   -- different attribute. Equivalences listed here should be derived from the
                                   -- attribute set definition, not from a particular server's behavior.
                                          }

TermListInfo ::= SEQUENCE{
 commonInfo          [0] IMPLICIT CommonInfo OPTIONAL,
      — Key elements follow:
 databaseName        [1] IMPLICIT DatabaseName,
      — Non-key brief elements follow:
 termLists           [2] IMPLICIT SEQUENCE OF SEQUENCE{
                          name        [1] IMPLICIT InternationalString,
                          title       [2] IMPLICIT HumanString OPTIONAL,
                               --Title is for users to see and can differ by language. Name, on the
                                      -- other hand, is typically a short string not necessarily meant to be
                                      -- human-readable, and not variable by language.
                          searchCost  [3] IMPLICIT INTEGER {
                                      optimized    (0),-- The attribute (or combination) associated
                                                   -- with this list will do fast searches.
                                      normal       (1),-- The attribute (combination) will work as
                                                   -- expected. So there's probably an index for the
                                                   -- attribute (combination) or some similar
                                                   -- mechanism.
                                      expensive    (2),-- Can use the attribute (combination), but it
                                                   -- might not provide satisfactory results.
                                                   -- Probably there is no index, or post-
                                                   -- processing of records is required.
                                      filter       (3)-- can't search with this attribute (combination) alone.
                                                   }  OPTIONAL,
                          scannable   [4]    IMPLICIT BOOLEAN, -- 'true' means this list can be scanned.
                          broader     [5]    IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
                          narrower    [6]    IMPLICIT SEQUENCE OF InternationalString OPTIONAL
                                      --  broader and narrower list alternative term lists related to this one.
                                      -- The term lists so listed should also be in this termLists structure.
                                                       }
      -- no non-brief elements
                          }

ExtendedServicesInfo ::= SEQUENCE {
 commonInfo               [0]  IMPLICIT CommonInfo OPTIONAL,
      — Key elements follow:
 type                     [1]  IMPLICIT OBJECT IDENTIFIER,
      — Non-key brief elements follow:
 name                     [2]  IMPLICIT InternationalString OPTIONAL,
                               -- should be supplied if privateType is 'true'
 privateType              [3]  IMPLICIT BOOLEAN,
 restrictionsApply        [5]  IMPLICIT BOOLEAN,  -- if 'true' see 'description'
 feeApply                 [6]  IMPLICIT BOOLEAN,  -- if 'true' see 'description'
 available                [7]  IMPLICIT BOOLEAN,
 retentionSupported       [8]  IMPLICIT BOOLEAN,
 waitAction               [9]  IMPLICIT INTEGER{
                                   waitSupported       (1),
                                   waitAlways          (2),
                                   waitNotSupported    (3),
                                   depends             (4),
                                   notSaying           (5)},
  — non-brief elements follow:
-- To get brief plus ‘description’ use esn ‘description’
  description                  [10]  IMPLICIT HumanString OPTIONAL,
  -- to get above elements and ‘specificExplain’ use esn ‘specificExplain’
  specificExplain              [11]  IMPLICIT EXTERNAL OPTIONAL,
                                     -- Use oid of specific ES, and select choice [3] ‘explain’. Format
                                     -- to be developed in conjunction with the specific ES definition.
-- to get all elements except ‘specificExplain’, use esn ‘asn’
  esASN                        [12]  IMPLICIT InternationalString OPTIONAL -- the ASN.1 for this ES
                                       }


— - - - - - - - - - - - Detail records
 --The detail records describe relationships among entities supported by the target. RetrievalRecordDetails describes
 -- the way that schema elements are mapped into record elements. This mapping may be different for each
 -- combination of database, schema, record syntax. The per-element details describe the default mapping.
 -- Origin-request re-tagging can change that mapping. When multiple databases are listed in a databaseNames
 -- element, the record applies equally to all of the listed databases.  This is unrelated to searching the databases
 -- together. AttributeDetails describes how databases can be searched. Each supported attribute is listed, and the
 -- allowable combinations can be described.
AttributeDetails ::= SEQUENCE {
  commonInfo                   [0] IMPLICIT CommonInfo OPTIONAL,
        — Key elements follow:
  databaseName                 [1] IMPLICIT DatabaseName,
        — Non-brief elements follow:
  attributesBySet              [2] IMPLICIT SEQUENCE OF AttributeSetDetails OPTIONAL,
                                         -- mandatory in full record
  attributeCombinations        [3] IMPLICIT AttributeCombinations OPTIONAL}

-- AttributeSetDetails referenced by AttributeDetails
        AttributeSetDetails ::= SEQUENCE {
                attributeSet            [0] IMPLICIT AttributeSetId,
                attributesByType        [1] IMPLICIT SEQUENCE OF AttributeTypeDetails }

          AttributeTypeDetails ::= SEQUENCE {
                attributeType           [0] IMPLICIT INTEGER,
                defaultIfOmitted        [1] IMPLICIT OmittedAttributeInterpretation OPTIONAL,
                attributeValues         [2] IMPLICIT SEQUENCE OF AttributeValue OPTIONAL }
                                        -- If no attributeValues are supplied, all values of this type are fully
                                        -- supported, and the descriptions in AttributeSetInfo are adequate.

            OmittedAttributeInterpretation ::= SEQUENCE {
                defaultValue                [0] StringOrNumeric OPTIONAL,
                                            -- A default value is listed if that’s how the server works
                defaultDescription          [1] IMPLICIT HumanString OPTIONAL }
                                            -- The human-readable description should generally be provided.
                                            -- It is legal for both default elements to be missing, which
                                            -- means that the target will allow the attribute type to be
                                            -- omitted, but isn’t saying what it will do.

            AttributeValue ::= SEQUENCE {
              value                     [0]  StringOrNumeric,
              description               [1]  IMPLICIT HumanString OPTIONAL,
              subAttributes             [2]  IMPLICIT SEQUENCE OF StringOrNumeric OPTIONAL,
              superAttributes           [3]  IMPLICIT SEQUENCE OF StringOrNumeric OPTIONAL,
              partialSupport            [4]  IMPLICIT NULL OPTIONAL }
            -- partialSupport indicates that an attributeValue is accepted, but may not be processed in the
            -- “expected” way. One important reason for this is composite databases: in this case partialSupport
            -- may indicate that only some of the subDbs support the attribute, and others ignore it.
```

```
TermListDetails ::= SEQUENCE{  -- one for each termList in TermListInfo
  commonInfo            [0]  IMPLICIT CommonInfo OPTIONAL,
          — Key elements follow:
  termListName          [1]  IMPLICIT InternationalString,
          — Non-key elements (all non-brief) follow:
  description           [2]  IMPLICIT HumanString OPTIONAL,
  attributes            [3]  IMPLICIT AttributeCombinations OPTIONAL,
                                    — Pattern for attributes that hit this list. Mandatory in full record
  scanInfo              [4]  IMPLICIT SEQUENCE {
                                    maxStepSize             [0] IMPLICIT INTEGER OPTIONAL,
                                    collatingSequence       [1] IMPLICIT HumanString OPTIONAL,
                                    increasing              [2] IMPLICIT BOOLEAN OPTIONAL} OPTIONAL,
                        -- Occurs only if list is scannable. If list is scannable and if scanInfo is omitted,
                        -- target doesn't consider these important.
  estNumberTerms        [5]  IMPLICIT INTEGER OPTIONAL,
  sampleTerms           [6]  IMPLICIT SEQUENCE OF Term OPTIONAL}

ElementSetDetails ::= SEQUENCE {
          -- ElementSetDetails describes the way that database records are mapped to record elements. This
          -- mapping may be different for each combination of database name and element set. The database record
          -- description is a schema, which may be private to the target. The schema's abstract record structure
          -- and tag sets provide the vocabulary for discussing record content; their presence in the Explain
          -- database does not imply support for complex retrieval specification.

  commonInfo            [0]    IMPLICIT CommonInfo OPTIONAL,
          — Key elements follow:
  databaseName          [1]    IMPLICIT DatabaseName,
  elementSetName        [2]    IMPLICIT ElementSetName,
  recordSyntax          [3]    IMPLICIT OBJECT IDENTIFIER,
          — Non-key Brief elements follow:
  schema                [4]    IMPLICIT OBJECT IDENTIFIER,
          — Non-brief elements follow:
  description           [5]    IMPLICIT HumanString OPTIONAL,
  detailsPerElement     [6]    IMPLICIT SEQUENCE OF PerElementDetails OPTIONAL -- mandatory in full record
                                  }

RetrievalRecordDetails ::= SEQUENCE {
  commonInfo            [0]    IMPLICIT CommonInfo OPTIONAL,
           — Key elements follow:
  databaseName          [1]    IMPLICIT DatabaseName,
  schema                [2]    IMPLICIT OBJECT IDENTIFIER,
  recordSyntax          [3]    IMPLICIT OBJECT IDENTIFIER,
          — Non-brief elements follow:
  description           [4]    IMPLICIT HumanString OPTIONAL,
  detailsPerElement     [5]    IMPLICIT SEQUENCE OF PerElementDetails OPTIONAL
                               -- mandatory in full record
                               }

-- PerElementDetails is referenced in RetrievalRecordDetails and ElementSetDetails.
          PerElementDetails ::= SEQUENCE {
              name            [0]    IMPLICIT InternationalString OPTIONAL,
                                           — If the name is omitted, the record syntax's name for this element
                                     -- is appropriate.
              recordTag       [1]    IMPLICIT RecordTag OPTIONAL,
                                           — The record tag may be omitted if tags are inappropriate for the record
                                     -- syntax, or if the origin can be expected to know it for some other reason.
              schemaTags      [2]    IMPLICIT SEQUENCE OF Path OPTIONAL,
                                           — The information from the listed schema elements is combined
                                     -- in some way to produce the data sent in the listed record tag. The
                                     -- 'contents' element below may describe the logic used.
              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,
                                    — ‘required’ really means that target will always supply the element.
        description       [12]  IMPLICIT HumanString OPTIONAL,
        contents          [13]  IMPLICIT HumanString OPTIONAL,
        billingInfo       [14]  IMPLICIT HumanString OPTIONAL,
        restrictions      [15]  IMPLICIT HumanString OPTIONAL,
        alternateNames    [16]  IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
        genericNames      [17]  IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
        searchAccess      [18]  IMPLICIT AttributeCombinations OPTIONAL }

     -- RecordTag referenced in PerElementDetails above
              RecordTag ::= SEQUENCE {
                  qualifier     [0] StringOrNumeric OPTIONAL,
                                          — E.g., tag set for GRS-1
                  tagValue      [1] StringOrNumeric}

SortDetails ::= SEQUENCE {
 commonInfo          [0]  IMPLICIT CommonInfo OPTIONAL,
          — Key elements follow:
 databaseName        [1]  IMPLICIT DatabaseName,
          — No non-key brief elements
          — Non-brief elements follow:
 sortKeys            [2]  IMPLICIT SEQUENCE OF SortKeyDetails OPTIONAL
                              -- mandatory in full record
                              }
     SortKeyDetails ::= SEQUENCE {
        description              [0]  IMPLICIT HumanString OPTIONAL,
        elementSpecifications    [1]  IMPLICIT SEQUENCE OF Specification OPTIONAL,
                                      — each specification is a way of specifying this same sort key
        attributeSpecifications  [2]  IMPLICIT AttributeCombinations OPTIONAL,
                                      — each combination is a way of specifying this same sort key
        sortType                 [3]  CHOICE {
                                            character     [0]  IMPLICIT NULL,
                                            numeric       [1]  IMPLICIT NULL,
                                            structured    [2]  IMPLICIT HumanString} OPTIONAL,
        caseSensitivity          [4]  IMPLICIT INTEGER {
                                            always       (0),  -- always case-sensitive
                                            never        (1),  -- never case-sensitive
                                            default-yes  (2),  -- case-sensitivity is as specified on request, and if not
                                                         -- specified, case-sensitive.
                                            default-no   (3)}  -- case-sensitivity is as specified on request, and if not
                                                         -- specified, not case-sensitive.
                                                          OPTIONAL}

ProcessingInformation ::= SEQUENCE{
 commonInfo          [0] IMPLICIT CommonInfo OPTIONAL,
      — Key elements follow:
 databaseName        [1] IMPLICIT DatabaseName,



 processingContext   [2] IMPLICIT INTEGER {
                              access                (0),    -- e.g., choosing databases
                              search                (1),    -- e.g., “search strategies” or search forms
                              retrieval             (2),    -- e.g., recommended element combinations
                              record-presentation   (3),    -- display of retrieved records
                              record-handling       (4)     -- handling (e.g., saving) of retrieved records
                  },
 name                [3] IMPLICIT InternationalString,
  oid                        [4]  IMPLICIT OBJECT IDENTIFIER,
            — No non-key brief elements
            — Non-brief elements follow:
  description                [5]  IMPLICIT HumanString OPTIONAL,
                                  -- use element set name ‘description’ to retrieve all except instructions.
  instructions               [6]  IMPLICIT EXTERNAL OPTIONAL -- mandatory in full record

                                         }


      VariantSetInfo ::= SEQUENCE {
-- A record in this category describes a variant set definition, i.e., classes, types, and values, for a specific
-- variant set definition supported by the target. Support by the target of a particular variant set definition
-- does not imply that the definition is supported for any specific database or element.
  commonInfo     [0] IMPLICIT CommonInfo OPTIONAL,
            — Key elements follow:
  variantSet     [1] IMPLICIT OBJECT IDENTIFIER,
            — Non-key brief elements follow:
  name           [2] IMPLICIT InternationalString,
            — Non-brief elements follow:
  variants       [3] IMPLICIT SEQUENCE OF VariantClass OPTIONAL
                           — mandatory in full record
                            }

            — Subsidiary structures for VariantSetInfo
                VariantClass ::= SEQUENCE {
                    name              [0] IMPLICIT InternationalString OPTIONAL,
                    description       [1] IMPLICIT HumanString OPTIONAL,
                    variantClass      [2] IMPLICIT INTEGER,
                    variantTypes      [3] IMPLICIT SEQUENCE OF VariantType}
                VariantType ::= SEQUENCE {
                    name              [0] IMPLICIT InternationalString OPTIONAL,
                    description       [1] IMPLICIT HumanString OPTIONAL,
                    variantType       [2] IMPLICIT INTEGER,
                    variantValue      [3] IMPLICIT VariantValue OPTIONAL}
                VariantValue ::= SEQUENCE {
                    dataType          [0] PrimitiveDataType,
                    values            [1] ValueSet OPTIONAL }

                ValueSet ::= CHOICE {
                    range          [0] IMPLICIT ValueRange,
                    enumerated     [1] IMPLICIT SEQUENCE OF ValueDescription }
                ValueRange ::= SEQUENCE {
                        — At last one the following must be supplied, both may be supplied.
                    lower          [0] ValueDescription OPTIONAL,
                    upper          [1] ValueDescription OPTIONAL }

                ValueDescription ::= CHOICE{
                    integer           INTEGER,
                    string            InternationalString,
                    octets            OCTET STRING,
                    oid               OBJECT IDENTIFIER,
                    unit              [1]  IMPLICIT Unit,
                    valueAndUnit      [2]  IMPLICIT IntUnit
                        — oid and unit can’t be used in a ValueRange
                                 }

UnitInfo ::= SEQUENCE {
  commonInfo                 [0] IMPLICIT CommonInfo OPTIONAL,
            — Key elements follow:
  unitSystem                 [1] IMPLICIT InternationalString,
            — No non-key brief elements
            — Non-brief elements follow:
  description                  [2] IMPLICIT HumanString OPTIONAL,
  units                        [3] IMPLICIT SEQUENCE OF UnitType OPTIONAL
                                          — mandatory in full record
                                }

            — Subsidiary structures for UnitInfo
                UnitType ::= SEQUENCE {
                    name            [0] IMPLICIT InternationalString OPTIONAL,
                    description     [1] IMPLICIT HumanString OPTIONAL,
                    unitType        [2] StringOrNumeric,
                    units           [3] IMPLICIT SEQUENCE OF Units}

                Units ::= SEQUENCE {
                    name            [0] IMPLICIT InternationalString OPTIONAL,
                    description     [1] IMPLICIT HumanString OPTIONAL,
                    unit            [2] StringOrNumeric}

CategoryList ::= SEQUENCE {
  commonInfo    [0] IMPLICIT CommonInfo OPTIONAL,
            — Only one record expected per Explain database. All elements appear in brief presentation.
  categories    [1] IMPLICIT SEQUENCE OF CategoryInfo }
                CategoryInfo ::= SEQUENCE {
                category            [1] IMPLICIT InternationalString,
                originalCategory    [2] IMPLICIT InternationalString OPTIONAL,
                description         [3] IMPLICIT HumanString OPTIONAL,
                asn1Module          [4] IMPLICIT InternationalString OPTIONAL}

— - - - - - - - - - - - - - Subsidiary definitions

CommonInfo ::= SEQUENCE {
  dateAdded                 [0] IMPLICIT GeneralizedTime OPTIONAL,
  dateChanged               [1] IMPLICIT GeneralizedTime OPTIONAL,
  expiry                    [2] IMPLICIT GeneralizedTime OPTIONAL,
  humanString-Language      [3] IMPLICIT LanguageCode OPTIONAL,
            -- following not to occur for brief:
  otherInfo                     OtherInformation OPTIONAL}

