ANSI/NISO Z39.50 2001 ISSN: xxxx-xxxx

Information Retrieval (Z39.50):

Application Service Definition and

Protocol Specification

***Status**: This is near-final text of a revision of Z39.50, being prepared for ballot. It incorporates ZIG-approved defect reports, amendments, clarifications, commentaries, implementor agreements editorial changes at the discretion of the editor, and other miscellaneous changes. This document is provided for those who would like to review it, particularly to identify errors so that they may be corrected before ballot. Please send comments to rden@loc.gov.*

Abstract: This standard specifies a client/server based protocol for Information Retrieval. It specifies procedures and formats for a client to search a database provided by a server, retrieve database records, and perform related information retrieval functions. The protocol addresses communication between information retrieval applications at the the client and server; it does not address interaction between the client and the end-user.

A Draft American National Standard
Developed by the
National Information Standards Organization

Published by the National Information Standards Organization
Bethesda, Maryland

NISO Press, Bethesda, Maryland, U.S.A.

DRAFT STANDARD SUBJECT TO CHANGE

Published by
NISO Press
4733 Bethesda Avenue, Suite 300
Bethesda, MD 20814
www.niso.org

Printed in the United States of America
ISSN: xxxx
ISBN: xxxx

This paper meets the requirements of ANSI/NISO Z39.48-1992 (Rainbow 3D 1997) Permanence of Paper.

Library of Congress Cataloging-in-Publication Data

National Information Standard Organization (U.S.)
xxxx

DRAFT STANDARD SUBJECT TO CHANGE

About NISO Standards

NISO Standards are developed by the Standards Committees of the National Information Standards Organization. The development process is a strenuous one that includes a rigorous peer review of proposed standards open to each NISO Voting Member and any other interested party. Final approval of the standard involves verification by the American National Standards Institute that its requirements for due process, consensus, and other approval criteria have been met by NISO. Once verified and approved, NISO Standards also become American National Standards.

The use of an ANSI/NISO Standard is voluntary. This is, the existence of this NISO Standard does not preclude anyone, whether or not that person has adopted the NISO Standard, from manufacturing, marketing, purchasing, or using products, processes, or procedures that so not conform to the NISO Standard. However, the use of the standards (those developed by NISO as well as other standards-developing organizations) has proven to be in the best interest of any industry wishing to increase its effectiveness and efficiency in the areas of product development, manufacturing, and marketing and, therefore, such use is encouraged by ANSI, NISO, and all other standards-developing organizations.

Table of Contents

1. Introduction

This standard, ANSI/NISO Z39.50-2001, Information Retrieval (Z39.50) Application Service Definition and Protocol Specification, defines an application protocol for search and retrieval of information in databases.

1.1 