HumanString ::= SEQUENCE OF SEQUENCE {
                        language    [0] IMPLICIT LanguageCode OPTIONAL,
                        text        [1] IMPLICIT InternationalString}

IconObject ::= SEQUENCE OF SEQUENCE{
            -- Note that the “SEQUENCE OF” is to allow alternative representations of the same icon; it is not
            -- intended to allow multiple icons.
            bodyType    [1] CHOICE{
                                ianaType        [1] IMPLICIT InternationalString,
                                z3950type       [2] IMPLICIT InternationalString,
                                otherType       [3] IMPLICIT InternationalString},
            content     [2] IMPLICIT OCTET STRING}

LanguageCode ::= InternationalString  — from ANSI/NISO Z39.53-1994

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

NetworkAddress ::= CHOICE {
  internetAddress           [0] IMPLICIT SEQUENCE {
                                      hostAddress     [0]  IMPLICIT InternationalString,
                              port            [1]  IMPLICIT INTEGER},
  osiPresentationAddress  [1] IMPLICIT SEQUENCE {
                              pSel            [0] IMPLICIT InternationalString,
                              sSel            [1] IMPLICIT InternationalString OPTIONAL,
                              tSel            [2] IMPLICIT InternationalString OPTIONAL,
                              nSap            [3] IMPLICIT InternationalString},
  other                   [2] IMPLICIT SEQUENCE {
                              type            [0] IMPLICIT InternationalString,
                              address         [1] IMPLICIT InternationalString}}
AccessInfo ::= SEQUENCE {
-- AccessInfo contains the fundamental information about what facilities are required to use this target
-- or server. For example, if an origin can handle none of the record syntaxes a database can provide,
-- it might choose not to access the database.
  queryTypesSupported        [0]    IMPLICIT SEQUENCE OF QueryTypeDetails OPTIONAL,
  diagnosticsSets            [1]    IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
  attributeSetIds            [2]    IMPLICIT SEQUENCE OF AttributeSetId OPTIONAL,
  schemas                    [3]    IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
  recordSyntaxes             [4]    IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
  resourceChallenges         [5]    IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
  restrictedAccess           [6]    IMPLICIT AccessRestrictions OPTIONAL,
  costInfo                   [8]    IMPLICIT Costs OPTIONAL,
  variantSets                [9]    IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
  elementSetNames            [10]   IMPLICIT SEQUENCE OF ElementSetName OPTIONAL,
  unitSystems                [11]   IMPLICIT SEQUENCE OF InternationalString}

-- begin auxiliary definitions for AccessInfo
-- Begin Query Details
QueryTypeDetails ::= CHOICE {
  private       [0]     IMPLICIT PrivateCapabilities,
  rpn           [1]     IMPLICIT RpnCapabilities,
  iso8777       [2]     IMPLICIT Iso8777Capabilities,
  z39-58        [100]   IMPLICIT HumanString,
  erpn          [101]   IMPLICIT RpnCapabilities,
  rankedList    [102]   IMPLICIT HumanString}

PrivateCapabilities ::= SEQUENCE {
  operators     [0]  IMPLICIT SEQUENCE OF SEQUENCE {
                          operator         [0] IMPLICIT InternationalString,
                          description      [1] IMPLICIT HumanString OPTIONAL } OPTIONAL,
  searchKeys    [1]  IMPLICIT SEQUENCE OF SearchKey OPTIONAL,  -- field names that can be searched
  description   [2]  IMPLICIT SEQUENCE OF HumanString OPTIONAL }

RpnCapabilities ::= SEQUENCE {
  operators                       [0] IMPLICIT SEQUENCE OF INTEGER OPTIONAL,
                                                      — Omitted means all operators are supported.
  resultSetAsOperandSupported     [1] IMPLICIT BOOLEAN,
  restrictionOperandSupported     [2] IMPLICIT BOOLEAN,
  proximity                       [3] IMPLICIT ProximitySupport OPTIONAL}


Iso8777Capabilities ::= SEQUENCE {
  searchKeys    [0]  IMPLICIT SEQUENCE OF SearchKey,  -- field names that may be searched
  restrictions  [1]  IMPLICIT HumanString OPTIONAL
                          — Omitted means supported, not specifying units.
                              }

ProximitySupport ::= SEQUENCE {
  anySupport          [0]   IMPLICIT BOOLEAN,
                         -- 'false' means no proximity support, in which case unitsSupported not supplied.
  unitsSupported      [1]   IMPLICIT SEQUENCE OF CHOICE{
                          known        [1] IMPLICIT INTEGER, -- values from KnownProximityUnit
                          private      [2] IMPLICIT SEQUENCE{
                                              unit              [0] IMPLICIT INTEGER,
                                            description        [1] HumanString OPTIONAL}} OPTIONAL}
SearchKey ::= SEQUENCE {
  searchKey             [0]  IMPLICIT InternationalString,
  description           [1]  IMPLICIT HumanString OPTIONAL }
-- End Query details

AccessRestrictions ::= SEQUENCE OF SEQUENCE {
  accessType                [0] INTEGER {
                                  any                   (0),
                                  search                (1),
                                  present               (2),
                                  specific-elements     (3),
                                  extended-services     (4),
                                  by-database           (5)},
  accessText                [1] IMPLICIT HumanString OPTIONAL,
  accessChallenges          [2] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL}

Costs ::= SEQUENCE {
  connectCharge         [0] IMPLICIT Charge OPTIONAL,         — Per-connection charge
  connectTime           [1] IMPLICIT Charge OPTIONAL,         — Time-based charge
  displayCharge         [2] IMPLICIT Charge OPTIONAL,         — Per-record charge
  searchCharge          [3] IMPLICIT Charge OPTIONAL,         — Per-search charge
  subscriptCharge       [4]  IMPLICIT Charge OPTIONAL,        -- Subscription charges
  otherCharges          [5] IMPLICIT SEQUENCE OF SEQUENCE{     — Other charges
                                forWhat     [1]  IMPLICIT HumanString,
                                charge      [2]  IMPLICIT Charge} OPTIONAL}

            Charge ::= SEQUENCE{
                cost          [1] IMPLICIT IntUnit,
                perWhat       [2] IMPLICIT Unit OPTIONAL,
                                  — e.g., “second,” “minute,” “line,” “record.”..
                text          [3] IMPLICIT HumanString OPTIONAL}
-- End Auxiliary definitions for AccessInfo

DatabaseList ::= SEQUENCE OF DatabaseName

AttributeCombinations ::= SEQUENCE {
  defaultAttributeSet       [0] IMPLICIT AttributeSetId,
                                  -- Default for the combinations. Also probably a good choice for the default
                                  -- in searches, but that isn’t required.
  legalCombinations         [1] IMPLICIT SEQUENCE OF AttributeCombination }

AttributeCombination ::= SEQUENCE OF AttributeOccurrence
                                  — An AttributeCombination is a pattern for legal combination of attributes

AttributeOccurrence ::= SEQUENCE {
              — An AttributeOccurrence lists the legal values for a specific attribute type in a combination.
    attributeSet            [0]  IMPLICIT AttributeSetId OPTIONAL,
    attributeType           [1]  IMPLICIT INTEGER,
    mustBeSupplied          [2]  IMPLICIT NULL OPTIONAL,
    attributeValues         CHOICE {
                                        any-or-none     [3] IMPLICIT NULL, — All supported values are OK
                                        specific        [4] IMPLICIT SEQUENCE OF StringOrNumeric}}
                                                                        — Only these values allowed
END
```

REC.2 Simple Unstructured Text Record Syntax

The Simple Unstructured Text Record Syntax (SUTRS) is intended to be used as a record syntax in a Search or Present response, to present textual data so that the origin may display them with little or no analysis and manipulation.

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.

This definition recommends that the maximum line length be 72 characters unless an alternative maximum is requested, for example via a variantRequest. This is not an absolute maximum, but it is recommended that targets make a best effort to limit lines to this length.

RecordSyntax-SUTRS
```
{Z39-50-recordSyntax SUTRS (101)} DEFINITIONS ::=
BEGIN
IMPORTS InternationalString  FROM Z39-50-APDU-1995;
        SutrsRecord ::=  InternationalString
-- 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
IMPORTS InternationalString  FROM Z39-50-APDU-1995;
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 InternationalString OPTIONAL, — LDR 06
        encodingLevel           [2]     IMPLICIT InternationalString OPTIONAL, — LDR 017
        format                  [3]     IMPLICIT InternationalString OPTIONAL, — 007 00-01
        receiptAcqStatus        [4]     IMPLICIT InternationalString OPTIONAL, — 008 06
        generalRetention        [5]     IMPLICIT InternationalString OPTIONAL, — 008 12
        completeness            [6]     IMPLICIT InternationalString OPTIONAL, — 008 16
        dateOfReport            [7]     IMPLICIT InternationalString OPTIONAL, — 008 26-31
        nucCode                 [8]     IMPLICIT InternationalString OPTIONAL, — 852 $a
        localLocation           [9]     IMPLICIT InternationalString OPTIONAL, — 852 $b
        shelvingLocation        [10]    IMPLICIT InternationalString OPTIONAL, — 852 $c
        callNumber              [11]    IMPLICIT InternationalString OPTIONAL, — 852 $h and $i
        shelvingData            [12]    IMPLICIT InternationalString OPTIONAL, — 852 $j thru $m
        copyNumber              [13]    IMPLICIT InternationalString OPTIONAL, — 852 $t
        publicNote              [14]    IMPLICIT InternationalString OPTIONAL, — 852 $z
        reproductionNote        [15]    IMPLICIT InternationalString OPTIONAL, — 843
        termsUseRepro           [16]    IMPLICIT InternationalString OPTIONAL, — 845
        enumAndChron            [17]    IMPLICIT InternationalString 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 InternationalString OPTIONAL,
        chronology              [2]     IMPLICIT InternationalString OPTIONAL,
        enumAndChron            [3]     IMPLICIT InternationalString OPTIONAL }
CircRecord ::=  SEQUENCE {
        availableNow            [1]     IMPLICIT BOOLEAN,
      availablityDate          [2]    IMPLICIT InternationalString OPTIONAL,
      availableThru            [3]    IMPLICIT InternationalString OPTIONAL,
      restrictions             [4]    IMPLICIT InternationalString OPTIONAL,
      itemId                   [5]    IMPLICIT InternationalString OPTIONAL,
      renewable                [6]    IMPLICIT BOOLEAN,
      onHold                   [7]    IMPLICIT BOOLEAN,
      enumAndChron             [8]    IMPLICIT InternationalString OPTIONAL,
      midspine                 [9]    IMPLICIT InternationalString OPTIONAL,
      temporaryLocation        [10]       IMPLICIT InternationalString OPTIONAL}
END
```

REC.4 Summary Record Syntax

```
  RecordSyntax-summary
    {Z39-50-recordSyntax summary (103)} DEFINITIONS ::=
    BEGIN
    IMPORTS OtherInformation, InternationalString  FROM Z39-50-APDU-1995;
    BriefBib ::= SEQUENCE {
      title                    [1]    IMPLICIT InternationalString,
      author                   [2]    IMPLICIT InternationalString OPTIONAL,
      callNumber               [3]    IMPLICIT InternationalString OPTIONAL,
      recordType               [4]    IMPLICIT InternationalString OPTIONAL,
      bibliographicLevel       [5]    IMPLICIT InternationalString OPTIONAL,
      format                   [6]    IMPLICIT SEQUENCE OF FormatSpec OPTIONAL,
      publicationPlace         [7]    IMPLICIT InternationalString OPTIONAL,
      publicationDate          [8]    IMPLICIT InternationalString OPTIONAL,
      targetSystemKey          [9]    IMPLICIT InternationalString OPTIONAL,
      satisfyingElement        [10]   IMPLICIT InternationalString OPTIONAL,
      rank                     [11]   IMPLICIT INTEGER OPTIONAL,
      documentId               [12]   IMPLICIT InternationalString OPTIONAL,
      abstract                 [13]   IMPLICIT InternationalString OPTIONAL,
      otherInfo                           OtherInformation OPTIONAL}

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

REC.5 Generic Record Syntax 1

```
RecordSyntax-generic  -- For detailed semantics, see Annex 14, RET.
  {Z39-50-recordSyntax GRS-1 (105)} DEFINITIONS ::=
  BEGIN
  EXPORTS Variant;
  IMPORTS IntUnit, Unit, InternationalString, StringOrNumeric, Term FROM Z39-50-APDU-1995;

  GenericRecord ::= SEQUENCE OF TaggedElement
  TaggedElement ::= SEQUENCE {
        tagType            [1] IMPLICIT INTEGER OPTIONAL,
                           -- If omitted, default should be supplied dynamically by tagSet-M;
                           -- otherwise it should be statically specified by the schema.
        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.
        content         [4] ElementData,
        metaData        [5] IMPLICIT ElementMetaData OPTIONAL,
        appliedVariant  [6] IMPLICIT Variant OPTIONAL}

ElementData ::= CHOICE{
        octets          OCTET STRING,
        numeric         INTEGER,
        date            GeneralizedTime,
        ext             EXTERNAL,
        string          InternationalString,
        trueOrFalse     BOOLEAN,
        oid         OBJECT IDENTIFIER,
        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, -- only for a non-leaf node
        usageRight              [2]     IMPLICIT Usage OPTIONAL,
        hits                    [3]     IMPLICIT SEQUENCE OF HitVector OPTIONAL,
        displayName             [4]     IMPLICIT InternationalString OPTIONAL,
                                        -- name for element that origin can use for display
        supportedVariants       [5]     IMPLICIT SEQUENCE OF Variant OPTIONAL,
        message                 [6]     IMPLICIT InternationalString OPTIONAL,
        elementDescriptor       [7]     IMPLICIT OCTET STRING OPTIONAL,
        surrogateFor            [8]     IMPLICIT TagPath OPTIONAL,
                                        -- the retrieved element is a surrogate for the element given by this path
        surrogateElement        [9]     IMPLICIT TagPath OPTIONAL,
                                        -- the element given by this path is a surrogate for the retrieved element
        other                   [99]    IMPLICIT EXTERNAL OPTIONAL}
                TagPath ::= SEQUENCE  OF SEQUENCE{
                        tagType         [1] IMPLICIT INTEGER OPTIONAL,
                        tagValue        [2] StringOrNumeric,
                        tagOccurrence   [3] IMPLICIT INTEGER 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
                        -- Same as defined by 'elementOrdering' in tagSet-M, though this may be
                        -- overridden by schema.
                        }

Usage ::= SEQUENCE      {
        type            [1] IMPLICIT INTEGER{
                                redistributable     (1),    — Element is freely redistributable.
                                restricted          (2),    — Restriction contains statement.
                                licensePointer      (3) -- Restriction contains license pointer.
                                },
        restriction     [2] IMPLICIT InternationalString OPTIONAL}

HitVector ::= SEQUENCE{
```

```
-- Each hit vector points to a fragment within the element, via location and/or token.
        satisfier               Term OPTIONAL, -- sourceword, etc.
        offsetIntoElement       [1] IMPLICIT IntUnit OPTIONAL,
        length                  [2] IMPLICIT IntUnit OPTIONAL,
        hitRank                 [3] IMPLICIT INTEGER OPTIONAL,
        targetToken             [4] IMPLICIT OCTET STRING OPTIONAL
                                -- Origin may use token subsequently within a variantRequest (in
                                --  an elementRequest) to retrieve (or to refer to) the fragment.
                                }

Variant ::= SEQUENCE{
        globalVariantSetId      [1] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
                                -- Applies to the triples below, when variantSetId omitted. If
                                -- globalVariantSetId omitted, default applies. Default may be provided by
                                -- the tagSet-M element defaultVariantSetId.
        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 is used.
                                        class           [1] IMPLICIT INTEGER,
                                        type            [2] IMPLICIT INTEGER,
                                        value           [3] CHOICE{
                                                                INTEGER,
                                                                InternationalString,
                                                                OCTET STRING,
                                                                OBJECT IDENTIFIER,
                                                                BOOLEAN,
                                                                NULL,
                                            -- Following need context tags:
                                                        unit            [1] IMPLICIT Unit,
                                                        valueAndUnit    [2] IMPLICIT IntUnit}}}
END
```

REC.6 Record Syntax For Extended Services Task Package

RecordSyntax-ESTaskPackage

```
   {Z39-50-recordSyntax ESTaskPackage (106)} DEFINITIONS ::=
BEGIN
IMPORTS Permissions, InternationalString, IntUnit, DiagRec FROM Z39-50-APDU-1995;

TaskPackage ::= SEQUENCE{
        packageType             [1]     IMPLICIT OBJECT IDENTIFIER,
                                -- oid of specific ES definition
        packageName             [2]     IMPLICIT InternationalString OPTIONAL,
        userId                  [3]     IMPLICIT InternationalString OPTIONAL,
        retentionTime           [4]     IMPLICIT IntUnit OPTIONAL,
        permissions             [5]     IMPLICIT Permissions OPTIONAL,
        description             [6]     IMPLICIT InternationalString OPTIONAL,
        targetReference         [7]     IMPLICIT OCTET STRING OPTIONAL,
        creationDateTime        [8]     IMPLICIT GeneralizedTime OPTIONAL,
        taskStatus              [9]     IMPLICIT INTEGER{
                                                pending         (0),
                                             active        (1),
                                             complete      (2),
                                             aborted       (3)},
      packageDiagnostics        [10]    IMPLICIT SEQUENCE OF DiagRec OPTIONAL,
      taskSpecificParameters    [11]    IMPLICIT EXTERNAL
                                -- Use oid for specific ES definition
                                -- (same oid as packageType above)
                                -- and select [2] “taskPackage.”
          }
END
```

Annex 6
(Normative)
RSC: Resource Report Formats

This standard defines and registers the resource report formats resource-1 and resource-2. The following object identifier are assigned:

resource-1 {Z39-50-resourceReport 1} (See RSC.1)

resource-2 {Z39-50-resourceReport 2} (See RSC.2)

RSC.1 Resource Report Format Resource-1

```
ResourceReport-Format-Resource-1
{Z39-50-resourceReport resource-1 (1)} DEFINITIONS ::=
BEGIN
IMPORTS InternationalString FROM Z39-50-APDU-1995;
--
ResourceReport ::= SEQUENCE{
estimates       [1] IMPLICIT SEQUENCE OF Estimate,
message         [2] IMPLICIT InternationalString}
--
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
currentSearchAssocCost     (13), -- estimated search cost for association so far
currentPresentAssocCost    (14), -- estimated present cost for this association so far
currentConnectAssocCost  (15), -- estimated connect time cost for association so far
currentOtherAssocCost      (16) -- estimated other cost (not included in 13-15) for association so far
                           }
END
```

RSC.2 Resource Report Format Resource-2

```
ResourceReport-Format-Resource-2
{Z39-50-resourceReport resource-2 (2)} DEFINITIONS ::=
BEGIN
IMPORTS InternationalString, StringOrNumeric, IntUnit FROM Z39-50-APDU-1995;
```

```
--
ResourceReport ::= SEQUENCE{
estimates       [1]     IMPLICIT SEQUENCE OF Estimate OPTIONAL,
message         [2]     IMPLICIT InternationalString OPTIONAL}
--
Estimate ::= SEQUENCE{
type            [1]     StringOrNumeric,
                        -- Numeric values of 1-16 are the same as used in Resource-1.
value           [2]     IMPLICIT IntUnit
                        --      When expressing currency:
                        --      unitSystem (of Unit) is ‘z3950’    (case insensitive)
                        --      unitType is ‘iso4217-1990’         (case insensitive)
                        --      unit is currency code from ISO 4217-1990.
}
END
```

Annex 7
(Normative)
ACC: Access Control Formats

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}**

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
IMPORTS InternationalString, DiagRec FROM Z39-50-APDU-1995;
--

 PromptObject ::= CHOICE{
   challenge          [1] IMPLICIT Challenge,
   response           [2] IMPLICIT Response}

 Challenge ::= SEQUENCE OF SEQUENCE {
   promptId           [1] PromptId,
                      -- Target supplies a number (for an enumerated prompt) or string (for a non
                      -- -enumerated prompt), for each prompt, and the origin returns it in response, for
                      -- this prompt, so target may correlate the prompt response with the prompt.
   defaultResponse    [2] IMPLICIT InternationalString OPTIONAL,
   promptInfo         [3] CHOICE{
                            character     [1] IMPLICIT InternationalString,
                            encrypted     [2] IMPLICIT Encryption} OPTIONAL,
                      -- Information corresponding to an enumerated prompt. For example if 'type', within
                      -- PromptId, is 'copyright', then promptInfo may contain a copyright statement.
   regExpr            [4] IMPLICIT InternationalString OPTIONAL,
                      -- A regular expression that promptResponse should match. See IEEE 1003.2
                      -- Volume 1, Section 2.8 "Regular Expression Notation." For example if promptId
                      -- is "Year of publication," regExpr might be "19[89][0-9]|20[0-9][0-9]".
   responseRequired   [5] IMPLICIT NULL OPTIONAL,
   allowedValues      [6] IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
                      -- e.g. promptId="Desired color"; allowed = 'red', 'blue','Green'.
   shouldSave         [7] IMPLICIT NULL OPTIONAL,
                      -- Target recommends that origin save the data that it prompts from the
                      -- user corresponding to this prompt, because it is likely to be requested again (so
                      -- origin might not have to prompt the user next time).

 dataType             [8] IMPLICIT INTEGER{
                               integer        (1),
                               date           (2),
                               float          (3),
                               alphaNumeric   (4),
                               url-urn        (5),
                               boolean        (6)} OPTIONAL,
                               -- Target telling origin type of data it wants. E.g., if "date" is specified,
                               -- presumably the origin will try to prompt something "date-like" from the user.
 diagnostic           [9] IMPLICIT EXTERNAL OPTIONAL
                               -- Intended for repeat requests when there is an error the origin
                               -- should report to the user from previous attempt.
                      }
 Response ::= SEQUENCE OF SEQUENCE {
   promptId            [1] PromptId,
                       -- Corresponds to a prompt in the challenge, or may be unprompted, for
                       -- example "newPassword." If unprompted, should be "enumerated."
                       -- If this responds to a non-enumerated prompt, then nonEnumeratedPrompt
                       -- should contain the prompt string from the challenge.
   promptResponse      [2] CHOICE{
                              string          [1] IMPLICIT InternationalString,
                              accept          [2] IMPLICIT BOOLEAN,
                              acknowledge     [3] IMPLICIT NULL,
                              diagnostic      [4] DiagRec,
                              encrypted       [5] IMPLICIT Encryption}}


 PromptId ::= CHOICE{
   enummeratedPrompt          [1] IMPLICIT SEQUENCE{
                                     type             [1] IMPLICIT INTEGER{
                                                         groupId          (0),
                                                         userId           (1),
                                                         password         (2),
                                                         newPassword      (3),
                                                         copyright        (4),
                                     -- When type on Challenge is 'copyright', promptInfo has text of
                                     -- copyright message to be displayed verbatim to the user. If
                                     -- promptResponse indicates 'acceptance', this indicates the user has 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.
                                                         sessionId        (5)},
                                     suggestedString     [2] IMPLICIT InternationalString OPTIONAL},
   nonEnumeratedPrompt        [2] IMPLICIT InternationalString}


 Encryption ::= SEQUENCE{
   cryptType          [1] IMPLICIT OCTET STRING OPTIONAL,
   credential         [2] IMPLICIT OCTET STRING OPTIONAL,
                      --random number, SALT, or other factor
   data               [3] IMPLICIT OCTET STRING}

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
IMPORTS InternationalString FROM Z39-50-APDU-1995;

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

Annex 8
(Normative)
EXT: Extended Services Defined by This Standard

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}**

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 that 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.

Table A8-1: Parameters Common to All Extended Services

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

The specific parameters are derived from the ES request parameter Task-specific-parameters. Table A8-1 provides a summary of common parameters. Their descriptions are included in 3.2.9.1. For parameters listed as both “origin supplied” and “target supplied,” when both origin and target supply a value, the target supplied value overrides the origin supplied value.

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 belonging to the current Z-association. The Persistent Result Set 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, 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 that 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 (e.g., within a query, or in Present, Sort, or Delete request).

The parameters of the Persistent Result Set Extended Service are those shown in Table A8-1 as well as those in Table A8-2.

Table A8-2 Specific Parameters for Persistent Result Set

Specific Task **Parameter**	**Origin** **Supplied**	**Target** **Supplied**	**Task Package** **Parameter**
originSuppliedResultSet	x (if appl)		
replaceOrAppend	x (if appl)		
targetSuppliedResultSet		x (if appl)	x (if appl)
numberOfRecords		x (opt)	x (opt)

originSuppliedResultSet—The origin supplies the name of a transient result set belonging to the Z-association. If function is 'create', the target is to create a persistent result set from this transient result set. If function is 'modify' the target is to either replace an existing persistent result set (corresponding to the specified package name) with this result set, or append this result set to an existing persistent result set. This parameter is mandatory when the value of the request parameter function is 'create' or 'modify', and is not included when function is 'delete'.

replaceOrAppend—This parameter occurs when function is 'modify' (and is valid only when the origin user has "modify-contents" permission). Its value is 'replace' or 'append' meaning that the specified result set is, respectively, to replace, or to be appended to, the existing persistent result set.

targetSuppliedResultSet—When the origin retrieves the task package, the target supplies the name of a transient result set, which then belongs to the Z-association. The result set is a copy of the persistent result set represented by the package. The target includes this parameter only when the task package is retrieved (i.e., not on an ES response) and does not include the parameter if the element set name on the Present request indicates that the parameter is not to be included.

numberOfRecords—The target indicates the total number of records in the persistent result set.

EXT.1.2 Persistent Query Extended Service

The Persistent Query Extended Service allows an origin to request that the target 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 A8-1 as well as those in Table A8-3.

Table A8-3 Specific Parameters for Persistent Query

Specific Task **Parameter**	**Origin** **Supplied**	**Target** **Supplied**	**Task Package** **Parameter**
querySpec	x		
actualQuery		x	x
databaseNames	x (opt)		x (opt)
additionalSearchInformation	x (opt)	x (opt)	

querySpec and ActualQuery—The origin supplies either the query to be saved or the name of another persistent query to be copied into this package. The target supplies the actualQuery: if the origin has supplied a query, the target uses that query; if the origin supplies a task package name, the target copies the corresponding query.

databaseNames—The origin optionally supplies a list of databases.

additionalSearchInformation—See 3.2.2.1.12.

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 A8-1 as well as those in Table A8-4.

Table A8-4: Specific Parameters for Periodic Query Schedule

Specific Task Parameter	Origin Supplied	Target Supplied	Task Package Parameter
activeFlag	x		x
querySpec	x		
actualQuery		x	x
databaseNames	x (if appl)		x (if appl)
period	x	x (opt)	x
expiration	x (opt)	x (opt)	x (opt)
resultSetPackageName	x (opt)	x (if appl)	x (if appl)
resultSetDisposition	x (if appl)		x (if appl)
alertDestination	x (opt)		x (opt)
exportParameters	x (opt)		x (opt)
lastQueryTime		x	x
lastResultNumber		x	x
numberSinceModify		x (opt)	x (opt)

activeFlag—On a Create request, if this flag is 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 and ActualQuery—The origin supplies either a query or the name of a Persistent Query Package. (If the origin supplies a query, or if the specified query package does not include a list of databases, then the databaseNames parameter is required.) The target supplies the actualQuery: if the origin has supplied a query, the target uses that query; if the origin supplies a task package name, the target copies the corresponding query.

databaseNames—The origin may supply a list of databases; the list is required if the origin supplies a query rather than a query package name for querySpec, or if the specified query package does not include a list of databases.

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—The origin may optionally supply a time/date for the target to discontinue execution of this Periodic Query. If the origin does not supply a value, the origin is proposing “no expiration.” 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.

resultSetPackageName—The origin may optionally supply the name of an existing Persistent Result Set package. If the origin omits this parameter, the target is to create a persistent result set, unless the parameter exportParameters is included.

resultSetDisposition—This parameter takes on the value ‘createNew’, ‘replace’, or ‘append’, indicating respectively whether the target is to create a new result set each time the query is invoked, replace the contents of the existing result set, or append any new results to the end of the result set. The value ‘createNew’ should be used only if the origin and target have an agreement about naming conventions for the resulting package. If the value of the parameter Period is ‘continuous’ it is recommended that the value of this parameter be ‘append’. The value ‘append’ allows the target to continually extend the result set by appending new records.

alertDestination—The origin may optionally supply a destination address for Alerts triggered by receipt of new Periodic Query results (e.g, fax number, X.400 address, pager number).

exportParameters—The origin may optionally supply the name, or actual contents, of an Export Parameter 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 target indicates the last time this Periodic Query was invoked.

lastResultNumber—The target indicates the number of new records obtained last time query was invoked.

numberSinceModify—The target indicates the total number of records obtained via invocation of the 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 parameters of the Item Order Extended Service are those shown in Table A8-1 as well as those in Table A8-5.

Table A8-5 Task-Specific Parameters for Item Order

Specific Task Parameter	Origin Supplied	Target Supplied	Task Package Parameter
requestedItem	x		
itemRequest	x (if appl)	x (if appl)	
supplementalDescription	x (opt)	x (opt)	
contactInformation	x (opt)		x (opt)
additionalBillingInfo	x (opt)	x (opt)	
statusOrErrorReport	x	x	
auxiliaryStatus	x (opt)	x (opt)	

requestedItem—The origin identifies the requested item, either by:

(a) a request whose format is defined externally, and which may be an Interlibrary Loan Request APDU of ISO 10161; or
(b) a result set item (name of a transient result set belonging to the current Z-association and an ordinal number of an entry within that result); or
(c) both.

itemRequest—If requestedItem is (a) (an interlibrary loan request, for example), the target copies it into the task package (although the target might first modify the request). If requestedItem is (b), the target may construct a corresponding item re quest; if it does not, then the requested item will not be identified within the task package.

supplementalDescription—The origin may supply additional descriptive information pertaining to the requested item, as a supplement to requestedItem.

contactInformation—The origin may optionally supply a name, phone number, and electronic mail address of a contact-person.

additionalBillingInfo—The origin may optionally indicate payment method, credit card information, customer reference, and customer purchase order number.

statusOrErrorReport—The target supplies a status or error report. The definition of the report is external to this standard, and may be based on the StatusOrErrorReport APDU of the ILL protocol.

auxiliaryStatus—The target may provide an auxiliary status as a supplement to the status information which might be provided by the statusOrErrorReport.

EXT 1.5 Database Update Extended Service

The database Update Extended Service allows an origin to request that the target update a database: insert new records, replace or delete existing records, or update elements within 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 requests may update a record whose content has changed.

The parameters of the databaseUpdate Extended Service are those shown in Table A8-1 as well as those in Table A8-6.

Table A8-6: Task-Specific Parameters for DatabaseUpdate

Specific Task **Parameter**	**Origin** **Supplied**	**Target** **Supplied**	**Task Package** **Parameter**
action	x		x
databaseName	x		x
schema	x (opt)		x (opt)
suppliedRecords	x		
recordIds	x (opt)		
supplementalIds	x (opt)		
correlationInfo	x (opt)		x (opt)
elementSetName	x (opt)		x (opt)
updateStatus		x (if appl)	x (if appl)
globalDiagnostics		x (if appl)	x (if appl)
taskPackageRecords		x (if appl)	x (if appl)
recordStatuses		x (if appl)	x (if appl)

action—The origin indicates recordInsert, recordReplace, recordDelete, or elementUpdate.

databaseName—The origin indicates the database to which the action pertains.

schema—The origin indicates the database schema that applies for this update.
Note: The action, databaseName, and schema are specified once, and apply to all of the included records.

suppliedRecords—The origin supplies one or more records. (Along with each the origin may also supply a recordId, supplemental identification, and correlation information; see following three parameters.) For recordInsert or recordReplace, the origin supplies whole records. For recordReplace or recordDelete, each supplied record (or corresponding supplemental identification or recordId) must include sufficient information for the target to identify the database record. For recordDelete, sufficient identifying information should be supplied for each record, but the whole record need not necessarily be supplied.

For elementUpdate, the elements within a supplied record are to replace the corresponding elements within the database record, and the remainder of the database record is unaffected. Records must be supplied in a manner that allows the corresponding elements in the database record to be identified (e.g., via tags defined by the schema). For any element within a supplied record, if there is no corresponding element within the database record, if there is more than a single occurrence of the corresponding element, or if the element is not sufficiently identified, the update will not be performed for that record. (For elementUpdate, supplementalId may be used for identification of the record, but not for identification of elements.)

recordIds—Corresponding to each supplied record the origin may optionally supply a record Id.

supplementalIds—Corresponding to each supplied record the origin may supply supplemental identification to allow the target to identify the database record, or to identify the correct version of the database record. This may be a timestamp, a version number, or may take some other form, for example, a previous version of the record.

CorrelationInfo—Corresponding to each supplied record, the origin may include one or both of the following:
- a correlationNote,
- a correlationIdentifier.

The correlationIdentifier may be used to identify the record only within the context of this update task, for correlation purposes only (i.e to correlate a task package record with its corresponding supplied record). It may be used in the task package in lieu of a record id, for a record that might not have an unambiguous record id.

ElementSetName—The origin indicates an element set name indicating which elements of the updated records are to be included in the task package. If omitted, updated records are not to be included in the task package.

updateStatus—This parameter occurs in the task package only when taskStatus is 'complete' or 'aborted'. It is one of the following:

success	-	Update performed successfully.
partial	-	Update failed for one or more records.
failure	-	Target rejected execution of the task (one or more non-surrogate diagnostics should be supplied in parameter globalDiagnostics).

globalDiagnostics—One or more non-surrogate diagnostics, supplied if updateStatus is Failure.

taskPackageRecords—When taskStatus is 'complete': the task package includes a structure for each supplied record. The structure may include part or all of the updated record (depending on 'elementSetName') or a surrogate diagnostic (when recordStatus, below, is 'failure'), as well as correlationInfo and record status (see next parameter).

When taskStatus is 'pending' or 'active': the task package includes the above for each record for which update action is complete. For those records for which action is not complete, the structure includes the correlationInfo and status.

recordStatuses—Corresponding to each task package record, the task package includes a record status:

success	-	The record was updated successfully.
queued	-	The record is queued for update, or the update is in process (this status may be used in lieu of inProcess, when the target does not wish to distinguish between these two statuses).
inProcess	-	The update for this record is in process.
failure	-	The update for this record failed. A surrogate diagnostic should be supplied in lieu of the record (within the structure corresponding to the record, within the parameter taskPackageRecords).

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; in fact, multiple invocations may be running simultaneously.

An Export Specification includes a delivery destination as well as other information that controls 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, electronic mail, file transfer, or a target-supported print device. The parameters of the Export Specification Extended Service are those shown in Table A8-1 as well as those in Table A8-7.

Table A8-7 Task-Specific Parameters for Export Specification

Specific Task Parameter	Origin Supplied	Target Supplied	Task Package Parameter
composition	x		x
exportDestination	x		x

composition—This parameter consists of a record syntax, element specification, variants, etc. of the records to be Exported.

exportDestination—The origin indicates an address or other destination instruction (e.g., e-mail address, printer address, fax number).

EXT 1.7 Export Invocation Extended Service

The Export Invocation Extended Service allows an origin to invoke an export specification. The origin may supply an export specification, or the name of an export specification that has been established by an Export Specification task as described in EXT 1.6. The parameters of the Export Invocation Extended Service are those shown in Table A8-1 as well as those in Table A8-8.

Table A8-8 Task-Specific Parameters for Export Invocation

Specific Task Parameter	Origin Supplied	Target Supplied	Task Package Parameter
exportSpecification	x		x
resultSetId	x		
resultSetRecords	x		
numberOfCopies	x		x
estimatedQuantity		x (opt)	x (opt)
quantitySoFar		x (opt)	x (opt)
estimatedCost		x (opt)	x (opt)
costSoFar		x (opt)	x (opt)

exportSpecification—The origin supplies the packageName, or actual contents, of an export specification.

resultSetId—The origin supplies the name of a transient result set, from which records are selected for export.

resultSetRecords—The origin indicates which records are to be exported. 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—The origin indicates the number of copies requested.

estimatedQuantity and quantitySoFar—The target optionally indicates the number of pages, message packets, etc., estimated in the information to be exported, and the actual amount exported so far.

estimatedCost and costSoFar—The target optionally supplies an estimate of the cost to export this information, and the cost accrued so far.

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

Each definition below corresponds to an individual extended service. Each structure occurs within an ES request or as a task package. Correspondingly, each is defined as a CHOICE of 'esRequest' and 'taskPackage'. If the structure occurs within an ES request, it occurs as the parameter taskSpecificParameters. The structure may occur as a task package either within an ES response (the parameter taskPackage), or in a record retrieved from an ES database, within the parameter taskSpecificParameters within the structure defined by the record syntax ESTaskPackage; see Annex 5, REC.6.

'esRequest' consists of all service parameters supplied by the origin in the ES request, divided into those that are to be retained in the task package ('to keep') and those that are not ('not to keep'). 'taskPackage' consists of all specific task parameters divided into those that are supplied by the origin ('originPart') and those supplied by the target ('targetPart'). Note that 'toKeep' (from 'esRequest') is always the same sub-structure as 'originPart' (from taskPackage), so that structure is shared, in OriginPartToKeep.

Each definition may define one or more of OriginPartToKeep, OriginPartNotToKeep, and TargetPart. In EXT.1, in the parameter table in the service definition for a specific ES, for each parameter:

- If the parameter is marked "origin supplied," but is *not* marked in the right column (i.e., it does not occur in the task parameter package) then that parameter is represented in OriginPartNotToKeep.
- If the parameter is marked "origin supplied," and also marked in the right column, then that parameter is represented in OriginPartToKeep.
- If the parameter is marked "target supplied" (in which case it will always also be marked in the right column), and not also marked "origin supplied" then that parameter is represented in TargetPart.
- If the parameter is marked "origin supplied," and also marked "target supplied" (in which case it will be marked in the right column), then it is a parameter for which the origin may suggest a value and the target may override that value. In this case the origin suggested value is represented in OriginPartNotToKeep and the target value (which may be the same) is represented in TargetPart.

ESFormat-PersistentResultSet

```
{Z39-50-extendedService PersistentResultSet (1)} DEFINITIONS ::=
BEGIN
IMPORTS InternationalString FROM Z39-50-APDU-1995;
PersistentResultSet ::= CHOICE{
   esRequest      [1] IMPLICIT SEQUENCE{
                  toKeep        [1] IMPLICIT NULL,
                  notToKeep   [2] OriginPartNotToKeep OPTIONAL},
   taskPackage   [2] IMPLICIT SEQUENCE{
                  originPart    [1] IMPLICIT NULL,
                  targetPart    [2] TargetPart OPTIONAL}}
OriginPartNotToKeep ::= SEQUENCE{
   originSuppliedResultSet   [1] IMPLICIT InternationalString OPTIONAL,
                  -- name of transient result set, supplied on request, mandatory unless function is 'delete'
   replaceOrAppend             [2] IMPLICIT INTEGER{  -- only if function is "modify"
                  replace       (1),
                  append        (2)} OPTIONAL}
```

```
TargetPart ::= SEQUENCE{
    targetSuppliedResultSet    [1] IMPLICIT InternationalString OPTIONAL,
                    -- Name of transient result set, supplied by target, representing the persistent result set to which
                    -- package pertains. 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 Query, InternationalString, OtherInformation FROM Z39-50-APDU-1995;
PersistentQuery ::= CHOICE{
   esRequest          [1] IMPLICIT SEQUENCE{
                      toKeep        [1] OriginPartToKeep OPTIONAL,
                      notToKeep   [2] OriginPartNotToKeep},
   taskPackage        [2] IMPLICIT SEQUENCE{
                      originPart    [1] OriginPartToKeep OPTIONAL,
                      targetPart    [2] TargetPart}}
OriginPartToKeep ::= SEQUENCE{
    dbNames                  [2] IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
    additionalSearchInfo   [3] OtherInformation OPTIONAL}
OriginPartNotToKeep ::= CHOICE{
    package             [1] IMPLICIT InternationalString,
    query               [2] Query}
TargetPart ::= Query
END
```

ESFormat-PeriodicQuerySchedule

```
{Z39-50-extendedService PeriodicQuerySchedule (3)} DEFINITIONS ::=
BEGIN
IMPORTS Query, InternationalString, IntUnit FROM Z39-50-APDU-1995
ExportSpecification, Destination FROM ESFormat-ExportSpecification;

PeriodicQuerySchedule ::= CHOICE{
   esRequest        [1] IMPLICIT SEQUENCE{
                                   toKeep          [1] OriginPartToKeep,
                                   notToKeep    [2] OriginPartNotToKeep},
   taskPackage     [2] IMPLICIT SEQUENCE{
                                   originPart      [1] OriginPartToKeep,
                                   targetPart      [2] TargetPart}}

OriginPartToKeep ::=SEQUENCE{
      activeFlag                          [1] IMPLICIT BOOLEAN,
      databaseNames                    [2] IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
      resultSetDisposition            [3] IMPLICIT INTEGER{
                                                    replace        (1),
                                                    append         (2),
                                                    createNew     (3)   -- Only if origin and target have agreement about
                                                                              -- naming convention for the resulting package,
                                                                              -- and only if no result set is specified.
                                             } OPTIONAL,     -- Mandatory on 'create' if result set is specified, in
                                                                     -- which case it must be 'replace' or 'append.
      alertDestination                 [4] Destination OPTIONAL,
      exportParameters               [5] CHOICE{
                                                    packageName       [1]    IMPLICIT InternationalString,
                                                    exportPackage      [2] ExportSpecification} OPTIONAL}
```

```
OriginPartNotToKeep ::= SEQUENCE{
    querySpec                    [1] CHOICE{
                                         actualQuery      [1] Query,
                                         packageName      [2] IMPLICIT InternationalString} OPTIONAL,
                                     -- mandatory for 'create'
    originSuggestedPeriod        [2] Period OPTIONAL,  -- mandatory for 'create'
    expiration                   [3] IMPLICIT GeneralizedTime OPTIONAL,
    resultSetPackage             [4] IMPLICIT InternationalString OPTIONAL}
TargetPart ::= SEQUENCE{
    actualQuery                  [1] Query,
    targetStatedPeriod           [2] Period,
                                     -- Target supplies the period, which may be same as origin proposed.
    expiration                   [3] IMPLICIT GeneralizedTime OPTIONAL,
                                     -- Target supplies value for task package. It may be the same as origin
                                     -- proposed or different from (and overrides) origin proposal, but if
                                     -- omitted, there is no expiration.
    resultSetPackage             [4] IMPLICIT InternationalString OPTIONAL,
                                     -- May be omitted only if exportParameters was supplied. Target
                                     -- supplies same name as origin supplied, if origin did supply a name.
    lastQueryTime                [5] IMPLICIT GeneralizedTime,
    lastResultNumber             [6] IMPLICIT INTEGER,
    numberSinceModify            [7] IMPLICIT INTEGER OPTIONAL}

    Period ::= CHOICE{
                unit                 [1] IMPLICIT IntUnit,
                businessDaily        [2] IMPLICIT NULL,
                continuous           [3] IMPLICIT NULL,
                other                [4] IMPLICIT InternationalString}
END
```

ESFormat-ItemOrder

```
{Z39-50-extendedService ItemOrder (4)} DEFINITIONS ::=
BEGIN
IMPORTS InternationalString FROM Z39-50-APDU-1995;
ItemOrder ::= CHOICE{
  esRequest       [1] IMPLICIT SEQUENCE{
                              toKeep       [1] OriginPartToKeep OPTIONAL,
                              notToKeep    [2] OriginPartNotToKeep},
  taskPackage     [2] IMPLICIT SEQUENCE{
                              originPart   [1] OriginPartToKeep OPTIONAL,
                              targetPart   [2] TargetPart}}
OriginPartToKeep ::= SEQUENCE{
    supplDescription         [1] IMPLICIT EXTERNAL OPTIONAL,
    contact                  [2] IMPLICIT SEQUENCE{
                                     name    [1]  IMPLICIT InternationalString OPTIONAL,
                                     phone   [2]  IMPLICIT InternationalString OPTIONAL,
                                     email   [3]  IMPLICIT InternationalString 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 InternationalString OPTIONAL,
                                customerPONumber     [3] IMPLICIT InternationalString OPTIONAL}
                                                                    OPTIONAL}
CreditCardInfo     ::= SEQUENCE{
        nameOnCard            [1] IMPLICIT InternationalString,
        expirationDate        [2] IMPLICIT InternationalString,
        cardNumber            [3] IMPLICIT InternationalString}

OriginPartNotToKeep ::= SEQUENCE{ -- Corresponds to 'requestedItem' in service definition.
                             --   Must supply at least one, and may supply both.
    resultSetItem            [1]  IMPLICIT SEQUENCE{
                                      resultSetId         [1]  IMPLICIT InternationalString,
                                      item                [2]  IMPLICIT INTEGER} OPTIONAL,
    itemRequest              [2]  IMPLICIT EXTERNAL OPTIONAL
                                  -- When itemRequest is an ILL-Request APDU,
                                  -- use OID {iso standard 10161 abstract-syntax (2) ill-apdus (1)}
                                  }

TargetPart ::= SEQUENCE{
    itemRequest              [1]  IMPLICIT EXTERNAL OPTIONAL,
        -- When itemRequest is an ILL-Request APDU, use OID 1.0.10161.2.1 (as above)
    statusOrErrorReport    [2] IMPLICIT EXTERNAL OPTIONAL,
        -- When statusOrErrorReport is an ILL Status-Or-Error-Report APDU, use OID  1.0.10161.2.1 (as above)
    auxiliaryStatus          [3]  IMPLICIT INTEGER{
                                      notReceived             (1),
                                      loanQueue               (2),
                                      forwarded               (3),
                                      unfilledCopyright       (4),
                                      filledCopyright         (5)} OPTIONAL}
END
```

ESFormat-Update

```
{Z39-50-extendedService Update (5)} DEFINITIONS ::=
BEGIN
IMPORTS DiagRec, InternationalString FROM Z39-50-APDU-1995;
Update ::= CHOICE{
   esRequest        [1] IMPLICIT SEQUENCE{
                                      toKeep        [1] OriginPartToKeep,
                                      notToKeep     [2] OriginPartNotToKeep},
   taskPackage      [2] IMPLICIT SEQUENCE{
                                      originPart    [1] OriginPartToKeep,
                                      targetPart    [2] TargetPart}}

OriginPartToKeep ::= SEQUENCE{
    action                            [1] IMPLICIT INTEGER{
                                              recordInsert        (1),
                                              recordReplace       (2),
                                              recordDelete        (3),
                                              elementUpdate       (4)},
    databaseName                      [2] IMPLICIT InternationalString,
    schema                            [3] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
    elementSetName                    [4] IMPLICIT InternationalString OPTIONAL}

OriginPartNotToKeep ::= SuppliedRecords
```

```
TargetPart ::= SEQUENCE{
    updateStatus            [1] IMPLICIT INTEGER{
                                        success  (1),
                                        partial  (2),
                                        failure  (3)},
    globalDiagnostics       [2] IMPLICIT SEQUENCE OF DiagRec OPTIONAL,
                                    -- These are non-surrogate diagnostics relating to the task,
                                    -- not to individual records.
    taskPackageRecords      [3] IMPLICIT SEQUENCE OF TaskPackageRecordStructure
                                    -- There should be a TaskPackageRecordStructure for every record
                                    -- supplied. The target should create such a structure for every
                                    -- record immediately upon creating the task package to include
                                    -- correlation information and status. The record itself would not
                                    -- be included until processing for that record is complete.
                            }

-- Auxiliary definitions for Update
SuppliedRecords ::= SEQUENCE OF SEQUENCE{
    recordId                [1] CHOICE{
                                    number   [1] IMPLICIT INTEGER,
                                    string   [2] IMPLICIT InternationalString,
                                    opaque   [3]  IMPLICIT OCTET STRING} OPTIONAL,
    supplementalId          [2] CHOICE{
                                    timeStamp        [1] IMPLICIT GeneralizedTime,
                                    versionNumber    [2] IMPLICIT InternationalString,
                                    previousVersion  [3] IMPLICIT EXTERNAL} OPTIONAL,
    correlationInfo         [3] IMPLICIT CorrelationInfo OPTIONAL,
    record                  [4] IMPLICIT EXTERNAL}

CorrelationInfo ::= SEQUENCE{
                        -- origin may supply one or both for any record:
    note    [1] IMPLICIT InternationalString OPTIONAL,
    id      [2] IMPLICIT INTEGER OPTIONAL}

TaskPackageRecordStructure ::= SEQUENCE{
        recordOrSurDiag         [1] CHOICE {
                                    record       [1] IMPLICIT EXTERNAL,
                                                     -- Choose 'record' if recordStatus is 'success', and
                                                     -- elementSetName was supplied.
                                    diagnostic   [2] DiagRec
                                                     -- Choose 'diagnostic', if RecordStatus is failure.
                                                                     } OPTIONAL,
                                    -- The parameter recordOrSurDiag will thus be omitted only if
                                    -- 'elementSetName' was omitted and recordStatus is
                                    -- 'success'; or if record status is 'queued' or in 'process'.
        correlationInfo         [2] IMPLICIT CorrelationInfo OPTIONAL,
                                    -- This should be included if it was supplied by the origin.
        recordStatus            [3] IMPLICIT INTEGER{
                                        success      (1),
                                        queued       (2),
                                        inProcess    (3),
                                        failure      (4)}}
END
```

ESFormat-ExportSpecification

```
{Z39-50-extendedService ExportSpecification (6)} DEFINITIONS ::=
BEGIN
EXPORTS ExportSpecification, Destination; IMPORTS CompSpec, InternationalString FROM Z39-50-APDU-1995;
ExportSpecification ::= CHOICE{
   esRequest        [1] IMPLICIT SEQUENCE{
                    toKeep        [1] OriginPartToKeep,
                    notToKeep     [2] IMPLICIT NULL},
   taskPackage      [2] IMPLICIT SEQUENCE{
                    originPart    [1] OriginPartToKeep,
                    targetPart    [2] IMPLICIT NULL}}
OriginPartToKeep ::= SEQUENCE{
      composition                 [1] IMPLICIT CompSpec,
      exportDestination           [2] Destination}
  Destination ::= CHOICE{
      phoneNumber           [1]   IMPLICIT InternationalString,
      faxNumber             [2]   IMPLICIT InternationalString,
      x400address           [3]   IMPLICIT InternationalString,
      emailAddress          [4]   IMPLICIT InternationalString,
      pagerNumber           [5]   IMPLICIT InternationalString,
      ftpAddress            [6]   IMPLICIT InternationalString,
      ftamAddress           [7]   IMPLICIT InternationalString,
      printerAddress        [8]   IMPLICIT InternationalString,
      other                 [100]      IMPLICIT SEQUENCE{
                                            vehicle             [1]   IMPLICIT InternationalString  OPTIONAL,
                                            destination         [2]   IMPLICIT InternationalString}}
END
```

ESFormat-ExportInvocation

```
{Z39-50-extendedService ExportInvocation (7)} DEFINITIONS ::=
BEGIN
IMPORTS InternationalString, IntUnit FROM Z39-50-APDU-1995
ExportSpecification FROM ESFormat-ExportSpecification;
ExportInvocation ::= CHOICE{
   esRequest        [1] IMPLICIT SEQUENCE{
                    toKeep        [1] OriginPartToKeep,
                    notToKeep     [2] OriginPartNotToKeep},
   taskPackage      [2] IMPLICIT SEQUENCE{
                    originPart    [1] OriginPartToKeep,
                    targetPart    [2] TargetPart OPTIONAL}}

OriginPartToKeep ::= SEQUENCE{
      exportSpec          [1] CHOICE{
                                packageName             [1] IMPLICIT InternationalString,
                                packageSpec             [2] ExportSpecification},
      numberOfCopies  [2] IMPLICIT INTEGER}

OriginPartNotToKeep ::= SEQUENCE{
      resultSetId                 [1] IMPLICIT InternationalString,
      records                     [2] CHOICE{
                                        all         [1] IMPLICIT NULL,
                                        ranges    [2]    IMPLICIT SEQUENCE OF SEQUENCE{
                                                              start        [1] IMPLICIT INTEGER,
                                                              count       [2] IMPLICIT INTEGER OPTIONAL
                                                                  -- Count may be omitted only on last range, to indicate
                                                                  -- "all remaining records beginning with 'start'."
                                                  }}}
```

```
TargetPart    ::= SEQUENCE{
        estimatedQuantity        [1] IMPLICIT IntUnit OPTIONAL,
        quantitySoFar            [2] IMPLICIT IntUnit OPTIONAL,
        estimatedCost            [3] IMPLICIT IntUnit OPTIONAL,
        costSoFar                [4] IMPLICIT IntUnit OPTIONAL}
END
```

Annex 9
(Normative)
USR: User Information Formats

User Information formats are defined for the following: 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. The following object identifier is assigned:

SearchResult-1

{Z39-50-userInfoFormat 1} (See USR.1)

UserInformation formats may include negotiation records, defined for the parameters userInformationField and otherInfo in the Init and InitResponse APDUs. These are described in USR.2.

USR.1 User Information Format SearchResult-1

SearchResult-1 is for use primarily within the AdditionalSearchInformation parameter in the Search Response. The format allows the target to provide information per query component (the whole query or a sub-query, possibly restricted to a subset of the specified databases). The target may also create and provide access to a result set for each query component.

This format may also be used as a Resource Report format, within the ResourceReport parameter of the resource-control request, to allow the target to report on the progress of the search. However, when used in this manner, the target should not create a result set for a query component unless processing for that component is complete.

```
UserInfoFormat-searchResult-1
{Z39-50-userInfoFormat searchResult-1 (1)} DEFINITIONS ::=
BEGIN
IMPORTS DatabaseName, Term, Query, IntUnit, InternationalString FROM Z39-50-APDU-1995;
SearchInfoReport ::= SEQUENCE OF SEQUENCE{
   subqueryId                [1] IMPLICIT InternationalString OPTIONAL,
                                                           -- shorthand identifier of subquery
   fullQuery                 [2] IMPLICIT BOOLEAN,         -- 'true' means this is the full query;
                                                           -- 'false',
                                                           -- a subquery
   subqueryExpression        [3] QueryExpression OPTIONAL, -- A subquery of the query as
                                                           -- submitted. May be whole query;
                                                           -- if so, "fullQuery" should be
                                                           -- 'true'.
   subqueryInterpretation    [4] QueryExpression OPTIONAL, -- how target interpreted subquery
   subqueryRecommendation    [5] QueryExpression OPTIONAL, -- target-recommended alternative
   subqueryCount             [6] IMPLICIT INTEGER OPTIONAL,
                                                           -- Number of records for this
                                                           -- subQuery, across all of the
                                                           -- specified databases. (If during
                                                           -- search, via resource control,
                                                           --number of records so far).
   subqueryWeight            [7] IMPLICIT IntUnit OPTIONAL,
                                                           -- relative weight of this subquery
   resultsByDB               [8] IMPLICIT ResultsByDB OPTIONAL}

ResultsByDB ::= SEQUENCE OF SEQUENCE{
   databases             [1] CHOICE{
                              all      [1] IMPLICIT NULL,
                                       -- applies across all of the databases in Search PDU
                              list     [2] IMPLICIT SEQUENCE OF DatabaseName
                                       -- applies across all databases in this list
                                 },
     count              [2] IMPLICIT INTEGER OPTIONAL,
                            -- Number of records for query component (and, as above, if during search,
                            -- via resource control, number of records so far).
     resultSetName      [3] IMPLICIT InternationalString OPTIONAL
                            -- Target-assigned result set by which subQuery is available. Should not
                            -- be provided unless processing for this query component is concluded (i.e.,
                            -- when this report comes during search, via resource control, as opposed
                            -- to after search, via additionalSearchInfo).
                                        }

QueryExpression ::=  CHOICE {
          term                [1] IMPLICIT SEQUENCE{
                                  queryTerm         [1] Term,
                                  termComment       [2] IMPLICIT InternationalString OPTIONAL},
          query               [2] Query}
END
```

USR.2 Negotiation Records

Negotiation records are defined for use within the parameters otherInfo (version 3 only) and userInformationField in the Init and InitResponse APDUs. No negotiation records are defined by this standard. Publicly defined negotiation record definitions are available from the Z39.50 Maintenance Agency.

In general, a negotiation record is defined for use as follows: the origin includes the negotiation record within the Init APDU (identified by its OID) to propose that some condition be in effect for the Z-association. The target may (but is not obligated to) respond to the proposal, using the same negotiation record format, and the target's response, if any, indicates whether the proposal is accepted, or may indicate a counterproposal, which will then be in effect for the Z-association. Thus a negotiation record definition should include the format of both the origin proposal and the target response.

The following rules and guidelines apply to the definition and use of negotiation records:

- A negotiation record should be defined for the purpose of negotiating a single item of information, except in the following case: negotiation of related items may be defined for the same negotiated record where it is not practical to separate their negotiation, for example, because of interdependence among the negotiation of these items.
- If the origin does not propose negotiation (i.e., does not submit a negotiation record) for a given item, then it is considered that "no negotiation takes place" for that item. If the origin does propose negotiation for an item, but the target does not respond (i.e., does not include a corresponding negotiation record), similarly, no negotiation takes place for that item.
- A negotiation record definition must not specify behavior governing the condition where no negotiation takes place. (If no negotiation takes place, neither origin nor target can be assumed to know any rules associated with the negotiation record definition.)
- If the target does not recognize the oid for a negotiation record submitted by the origin, it should ignore it (and not return a negotiation record of that type).

Note: When the target does not recognize the oid of a negotiation record, the target cannot be certain that it is indeed a negotiation record. Therefore, care should be taken in general in defining user information formats to ensure that if the target does not recognize an oid it may ignore it with impunity.

- If the origin does not submit a negotiation record of a particular type in the Init request, then the target is not to include a negotiation record of that type in the response.
- If multiple negotiation records are included in an Init request, there is no significance to their order, and there is no relationship between them: for example, if the origin includes two negotiation records, and the target does not recognize the first (in which case negotiation fails for the first) negotiation may still succeed for the second.

Annex 10
(Normative)
ESP: Element Specification Formats

This Standard defines and registers the element specification format eSpec-1, and assigns it the following object identifier:

eSpec-1 {Z39-50-elementSpec 1}

ElementSpecificationFormat-eSpec-1 *-- For detailed semantics, see Annex 14, RET.*

```
{Z39-50-elementSpec eSpec-1 (1)} DEFINITIONS ::=
BEGIN
IMPORTS Variant FROM RecordSyntax-generic
StringOrNumeric, InternationalString FROM Z39-50-APDU-1995;
--
Espec-1 ::= SEQUENCE{
    elementSetNames         [1] IMPLICIT SEQUENCE OF InternationalString OPTIONAL,
                            -- Origin may include one or more element set names, each
                            -- specifying a set of elements. Each of the elements is to be
                            -- treated as an elementRequest in the form of simpleElement,
                            -- where occurrence is 1.
    defaultVariantSetId     [2] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
                            -- If supplied, applies whenever variantRequest
                            -- does not include variantSetId.
    defaultVariantRequest   [3] IMPLICIT Variant OPTIONAL,
                            -- If supplied, then for each simple elementRequest that does not
                            -- include a variantRequest, the defaultVariantRequest applies.
                            -- (defaultVariantRequest does not apply to a compositeRequest.)
    defaultTagType          [4] IMPLICIT INTEGER OPTIONAL,
                            -- If supplied, applies whenever 'tagType' (within 'tag' within TagPath)
                            -- is omitted.
    elements                [5] IMPLICIT SEQUENCE OF ElementRequest OPTIONAL}
--

ElementRequest::= CHOICE{
    simpleElement           [1] IMPLICIT SimpleElement,
    compositeElement        [2] IMPLICIT SEQUENCE{
                elementList     [1] CHOICE{
                                    primitives      [1] IMPLICIT SEQUENCE OF InternationalString,
                                                    -- Origin may specify one or more element
                                                    -- set names, each identifying a set of elements,
                                                    -- and the composite element is the union.
                                    specs           [2] IMPLICIT SEQUENCE OF SimpleElement},
                deliveryTag     [2] IMPLICIT TagPath,
                                -- DeliveryTag tagPath for compositeElement may not
                                -- include wildThing or wildPath.
              variantRequest    [3] IMPLICIT Variant OPTIONAL}}

SimpleElement ::= SEQUENCE{
        path                        [1] IMPLICIT TagPath,
        variantRequest              [2] IMPLICIT Variant OPTIONAL}
TagPath ::= SEQUENCE OF CHOICE{
    specificTag     [1] IMPLICIT SEQUENCE{
                            tagType     [1] IMPLICIT INTEGER OPTIONAL,
                                        -- If omitted, then 'defaultTagType' (above) applies, if supplied, and
                                        -- if not supplied, then default listed in schema applies.
                            tagValue    [2] StringOrNumeric,
                        occurrence      [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 is included, then single occurrence is requested
                  howMany         [2] IMPLICIT INTEGER OPTIONAL
                  -- For example, if ‘start’ is 5 and ‘howMany’ is 6, then request is for
                  -- “occurrences 5 through 10.”
                                  }}
END
```

Annex 11
(Normative)
VAR: Variant Sets

This standard defines and registers the variant set variant-1, and assigns it the following object identifier:
variant-1 {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 (*For detailed semantics of variant-1, see Annex 14, RET)*.

Class	Type	Value(s)
1 = variantId		
	Class 1 may be used within a supportedVariant, variantRequest, or appliedVariant.	
	1 = variantId	OCTET STRING
2 = BodyPartType		
	Class 2 may be used within a supportedVariant, variantRequest, or appliedVariant.	
	1 = ianaType/subType	InternationalString: "<ianaType>/<subType>" e.g. "application/postscript", where <ianaType> and <subType> are registered with IANA (Internet Assigned Numbers Authority)
	2 = Z39.50Type[/subType]	InternationalString: e.g. "'sgml/'dtdName" (for example "sgml/TEI") or "sgml"
	3 = otherType[/subType]	InternationalString; bilaterally agreed upon
	Note: subtype is optional for types 2 and 3.	
3 = formatting/presentation		
	Class 3 may be used within a supportedVariant, variantRequest, or appliedVariant.	
	1 = characters per line	INTEGER
	2 = line length	IntUnit
	3 = lines per page	INTEGER
	4 = dots per inch	INTEGER
	5 = paperType-Size	InternationalString; e.g. A-1, B, C.
	6 = deliverImages	BOOLEAN
	7 = PortraitOrientation	BOOLEAN ('true' means "portrait")
	8 = textJustification	InternationalString; 'left', 'right', 'both', or 'center'
	9 = fontStyle	InternationalString
	10 = fontSize	InternationalString
	11 = fontMetric	InternationalString
	12 = lineSpacing	INTEGER
	13 = numberOfColumns	INTEGER
	14 = verticalMargins	IntUnit
	15 = horizontalMargins	IntUnit
	16 = pageOrderingForward	BOOLEAN
	17 = beginDocsOnNewPage	BOOLEAN ('false' means "concatenate documents")
	18 = termHighlighting	BOOLEAN
	19 = footnoteLocation	InternationalString: 'inline', endOfPage', 'endEachDoc', 'endLastDoc'
	20 = paginationType	InternationalString

4 = Language/CharacterSet

Class 4 may be used within a supportedVariant, variantRequest, or appliedVariant.

1 = language	InternationalString (from ANSI/NISO Z39.53-1994)
2 = registered character set	INTEGER: registration number from ISO International Register of Character Sets
3 = character set id	OBJECT IDENTIFIER
4 = encoding id	OBJECT IDENTIFIER
5 = private string	InternationalString

5 = Piece

Class 5 may be used within a variantRequest or appliedVariant.

1 = what fragment wanted	INTEGER *(variantRequest only)* 1 = start 2 = next 3 = previous 4 = current 5 = last
2= what fragment returned	INTEGER *(appliedVariant only)* 1 = start 2 = middle 3 = last 4 = end for now 5 = whole
3 = start	IntUnit
4 = end	IntUnit
5 = howMuch	IntUnit
6 = step	INTEGER or IntUnit
7= targetToken	OCTET STRING

6 = meta-data requested

Class 6 may be used within a variantRequest only.

1 = cost	Unit or NULL
2 = size	Unit or NULL
3 = hits, variant-specific	NULL
4 = hits, non-variant-specific	NULL
5 = variant list	NULL
6 = is variant supported?	NULL
7 = document descriptor	NULL
8 = surrogate information	NULL
998 = all meta-data	NULL
999 = other meta-data	OBJECT IDENTIFIER

7 = meta-data returned

Class 7 may be used within a supportedVariant or appliedVariant.

1 = cost	IntUnit
2 = size	IntUnit
3 = integrity	INTEGER
4 = separability	INTEGER
5 = variant supported	BOOLEAN

8 = Highlighting

Class 8 may be used within a variantRequest or appliedVariant.

1 = prefix	OCTET STRING
2 = postfix	OCTET STRING
3 = server default	NULL *(variantRequest only)*

9 = miscellaneous

1 = NoData	NULL *(variantRequest only)*
2 = Unit	Unit *(variantRequest only--origin requests element according to specific unit)*
3 = Version	InternationalString

Annex 12 (Normative) TAG: TagSet Definitions and Schemas

A database schema represents a common understanding, shared by the origin and target, of the information contained in the records of the database represented by that schema, to allow retrieval of portions of that information.

The primary component of a database schema is an abstract record structure, which lists schema elements in terms of their tagPaths. A tagPath is a representation of the hierarchical path of an element, expressed as a sequence of nodes, each represented by a tag. Each tag in a tagPath consists of a tagType and tagValue. The tagType is an integer; the tagValue may be an integer or character string. The tagType qualifies the tagValue; it might identify a tagSet, which might be registered (or alternatively, it might be defined locally within the schema).

Also included in a schema is a definition of how the various tagTypes are used within the tagPaths for the schema elements. The definition might simply be a mapping of tagTypes to tagSets.

For all schemas, tagTypes 1 through 3 are assumed to have the following meaning:

tagType	**used to qualify:**	
1	an element defined in tagSet-M	(see TAG.2.1)
2	an element defined in tagSet-G	(see TAG.2.2)
3	a tag locally defined by the target (intended primarily for string tags, but numeric tags are not precluded)	

For a detailed description of the use of schemas, tagSets, etc. see Annex 14, RET.

TAG.1 Schema Definitions

This standard registers the following object identifiers for Schemas:

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

TAG.2 TagSet Definitions

This standard defines and registers the tag set definitions tagSet-M and tagSet-G. TagSet-M includes elements intended for use as meta-data associated with a database record. TagSet-G includes generic elements.

The object identifier for these definitions are:

tagSet-M {Z39-50-tagSet 1}

tagSet-G {Z39-50-tagSet 2}

For detailed semantics of the elements defined in these tagSets, see Annex 14, RET.

In addition, this standard registers the following tagSet:

tag-Set-STAS {Z39-50-tagSet 3}

TAG.2.1 Definition of tagSet-M

Element	tag	Recommended ASN.1 datatype
schemaIdentifier	1	OBJECT IDENTIFIER
elementsOrdered	2	BOOLEAN
elementOrdering	3	INTEGER
defaultTagType	4	INTEGER
defaultVariantSetId	5	OBJECT IDENTIFIER
defaultVariantSpec	6	VariantSpec
processingInstructions	7	InternationalString
recordUsage	8	INTEGER
restriction	9	InternationalString
rank	10	INTEGER
userMessage	11	InternationalString
url	12	InternationalString
record	13	structured
local control number	14	InternationalString
creation date	15	GeneralizedTime
dateOfLastModification	16	GeneralizedTime
dateOfLastReview	17	GeneralizedTime
score	18	INTEGER
wellKnown	19	InternationalString
recordWrapper	20	structured
defaultTagSetId	21	OBJECT IDENTIFIER

schemaIdentifier—Identifies the schema in use. This element is available 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 parent) are presented as follows: tagTypes are ascending; for elements with the same tagType, integer tag values are ascending, and precede elements with string tags (which are not necessarily ordered).

elementOrdering—How sibling elements with the same tag are ordered:

1 =	“Normal” consumption order (pages, frames).
2 =	Chronological, e.g., 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 may not) be ordered by private agreement.
7 =	Singleton; never more than one occurrence.

defaultTagType—The tagType that applies for any element for which tagType 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.

recordUsage

1 =	Redistributable.
2 =	Restricted, and the tagSet-M element ‘restriction’ (defined below) contains the restriction.
3 =	Restricted, and the restriction contains a license pointer.

restriction—This element, if present, should immediately follow recordUsage, and is a statement (if recordUsage is 1 or 2), or a pointer to the license (if recordUsage is 3).

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

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

url—Uniform resource locator. This is a URL for the record.

record—This element may be used for nested records, when the database record itself includes database records (possibly from a different database). Note that tagSet-M elements that occur subordinate to this element apply only to that nested record.

localControlNumber—An identifier of the record, unique within the database.

creationDate—Date that the record was created.

dateOfLastModification—Most recent date that this record was modified.

dateOfLastReview—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). The normalization factor should be specified in the schema.

wellKnown—When an element is defined to be "structured into locally defined elements," the target may use this tag in lieu of, or along with, locally defined tags. For example, an element named 'title' might be described to be "locally structured." The target might present the element structured into the following subelements: 'wellKnown', "spineTitle," and "variantTitle," where the latter two are string tags, target defined. In this case, 'wellKnown' is assumed to mean "title."

recordWrapper—This element may be used to represent the root of the record, particularly when the record otherwise has no root. The origin may request the record skeleton by reference to this element.

defaultTagSetId —This element may be used in lieu of defaultTagType, to identify the default tag set.

TAG.2.2 Definition of tagSet-G

Element	**tag**	**Recommended ASN.1 datatype**
title	1	InternationalString
author	2	InternationalString
publicationPlace	3	InternationalString
publicationDate	4	InternationalString or GeneralizedTime
documentId	5	InternationalString
abstract	6	InternationalString
name	7	InternationalString
date	8	GeneralizedTime
bodyOfDisplay	9	InternationalString
organization	10	InternationalString
postalAddress	11	InternationalString
networkAddress	12	InternationalString
eMailAddress	13	InternationalString
phoneNumber	14	InternationalString
faxNumber	15	InternationalString
country	16	InternationalString
description	17	InternationalString
time	18	IntUnit
DocumentContent	19	OCTET STRING

These elements (with the exception of bodyOfDisplay) are for generic use and their definitions are not supplied.

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

Annex 13
(Informative)
ERS: Extended Result Set Model

Section 3.1.6 (Model of a Result Set) notes that in the extended result set model for searching, the target maintains unspecified information that the target might maintain to perform proximity operations requiring the extended model, or to evaluate restriction operands.

ERS.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 which may be used in a proximity operation as a surrogate for the search that created the result set.

Example 1:
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 be 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 2:
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.

ERS.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 3. 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 4.

Example 3:
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, and 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 4:
In this example, R and C are two databases. 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 that 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 that 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 examples only.

Annex 14
(Informative)
RET: Z39.50 Retrieval

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*, idiomatically speaking, is the transfer of result set records from the target to the origin.

This Annex describes retrieval, and thus assumes the existence of a result set. For simplicity, it is assumed that the result set has a single record (although Z39.50 retrieval allows an origin to request the retrieval of various combinations of result set records) and this Annex focuses on the capabilities provided by Z39.50 retrieval for retrieving information from that record.

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* (sometimes referred to as "answers"); 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 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 supports the following basic capabilities:

- The origin may request specific logical information *element*s from a record (via an element specification, described below).
- The origin and target may share a name space for tagging elements (via a schema and tagsets, described below), so that elements will be properly identified: by the origin, within an element specification; and by the target, within a retrieval record.
- The origin may request an individual element according to a specific representation or format (via variants, described below).
- The origin may specify how the elements, collectively, are to be packaged into a retrieval record (via a record syntax, described below).

Correspondingly, Z39.50 retrieval has four primary functions:

- Element *selection* (see note)
- Element *tagging*
- Element *representation*
- *Record* representation

Note: *Element* selection pertains to *retrieval*, and should not be confused with *record* selection which pertains to *searching*. Element selection pertains to selection of information elements from already-selected database records.

RET.2 Retrieval Object Classes

This section, RET.2, describes object classes used by these retrieval functions. RET.3 describes in detail specific object definitions that are defined within this standard.

- element specifications (*elementSpec*s), see RET.2.1
- tagSets, see RET.2.1
- schema definitions, see RET.2.2
- variant specifications (*variantSpec*s), see RET.2.3
- record syntaxes, see RET.2.4.

Following is a brief overview of the object classes.

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 specific elements (a set of *elementRequest*s).

A tagSet defines a set of elements, and specifies names and recommended datatypes for individual elements within that set. The name of an element is called its *tag*, and may be used alone (in an elementRequest) or accompanying the element it names (within a retrieval record).

A schema defines an *abstract record structure* (see RET.2.2). The schema definition refers to one or more tagSets.

Although an elementSpec is used primarily for selection, it might have representation aspects: each elementRequest may include a *variantRequest*, used primarily for element representation, to specify the particular form of an element, for example how an element is to be formatted. (However, a variantRequest may include limited selection: it might ask for a specific *piece* or *fragment* of an element.)

A variantRequest is one of three usages of a variantSpec:

- A variantRequest is a variantSpec occurring within an elementRequest.
- An *appliedVariant* is a variantSpec applied to an element by the target, when that element is included in a retrieval record.
- The target might provide a list of the variantSpecs supported for a given element; each is referred to as a *supportedVariant*.

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

To summarize:

- An elementSpec is used (primarily) for element selection.
- A variantRequest is used for element representation.
- A record syntax is used for record representation.
- A tagSet is used for element tagging, both within an elementSpec (for element selection) and a record syntax (for record representation).
- A schema defines an abstract record structure.

RET.2.1 Element Specification Features and TagSets

An elementSpec may be included in a Present request to specify the desired elements to comprise a retrieval record. For example, the origin might request that the retrieval record consist of the two elements ‘author’ and ‘title’. The elementSpec may express this in one of two ways:

- An *element set name* (a primitive name) might be defined, for example ‘authorTitle’, whose definition means “present the author and title.”
- A dynamic specification may be used, allowing the origin to select arbitrary elements, dynamically.

The use of an element set name as an elementSpec has a significant limitation: one would need to be defined for every possible combination of elements that might be requested.

For Z39.50 version 2, only the primitive form is allowed; the elementSpec must be an element set name (whose ASN.1 type is VisibleString). Version 3 allows the elementSpec to alternatively assume the ASN.1 type EXTERNAL (thus referencing an external definition, 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.2.1.1. Simple Numeric Tags

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

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

In this example, each element requested is represented by an integer. Both origin and target are assumed to share a common definition, a *tagSet*, which assigns integers to elements. The integer is the name, or *tag*, of the element. In this example, the tagSet might assign the integers 1 to ‘title’ and 2 to ‘author’.

RET.2.1.2 String Tags

It is not always desirable to restrict element tags to integers. *String* tags are useful for some applications, so the element request might take the slightly more complex form:

```
ElementRequest ::= StringOrNumeric
```

Note that StringOrNumeric is a type defined within, and exported by Z39-50-APDU, defined as:

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

In this case, the tagSet might declare that “author may also be referenced by the string tag ‘author’, and title by ‘title’.”

RET.2.1.3 Tag Types

Often it will be necessary (or useful) to request elements not all of whose tags are defined by a single tagSet. This capability presents an important benefit, allowing multiple name spaces for tags, so that tagSet definitions may be developed independently. However, this capability requires that tags be qualified by reference to tagSet.

A schema definition (see RET.2.2) may assign an integer to identify a tagSet (it identifies the tagSet only within the context of the schema definition). This tagSet identifier is called a *tagType*. Note that a tagSet definition is a registered object and thus is persistently identified by an object identifier. The (integer) tagType is used as a short-hand identifier.

Extending the above example to incorporate tagTypes, the elementRequest could be defined as:

```
ElementRequest ::= SEQUENCE{
        tagType          [1] IMPLICIT INTEGER,
        tagValue         [2] StringOrNumeric}
```

RET.2.1.4 Tag Occurrence

A database record often contains recurring elements. An origin might want the Nth occurrence of a particular type of element (e.g., "the fourth image"). To introduce recurrence into the above example, the elementRequest could be defined as:

```
ElementRequest ::= SEQUENCE{
        tagType          [1] IMPLICIT INTEGER OPTIONAL,
        tagValue         [2] StringOrNumeric,
        tagOccurrence    [3] IMPLICIT INTEGER}
```

RET.2.1.5 Tag Paths

A database record is not necessarily a flat set of elements, it may be a hierarchical structure, or tree (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). This example introduces the concept of a tag path, which is simply a nested sequence of tags (each tag within the sequence is qualified by a type and occurrence). A tag path can be incorporated by replacing the first line of ASN.1 in the previous example, with:

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

RET.2.1.6 VariantRequests

Finally, the origin may wish to qualify an elementRequest with a variantRequest, to specify a particular composition (e.g., PostScript), language, character set, 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.2.3.

RET.2.2 Schema and Abstract Record Structure (ARS)

A database schema represents a common understanding shared by the origin and target of the information contained in the records of the database represented by schema. The primary component of a schema is an *abstract record structure,* (ARS). It lists schema elements in terms of their tagPaths, 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 elements within the record; see RET.2.2.5.)

An ARS is defined in terms of one or more tagSets. The schema itself may define a tagSet, and may also refer to externally defined tagSets. In the simple example of an ARS that follows (Table A14-1), assume that the following tagSet has been defined:

Table A14-1: Simple Example of an Abstract Record Structure

Tag	Element	Recommended dataType
1	title	InternationalString
7	name	InternationalString
16	date	GeneralizedTime
18	score	INTEGER
14	recordId	InternationalString
<*locally* defined string *tag*>	objectElement	InternationalString or OCTET STRING

In the ARS example shown in Table A14-1, each "schema element" refers to an element from the above tagSet. 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 7, 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.

RET.2.2.1 Relationship of Schema and TagSet

In the example, at first glance it appears there need not be separate tables for tagSet and ARS, they could be combined into a single table. When the tagSet is defined within a schema, then there may be no need to distinguish between the tagSet and schema. However, the tagSet might instead be defined externally and referenced by the schema.

A schema may define a tagSet as in the example, 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. But some of the elements in the example might also be included in a different schema. For example, another schema might also define title and name, and that schema should be able to use the same tags. For this purpose, tagSets may be registered, independent of schema definitions.

It is anticipated that there will be several, but not a large number of tagsets defined, and that many 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 than one tagSet defined because it would be difficult to manage a single tagSet that meets the needs of all schemas.)

RET.2.2.2 Tag Types

As noted in RET.2.1.3, within a Present request or Present response, elements are identified by their tag, and tags are qualified by *tag type*. The tag type is an integer, identifying the tagSet to which it belongs. A schema lists each tagSet referenced in its ARS and designates an integer to be used as the tag type for that tagSet.

Z39.50 currently defines two tagSets, tagSet-M and tagSet-G. These are described in RET.3.4. TagSet M includes elements to be used primarily to convey meta-information about a record, for example dateOfCreation; tagSet-G includes primarily generic elements, for example 'title', 'author'.

Among the schema elements defined in the example above, title and name are defined in tagSet-G; date, score, and recordId are defined in tagSet-M.

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

1 --> tagSet-M
2 --> tagSet-G
3 --> locally defined tags (intended primarily for string tags, but numeric tags are not precluded).

In the notation below, where (x,y) is used, 'x' is the tagType and 'y' is the tag. In Table A14-1 the following column would be inserted on the left:

TagPath
(2,1)
(2,7)
(1,16)
(1,18)
(1,14)
(3,<locally defined string tag>)

RET.2.2.3 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. 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 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.) The ARS would be as follows (definition column omitted):

Tag path	Element	Mandatory?	Repeatable?
(2,1)	Title	yes	no
(2,7)	Name	no	yes
(1,16)	Date	no	no
(1,18)	Score	no	no
(1,14)	RecordId	no	no
(3, <stringTag>)	Object Element	yes	yes

RET.2.2.4 Structured Elements

In the preceding examples the ARSs are flat; all elements are data- elements, i.e., leaf-nodes. In the ARS in Table A14-2, hierarchy is introduced; the ARS includes structured elements (i.e., elements whose tagPath has length greater than 1).The ARS Table A14-2 is part of a schema for a database in which each record describes an information resource. It assumes the following tagSet:

Table A14-2: ARS with Hierarchy

Tag	Element Name	Recommended DataType
25	linkage	InternationalString
27	recordSource	InternationalString
51	purpose	InternationalString
52	originator	InternationalString
55	orderProcess	InternationalString
70	availability	(structured)
90	distributor	(structured)
94	pointOfContact	(structured)
97	crossReference	(structured)

The notation (x,y)/(z,w) is used below to mean element (z,w) is a sub-element of 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. In this example, the schema designates that the tagType for the above defined tagSet is 4.

Several elements in this ARS are (implicitly) imported from tagSet-G (those with tagType-2). These are: title, abstract, name, organization, postalAddress, and phoneNumber.

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.

RET.2.3 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 (*variantSpec*) takes the form of a variantRequest, appliedVariant, or supportedVariant. In all cases, a variantSpec is a sequence of *variantComponent*s, 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 attribute values within the attribute lists). The variant set definition *variant-1* is defined in Annex 11, VAR, and is described in detail in RET.3.3.

RET.2.4 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 (listed at the beginning of Annex 5, REC)
- Explain (REC.1)
- OPAC and Summary (REC.3 and REC.4)
- Extended Services (REC.6).

Generic record syntaxes are further categorized: they are *structured* or *unstructured*. Structured record syntaxes are able to identify TagSet elements. GRS-1, a generic, structured syntax, is defined in REC.5, and is described in detail in RET.3.2. SUTRS (Simple Unstructured Text Record Syntax) is a generic, unstructured syntax, defined in REC.2.

RET.3 Retrieval Objects Defined in this Standard

In the remainder of this Annex, detailed descriptions are provided for the following retrieval objects defined in this standard: element specification format eSpec-1, record syntax GRS-1, variant set variant-1, and tagSets tagSet-M and tagSet-G. Within these descriptions it is assumed that these objects are used together; for example, in the description of eSpec-1 it is assumed that GRS-1 is to be used as the record syntax. In general, however, no such restriction applies; eSpec-1 may be used as an element specification in conjunction with SUTRS for example.

RET.3.1 Element Specification Format eSpec-1

The element specification format eSpec-1 is defined in Annex 10, ESP. An element specification taking this form is basically a set of elementRequests, as seen in the last member of the main structure:

```
elements [4] IMPLICIT SEQUENCE OF                    ElementRequest ......
```

Each elementRequest may be a “simple element” or a “composite element,” as distinguished by the ElementRequest definition:

```
ElementRequest::= CHOICE{
        simpleElement           [1] ...
        compositeElement        [2] ...
```

Simple elements are described in RET.3.1.1. A composite element is constructed from one or more simple elements, described in RET.3.1.2. Note however an elementRequest that takes the form of simpleElement might actually result in a request for multiple elements (see RET.3.1.1.3).

The element specification may include additional elementRequests, resulting from ‘elementSetNames’ in the first member of the main sequence. All elementRequests resulting from ‘elementSetNames’ are simple elements.

Also included in the main structure are a default variantSetId and a default variantRequest. These are described in RET.3.1.1.5.

RET.3.1.1 Simple Element

A request for a simple element consists of the tagPath for the element, together (optionally) with a variantRequest. The tagPath identifies a node of the logical tree (or possibly several trees) representing the hierarchical structure of the abstract database record to which the element specification is applied.

A tagPath is a sequence of nodes from the root of a tree to the node that the tagPath represents, where each node is represented by a tag. The end-node of a tagPath might be a leaf-node containing data, or a non-leaf node; in the latter case, the request pertains to the entire subtree whose root is that node, and GRS-1 will present the subtree recursively (see RET.3.2.1.1).

RET.3.1.1.1 Tag

Each tag is qualified by a tagType. Thus a tag consists of a tagType and a tagValue. (A tag is further qualified by its “occurrence”; see RET.3.1.1.2.) Each tagType is an integer, and each tagValue may be either an integer or string.

Every tag along a tagPath is assumed to have a tagType, either explicit or implicit; it may be supplied explicitly within the specification, and if it is omitted, a default applies (the default should be listed within the schema in use). Tags along a tagPath may have different tagTypes.

RET.3.1.1.2 Occurrence

Each node along a tagPath is distinguished not only by its tag, but also by its occurrence among siblings with the same tag. A record might contain recurring elements, and the origin might wish to request the Nth occurrence of a particular element (e.g., “the fourth image”). The specification of the “occurrence” of a node may be omitted, in which case it defaults to 1. Occurrence may explicitly be specified as “last” (this capability is provided for the case where the origin does not know how many occurrences there are, but however many, it wants the last).

RET.3.1.1.3 Multiple Simple Elements

In some cases a 'simpleElement' request (within the ElementRequest structure) results in multiple simple elements. This may occur in the following cases: If a tagPath identifies a non-leaf node, the request represents the entire subtree (it is logically equivalent to individual simple requests for each subordinate leaf-node).

- 'occurrence' may be specified as 'all', meaning "all nodes with a given tag."
- 'occurrence' may be specified in the form of a range (e.g., 1 through 10).
- The tagPath may include a wild card (see RET.3.1.5) in lieu of a specific tag.

RET.3.1.1.4 Wild-cards

A tagPath may be viewed as an expression containing tags and wild cards. There are two types of wild cards, wildThing and wildPath, described in RET.3.1.1.4.1 and RET.3.1.1.4.2.

For this discussion of wild-cards, consider the sample record whose hierarchical structure is shown in Figure A14-1 below.

Each cell in the diagram represents an element whose tagPath is indicated within the cell. The numbers within the tagPath are tagValues; for simplicity, tagTypes are omitted, and assumed all to be the same. Leaf-nodes are highlighted by double-lined cells.

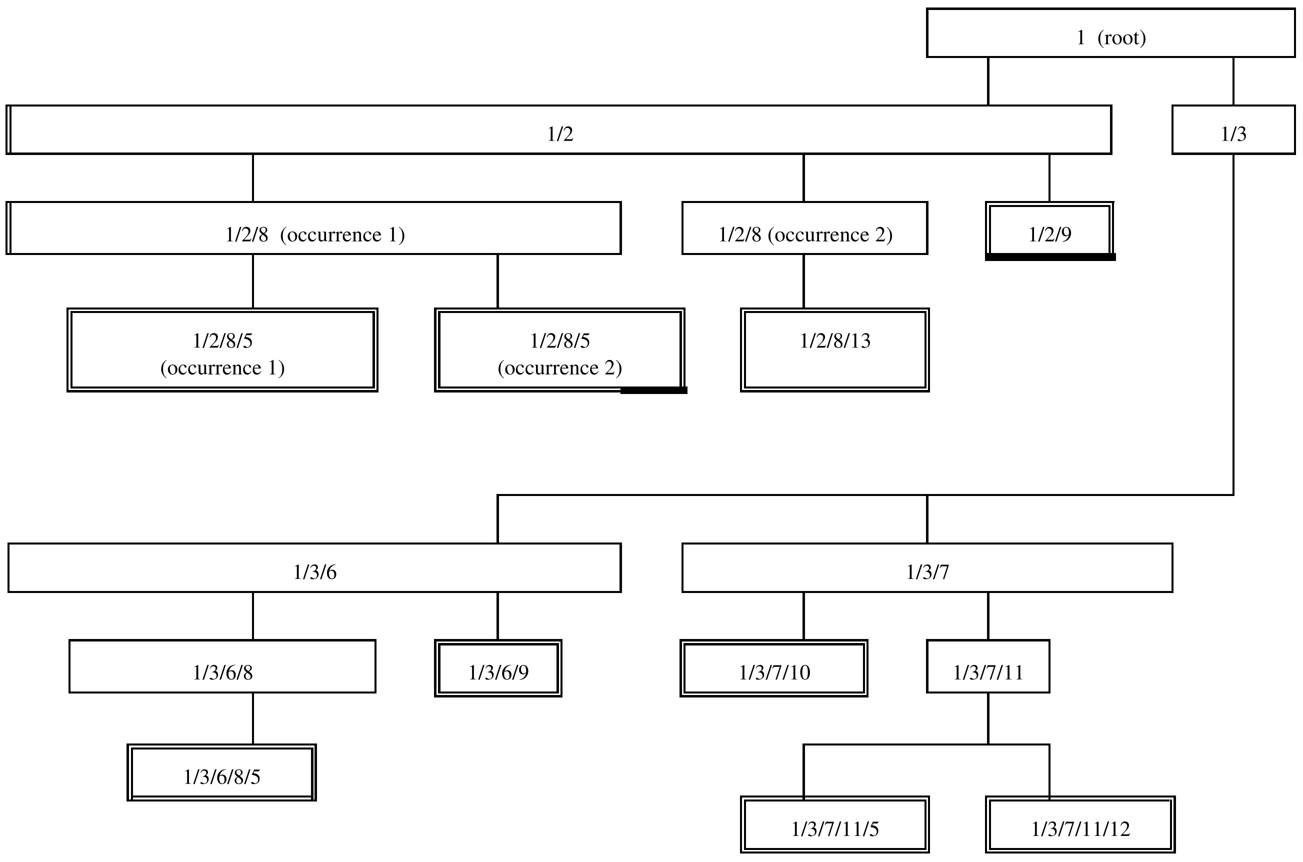
For example, the tagPath 1/3/7 represents the (non-leaf-node) element with tag 7 subordinate to the element with tag 3 subordinate to the element with tag 1. 1/3/7/11/12 represents the element whose (leaf-) node has tag 12.

RET.3.1.1.4.1 WildThing

A tagPath expression may include the wild card 'wildThing' in lieu of a tag. WildThing takes the form of an occurrence specification. For example, the tagPath expression '1/2/wildThing (occurrence 3)' would represent the node 1/2/9, because it is the third child of the node 1/2.

The expression '1/wildThing (occurrence 2)' would be equivalent to the path 1/3 (it refers to the entire subtree whose node has tag 3).

Figure A14-1: Sample Record Illustrating Hierarchical Structure and Wildcards



RET.3.1.1.4.2 WildPath

A tagPath expression may include the wild card ‘wildPath’ in lieu of a tag. WildPath matches any sequence of tags, along any path such that the tag following wildPath in the expression follows that sequence in the matched path. For example, either of the expressions ‘wildpath/5’ or ‘1/wildPath/5’ would result in all paths ending in 5. It would match:

1/2/8 (occurrence 1)/5 (occurrence 1)
1/2/8 (occurrence 1)/5 (occurrence 2)
1/3/6/8/5, and
1/3/7/11/5

The expression ‘1/2/wildPath/5’ would match the first two listed above, and the expression ‘1/3/wildPath/5’ would match the last two.

RET.3.1.1.5 Variant Request

Each request for a simple element may optionally include a variantRequest. Note that the main structure of eSpec-1 optionally includes ‘default-VariantRequest’. If the element request does not include a variantRequest then ‘defaultVariant-Request’ applies if it occurs in the main structure. If the element request does not include a variantRequest and ‘defaultVariantRequest’ does not occur in the main structure, there is no variant request associated with the element request.

The main structure also optionally includes ‘defaultVariantSetId’. A variant specification may or may not include a variantSetId. If the element request includes a variantRequest which does not include a variantSetId, then ‘defaultVariantSet’ applies. (If the element request includes a variantRequest which does not include a variantSetId, and if ‘defaultVariantSet’ does not occur in the main structure then the variantRequest is in error.)

RET.3.1.2 Composite Elements

An elementRequest for a compositeElement takes the form of a list of simple elements (as described in RET.3.1; alternatively, the simple elements may be specified by one or more element set names), a delivery tag, and an optional variantRequest. The simple elements are to be combined by the target to form a single (logical) element, to which the (optional) composite variant is to be applied, and the target is to present the element using the supplied delivery tag.

RET.3.2 Generic Record Syntax GRS-1

A GRS-1 structure is a retrieval record representing a database record. Its logical content is a tree representing the hierarchical structure of the abstract database record, or a sequence of trees if the abstract record itself does not have a root.

RET.3.2.1 General Tree Structure

The top level “SEQUENCE OF TaggedElement” might be a single instance of TaggedElement, representing the root of a single tree representing the record (in the degenerate case, the record consists of a single element). Alternatively, the top-level SEQUENCE OF might contain multiple instances of TaggedElement, in which case there is no single root for the record; the record is represented by multiple trees, any or each of which might be a single element (thus the GRS-1 structure may represent a flat sequence of elements).

Any leaf-node within the GRS-1 structure might correspond to an individual elementRequest that is included in the corresponding eSpec-1 element specification. A non-leaf node may correspond to an elementRequest; if an eSpec-1 elementRequest tagPath ends at a non-leaf node, then the request is for the entire subtree represented by that node.

RET.3.2.1.1 Recursion and SubTrees

Each instance of TaggedElement may, via recursion, contain a subtree. Beginning at the root of the tree (or at one of the top level nodes) TaggedElement identifies an immediately subordinate node, via tag and occurrence. If the CHOICE for ‘content’ is ‘subtree’, then the identified node is a non-leaf node: ‘subtree’ is itself defined as SEQUENCE OF TaggedElement, so the next level of nodes is thus defined. Recursion may be thus used to describe arbitrarily complex trees.

RET.3.2.1.2 Leaf-nodes

Along any path described by the GRS-1 record, eventually a leaf-node is encountered (‘content’ other than ‘subtree’). The content of the leaf-node is one of the following:

- Data; see RET.3.2.2.
- Empty, for one of the following reasons:

  — The requested element does not exist.
  — It exists, but there is no data.
  — The elementRequest specified (via a variant-1 variantRequest) that no data was to be returned. (This is probably because only meta-data was desired. So it is likely that the variantRequest also requested meta-data, and that meta-data accompanies this node; see RET.3.2.3.)

- A diagnostic.

RET.3.2.2 Data

When a leaf-node contains data, then 'content' is one of the following ASN.1 types: OCTET STRING, INTEGER, GeneralizedTime, EXTERNAL, InternationalString, BOOLEAN, OBJECT IDENTIFIER, or IntUnit. That is, the CHOICE for ElementData is one of these, and the actual data must assume the chosen type. An appliedVariant may also be indicated, by including appliedVariant from the main structure.

RET.3.2.3 Meta-data

When a leaf-node contains data or is empty, 'metaData' may be included, containing meta-data for the element. The meta-data may be included along with the data, or in lieu of the data if the elementRequest asked that no data be returned (i.e., 'content' is 'noDataRequested'). Meta-data would not be included when 'content' is 'elementNotThere', 'elementEmpty', or 'diagnostic'.

MetaData for a leaf-node may be any or all of the following:

- *usageRight*: the target may declare that the element is freely distributable, or that restrictions apply. In the latter case, the target supplies either a restriction in the form of a text message, or a license pointer.
- *hits*; see RET.3.2.3.1.
- *displayName*: A name for the element, suggested by the target, for the origin to display.
- *supportedVariants*; see RET.3.2.
- *message*: A message for the origin to display to the user, associated with this element.

There is also one case where meta-data may be included for a non-leaf node:

- *seriesOrder*; see RET.3.2.3.2.

RET.3.2.3.1 Hits

Associated with an element may be one or more *hit vectors*. Each points to a fragment within the element. Each such fragment bears some relationship to the search that caused the record (to which the element belongs) to be included in the result set (from which the record is being presented). Note that the association of a hit vector to an element is meaningful only within the context of that search.

A hit vector may optionally include a 'satisfier': for example, a term from the query, which occurs within that fragment of the element (to which the hit vector points).

The target might return hit vectors along with an element, so that the origin may be able to quickly locate the satisfying portions of the element, and perhaps even highlight the satisfier(s) for display to the user.

The target might return part of an element and include hit vectors, some of which point within the retrieved portion, and others which point to fragments not included, to indicate to the origin what fragment to request to retrieve other relevant parts of the element.

A hit vector may include location information: offset (location within the element where the fragment begins) and length. Both are expressed in terms of IntUnit, so for example, the location information might indicate an offset of "page 10" and length of "one page," meaning that the satisfier occurs on page 10 (or that the fragment is page 10).

Note: if there are multiple hit vectors with the same satisfier, occurring on the same page, and if the target wishes to indicate 'rank' (see below), it will need to use a unit with finer granularity than 'page'.

The hit vector may also include 'rank', relative to the other hits occurring within this set of hitVectors. Rank is a positive integer with a value less than or equal to the number of hit vectors. More than one hit may share the same rank.

Finally, the target may assign a token to the hit vector, which points to the fragment associated with the hit. The origin may use the token subsequently, but within the same Z-association, within a variantRequest (in an elementRequest) to retrieve (or to refer to) the fragment.

The target might provide location information, or a token, which may be used subsequently to retrieve the specific fragment. The target might provide both location information and a token: for example, the location information might indicate "page 10"; the origin may subsequently retrieve the pages before and after, inclusive (i.e., pages 9-11). If the target also supplies a token, the origin might retrieve the "previous fragment" or "following fragment."

Location information is always variant-specific. A token, however, may be variant-specific or variant-independent. The origin might request "hits: non-variant-specific" for an element (via variant-1), and specify 'noData'. The hit vectors returned would be variant-independent (thus only a token, and no location information, would be included in each hit vector). The origin could subsequently use a token in an elementRequest to retrieve the corresponding fragment, independent of what variantRequest was included in the elementRequest.

The origin might request 'hits: variant-specific' for an element, for a particular variant. The target might return location information or tokens, or both, but in any case, the hit vectors would apply only for that variant. The origin could subsequently use either the location information or token in an elementRequest to retrieve the corresponding fragment, but only when specifying that variant.

As an alternative to hit vectors, see RET.3.3.1.8, Highlighting.

RET.3.2.3.2 Series Order

The target might include the meta-data ‘seriesOrder’ (for a non-leaf node only). It indicates how immediately-subordinate elements with the same tag are ordered. Values are listed in TAG.2.1, but may be overridden by the schema.

The values are the same as those for elementOrdering (see RET.3.4.1.2.3) which applies at the record level (i.e., it applies throughout the record, and pertains wherever sibling elements with the same tag occur).

RET.3.3 Variant Set Variant-1

This section describes the variant set variant-1.

RET.3.3.1 variant-1 Classes

This section describes the classes, types, and values defined for the variant set variant-1.

RET.3.3.1.1 VariantId

Variant-1 class 1, ‘variantId’, may be used to supply an identifier for a variant specification. (There is only one type within class 1, so the variantId is always class 1, type 1). It is a *transient* identifier; it may be used to identify a particular variant specification during a single Z-association. (A variantId should not be confused with *variant set id*, which identifies a *variant set definition*.)

A variantId may be included within a supportedVariant, variantRequest, or applied-Variant. The variantList for an element may be supplied by the target (see 3.3.2). It consists of a list of supportedVariants for the element. Each may include a variantId, which may be used subsequently by the origin within a variantRequest (within an elementRequest), to identify that supportedVariant (i.e., that variant form of the element), in lieu of explicitly constructing a variant. A variantId may be used within an appliedVariant, supplied by the target in case the origin wishes to use it in a subsequent request, possibly overriding some of the variant parameters.

RET.3.3.1.2 BodyPartType

Variant-1 class 2, ‘BodyPartType’, allows representation of the structure, or “body part type,” of an element. It may be used within a supported-Variant, variantRequest, or appliedVariant.

There are three types: type 1 is ianaType/subType, for content types registered with IANA (Internet Assigned Numbers Authority). Type 2 is for body part types registered by the Z39.50 Maintenance Agency (type 2 is used generally for formats that have not yet been otherwise officially registered). Type 3 is for bilaterally agreed upon body part types.

Following are some of the IANA contentType/Subtypes registered:

Type	Subtype
text	plain
	richtext
	tab-separated-values
application	octet-stream
	postscript
	oda
dx	wordperfect5.1
	pdf
	zip
	macwriteii
	msword
image	jpeg
	gif
	ief
	tiff
audio	basic
video	mpeg

PostScript, for example, would be indicated by the triple (2,1, 'application/postscript'). SGML is not registered yet by IANA, so it is registered as a Z39.50 body part type. It may be indicated by (2,2, 'sgml/<dtd>') where <dtd> is the name of the SGML dtd.

A Z39.50 body part type will be registered only if it is not registered as an IANA type. If it is subsequently adopted by IANA, it is recommended that it be referenced as such.

RET.3.3.1.3 Formatting/Presentation

Variant-1 class 3, 'formatting', may be included within a variantRequest, appliedVariant, or supportedVariant. It indicates additional formatting parameters such as line length, lines per page, font style, and margins.

RET.3.3.1.4 Language/CharacterSet

Variant-1 class 4, 'language/characterSet', may be included within a variantRequest, appliedVariant, or supportedVariant. It indicates language and/or character set.

RET.3.3.1.5 Piece

Variant-1 class 5, 'piece' may be included within a variantRequest (type 1) or appliedVariant (type 2), to refer to a specific *piece* or *fragment* of an element.

The origin may use type 1 to request:

- A fragment beginning at the beginning of the element ('start');
- The 'next' fragment (relative to the fragment indicated by targetToken, see type 7);
- the 'previous' fragment;
- the 'current' fragment (the fragment indicated by targetToken);
- the 'last' fragment (within the element).

The target may use type 2 to indicate that the presented fragment:

- begins at the beginning of, but is not the whole element ('start');
- neither starts at the beginning of, nor ends at the end of the element ('middle');
- does not begin at the beginning of, but ends at the end of the element ('end');
- ends at the end of the element, but the element may grow in the future ('endForNow'); or
- is the 'whole' element.

The target may use types 3, 4 (or 5), and 6 in lieu of type 2, to indicate the 'start' and 'end' (e.g., starts at page 1 and ends at page 100) or 'start' and 'howMuch' (e.g., starts at page 1, 100 pages) of the fragment and optionally, a 'step' size. For example, the target could indicate that the fragment starts at byte 10,000 and ends at byte 20,000 (in this case a step of 1 would be indicated, or implied if 'step' is omitted); or it starts on page 100, ends on page 200, and includes every 5th page.

Similarly, the origin may use types 3, 4 (or 5), and 6 to request a fragment. In a variantRequest these types may be used to further qualify a fragment indicated by types 2 and 7. For example, the request might specify a targetToken, previous fragment (5,1,3), as well as a start and end, in which case the start and end are relative to the indicated fragment, i.e., relative to the fragment immediately prior to that indicated by the target token.

The target may use type 7 in an appliedVariant to supply a token as an identifier of the supplied fragment, and the origin may subsequently use the token in a variantRequest to identify that fragment.

RET.3.3.1.6 MetaData Requested

Variant-1 class 6, 'meta-data requested' may be included within a variantRequest, to request meta-data associated with an element.

The origin might want to know, for example, the cost to retrieve a particular element in PostScript, as well as the page count (of the PostScript form of the element). The following variant specifiers would be included within the variantRequest for that element:

```
    (2,1, 'application/postscript')-- PostScript
    (6,1, NULL)-- cost, please
    (6,2, Unit:pages)-- size in pages, please
    (9,1, NULL)-- no data (just the
-- above metaData)
```

Alternatively, a variantId might be used in place of a set of explicit specifiers (i.e., in place of the PostScript specifier, in this example) if the origin knows the variantId of a variant for which it wants cost or size information. (Although if the origin knows the variantId, it may already have cost or size information because it may have obtained that id within a variantList, and if so, the target may have included the cost and page information within the supportedVariant.)

The origin might also ask for the location of hits within the element (see RET.3.2.3.1). An element might have hits that are specific to a variant, and may also have non-variant-specific hits. The request above might also ask for hits specific to the particular variant (i.e., PostScript), using (6,3, NULL) or non-variant-specific hits, using (6,4, NULL). In either case, the request is for the target to return hit vectors within the retrieved GRS record.

The origin may request that the target supply the variant list for an element via the specifier (6,5, NULL). The target would supply the variant list (consisting of a list of supportedVariants) within the GRS structure (not within the appliedVariant). See RET.3.3.2.

The origin may use (6,6, NULL) to inquire whether a particular variant is supported. An example is provided in RET.3.3.2.

RET.3.3.1.7 Meta-data Returned

Variant-1 class 7, 'meta-data returned' may be included within an appliedVariant or supportedVariant. There are several categories of element MetaData. Those of class 7: cost, size, integrity, and separability, are singled out for representation within variant-1, because the target may include those within a supportedVariant. Other metaData, including hits and variantList, are included within the GRS-1 structure. Hits are described in RET.3.2.3.1.

RET.3.3.1.8 Highlighting

Variant-1 class 8, 'highlighting', may be included within a variantRequest or an appliedVariant. Highlighting may be used as an alternative, or in addition, to hit vectors, described in RET.3.2.3.1.

The origin may include 'prefix' and 'postfix' in a variantRequest to request that the target insert the specified strings into the actual data surrounding hits so that the origin, upon retrieving the data, may simply locate the strings for fast access to the hits. The origin may use 'server default' in lieu of 'prefix' and 'postfix' to indicate that the target should select the strings for highlighting.

The target may include 'prefix' and 'postfix' in an appliedVariant to indicate the strings used within the element for highlighting hits.

RET.3.3.2 VariantList

The thoroughness of the variantList supplied by the target may depend on the implementation. For example, for an element (representing a document) that the target provides in PostScript, consider the following cases:

- The document might already exist in print format, and the target might support only that single PostScript variant.
- The target might support a few variant forms, varying by language.
- The target might support many variant forms; varying by language.
- The target might support many variant forms, varying by language, and also varying by formatting/presentation parameters, including lines per page, font style, etc.

The target might list a single supported-Variant in the variant list for the element, indicating that the element is available in PostScript. In that case the origin cannot necessarily conclude which of the above cases applies. The target might instead list three supportedVariants, each indicating PostScript and a language. In that case, it may be reasonable for the origin to surmise that the element is available in those three languages only, but the origin probably cannot deduce which formatting parameters apply. The target might further indicate one or more formatting parameters within each supportedVariant. Again, the extent to which the origin may deduce what other variations are supported will depend on the implementation.

The origin may explicitly inquire whether a particular variant is supported, by constructing the desired variant (including all of the desired formatting parameters, etc.) and indicating "is variant supported?" using the triple (6,6, NULL). The variantRequest might also request that the target provide cost (6,1, NULL) and size (6,2, NULL) information if the variant does exist. The target would respond that the requested variant is or is not supported by supplying an appliedVariant (with the element) with the same parameters, and including the triple (7,5, TRUE or FALSE). If the target indicates TRUE (that the variant is supported) it may also supply a variantId that the origin may then use to request the variant.

The origin may construct a variantRequest that includes a variantId along with additional variant specifiers. Suppose the target lists the following supportedVariant:

(1,1, <variantId>)	-- identifies this variant
(2,1, 'application/postscript')	-- in PostScript
(4,1, 'por')	-- language: Portuguese

The element is thus available in PostScript, in Portuguese. The origin may submit a variant-Request consisting of only:

(1,1, <variantId>)

to request the element in PostScript, in Portuguese.

Suppose, instead, the target lists the following supportedVariant:

```
(1,1, <variantId>)                -- identifies this variant
(2,1, 'application/postscript')   -- in PostScript
```

Thus the target indicates that the element is available in PostScript, but no other variant information is provided.
The origin may submit a variantRequest consisting of only:

```
(1,1, <variantId>)
(4,1 'por')
```

Again, this is to request the element in PostScript, in Portuguese.

Or, the origin may submit the following variantRequest:

```
(1,1, <variantId>)
(4,1, 'por')
(4,2, 84)            -- Portuguese character set
(5,3, page 1)        -- begin on page 1
(5,4, page 100)      -- end on page 100
```

to request the element in PostScript, in Portuguese, Portuguese character set, pages 1-100.

RET.3.4 TagSets Defined in the Standard

Annex Tag defines two tagSets, tagSet-M (for elements which convey meta- and related information about a record) and tagSet-G (primarily for generic elements). These two tagSets are described in RET.3.4.1 and RET.3.4.2.

RET.3.4.1 TagSet-M

TagSet-M defines a set of elements that the target might choose to return within a retrieval record, even though the element was not requested and in fact is not actually information contained within the database record. Rather, it is information *about* the database record, retrieval record, or result set record. Within a GRS-1 record, the target returns tagSet-M elements in exactly the same manner that it returns elements from any other tagSet.

TagSet-M elements fall into three categories.

1. Meta-information about the database record:
   - processingInstructions
   - recordUsage
   - restriction
   - userMessage
   - url
   - local control number
   - creation date
   - dateOfLastModification
   - dateOfLastReview

2. Elements defined to facilitate the construction and processing of the retrieval record:
   - schemaIdentifier
   - elementsOrdered
   - elementOrdering
   - defaultTagType
   - defaultVariantSetId
   - defaultVariantSpec
   - record
   - wellKnown
   - recordWrapper

3. Elements pertaining to the record's entry in the result Set:
   - rank
   - score

RET.3.4.1.1 Meta-Information

The definitions for these elements are provided in TAG.2.1. Any of these elements may or may not actually occur within the database record. However, it is emphasized that these elements *describe the database record*; they do not pertain to elements within the database record which may in fact be meta-information about some object other than the record itself.

For example, tagSet-M element ‘url’ refers to a URL for the database record. The database record itself may contain URLs for resources that the record describes; tagSet-M element ‘url’ does not pertain to those.

RET.3.4.1.2 Information about the Retrieval Record

RET.3.4.1.2.1 schemaIdentifier

A retrieval record is meaningful only within the context of a schema definition. In many (perhaps most) cases the target may reasonably expect that the origin knows which schema definition applies to a particular retrieval record. In those cases the target need not explicitly identify the schema. This element is provided for cases where there is a possibility of uncertainty about which schema applies.

This element is also useful for retrieval records that include subordinate or nested records which are defined in terms of different schemas. See RET.3.4.1.2.5.

This element, if provided, should normally occur as the first element within the retrieval record (or within a subordinate or nested record) and for that reason is assigned tag 1, in case the target wishes to present elements in numerical order (see RET.3.4.1.2.2).

RET.3.4.1.2.2 elementsOrdered

This is a BOOLEAN flag indicating whether the elements of the retrieval record are presented in order by tag. The ordering is described in Annex 12, TAG.2.1. This element is defined because it may be useful for an origin to know whether elements are presented in order when trying to locate a particular element within the retrieval record.

This element, if provided, should normally occur as the first element within the retrieval record, or the second if schemaIdentifier is provided, and for that reason is assigned tag 2.

RET.3.4.1.2.3 elementOrdering

For a retrieval record containing recurring elements, i.e., sibling elements with the same tag, the target might present these elements according to some logical order, for example, chronological, increasing generality, concentric object snapshots, or normal consumption (i.e., pages, frames). This element indicates the order; values are listed in TAG.2.1. Note that the values are the same as those for seriesOrder (see RET.3.2.3.2) which applies at the element level, i.e., it pertains to sub-elements of an element. This element, element-Ordering, applies at the record level, i.e., it applies throughout the record, and pertains wherever sibling elements with the same tag occur.

RET.3.4.1.2.4 Defaults (tagType, variantSetId, and variantSpec)

defaultTagType, if provided, is the assumed tagType for presented elements where the tagType is omitted. It is defined solely to allow simplification of the retrieval record. If there is a predominant tagType within the retrieval record, this meta-element allows the target to omit the tagType for those elements with that tagType.

Note that the schema may also list a default tagType. If so, then defaultTagType, if it occurs, overrides the schema-listed default. If the schema does not list a default tagType, and if this element does not occur, then every tag within the retrieval record must include a tagType.

defaultVariantSetId is the assumed variantSetId for appliedVariants within the retrieval record that omit the variantSetId. defaultVariantSpec, if provided, is the assumed appliedVariant for all elements within the retrieval for which an appliedVariant is not provided. The schema may also list a default variantSetId and/or appliedVariant. If so, then these elements, if they occur, override the schema-listed default. If the schema does not list a default variantSetId and defaultVariantSetId is not provided, then every appliedVariant within the retrieval record must include a variantId. If the schema does not list a default appliedVariant and defaultVariantSpec is not provided, then for elements within the retrieval record for which an appliedVariant is not supplied, no appliedVariant is assumed to apply.

RET.3.4.1.2.5 Record

The tagSet-M element ‘record’ may be used to present nested or subordinate records.

A retrieval record represents a single database record, but that database record may contain elements which in turn represent database records (possibly replicated from a different database). For example, a database may contain records representing queued database updates. Each such record might contain a set of database records to be contributed to some other database. As another example, an OPAC database might have records defined so each includes a bibliographic record and a corresponding holdings record, and the holdings record in turn might include a series of circulation records.

It is important to note that although a single retrieval record may include an arbitrary number of subordinate records, or arbitrarily nested records, the retrieval record nevertheless represents a single result set record.

A subordinate (or nested) record defined in this manner may be presented according to a schema different from the schema applying to the retrieval record. The tagSet-M element schemaIdentifer may be included within the element representing a record, and if so, it applies only within that element.

RET.3.4.1.2.6 wellKnown

Some schema developers anticipate that for certain elements, different targets will want to provide several alternative forms of the element. The element 'wellKnown' is defined in order to support this flexibility.

Suppose a schema defines the element 'title'. The intent may be that the target simply return a single value, what the target considers to be the title. In that case, 'title' should be a leaf-node defined from tagSet-G, and 'wellKnown' does not apply.

But suppose the target wishes to return the element 'title' encompassing several forms of the title, including one that the origin will recognize to be the default in case it does not understand any of the others (in which case it may ignore all except the default, or may still display them to the end-user, who might understand them even if the origin does not). The origin returns the single element 'title', which is structured into the following sub-elements:

- the default title
- 'abbreviatedKeyTitle'
- 'formerTitle'
- 'augmentedTitle'
- 'romanizedTitle'
- 'shortenedTitle'.

The additional forms of title (i.e., those other than the default title) might use the above string tags, locally defined, or they may be known tags defined in other tag sets. However, the default title has a distinguished integer tag that is assigned to the tagSet-M element wellKnown, to distinguish it.

The element wellKnown is thus always subordinate to a parent element whose semantics are known (e.g., 'title', 'address', 'name'), and the parent element is structured into one or more forms of that element, one of which is a default form, distinguished by the tag for the element wellKnown. The context of the element wellKnown is known from its parent.

RET.3.4.1.2.7 recordWrapper

This element is defined for use in presenting a record with no root (e.g., a flat record, or a record whose hierarchical structure is that of multiple trees). When the origin requests this element, the request is interpreted as a request for the entire record to be presented subordinate to this element. It is defined primarily to be used in conjunction with a variantRequest specifying 'noData', for the purpose of retrieving a skeleton record (i.e., tags only, no data). If a record does have a root, then if this element occurs, the record's real root is presented subordinate to this element.

RET.3.4.1.3 Information about Result Set Record

TagSet-M elements rank and score provide information pertaining to a record's entry in the result Set. A record may have both a *rank* and a *score*. The rank of a result set record is an integer from 1 to N, where there are N entries in the result set (each record should have a unique rank). The score of a result set record is an integer from 1 to M where M is the normalization factor, which may be independent of the size of the result set, and more than one record may have the same score. The normalization factor should be specified in the schema.

RET.3.4.2 TagSet-G

TagSet-G includes generic elements that may be of general use for schema definitions. They are all self-explanatory, except perhaps the element bodyOfDisplay.

RET.3.4.2.1 bodyOfDisplay

The target might combine several elements of a record into this single element, in a display format, for the origin to display to the user.

For a given schema, perhaps for a particular application, some origins may need the target to distinguish all elements in a retrieval record, perhaps because the origin is going to replicate the record. In other cases, the origin is satisfied for the target to package all elements into display format for direct display to the end-user. In either of these cases, bodyOfDisplay is not applicable (in the latter case the target may use the SUTRS record syntax instead of GRS-1).

In some cases though, the origin may need some of the elements distinguished, but is satisfied to have the target package the remaining elements into a single retrieval element for display. In these cases bodyOfDisplay may be useful.

Suppose the target wishes to present twenty elements of a record, but only the first three elements are intended for origin use, and the remaining elements are intended to be transparently passed to the user. Rather than packaging all twenty elements, the target instead may send four elements, where the fourth delivery element packages the latter seventeen original elements, in a display format.

The bodyOfDisplay element is similar to a composite element (as described in RET.3.1.2) in the fact that a single retrieval element packages multiple logical elements. But bodyOfDisplay differs from a composite element in three respects:

- The target, not the origin, selects the subset of elements for packaging.
- In a composite element there may be semantics conveyed by the tag that the origin or user might understand. For example a request for a composite element may ask for the b subfield of the 245 field concatenated with c subfield of 246 sent back as deliveryElement called ‘title’ (there may be some recognizable semantics associated with the tag ‘title’). The bodyOfDisplay element has no semantics other than telling the origin “here is a composite element for display.”
- The resultant element should always be in display format. A composite element may assume display format, but it may also assume other formats, as determined by the variant.

Annex 15 (Informative) PRO: Z39.50 Profiles

This Annex lists Z39.50 profiles approved by the Open Systems Environment Implementors Workshop (OIW) Special Interest Group on Library Applications (SIG/LA).

At the time of publication of this standard, the following profiles have been approved by the OIW SIG/LA:

1. GILS

Application Profile for the Government Information Locator Service: GILS specification for ANSI/NISO Z39.50 as well as other aspects of a GILS conformant server that are outside the scope of Z39.50. The GILS Profile provides the specification for the overall GILS application including the GILS core, which is a subset of all GILS locator records, and which completely specifies the use of Z39.50 in this application.

2. WAIS

WAIS Profile of Z39.50 Version 2 (Version 1.4): Application Profile for WAIS (Wide Area Information Servers) network publishing systems. Based on Z39.50 Version 2 as specified in ANSI/NISO Z39.50-1995.

3. ATS-1

Specifies the use of the attribute set bib-1 within a Z39.50 type-1 query for searching by author, title, or subject, to provide basic search access to bibliographic databases. Its purpose is to ensure that complying origins and targets can provide basic search access to bibliographic databases, similar to the common online catalog systems used in many libraries.

4. Using Z39.50-1992 Directly over TCP

Based on an Internet RFC “Using the Z39.50 Information Retrieval Protocol in the Internet Environment,” this profile addresses (and its scope is limited to):

- Z39.50 layered directly over TCP (without the use of the OSI ACSE, Presentation, or Session protocols).
- Z39.50-1992 (extensions for Z39.50-1995 to be developed). The profile does *not* address Z39.50-1988.
- Communication over the Internet.

For information on how to obtain these documents, refer to: **http://lcweb.loc.gov/z3950/agency**

Annex 16
(Informative)
Designation of Maintenance Agency

Questions concerning the implementation of this standard should be sent to the Z39.50 Maintenence Agency at the Library of Congress, Network Development and MARC Standards Office, Washington, DC 20540; telephone 202-707-6237; e-mail: ndmso@loc.gov.

ICS 35.240.30

Descriptors: documentation, computer applications, bibliographic data bases, information retrieval, data processing, information interchange, application layer, services, protocols.

Price based on 154 pages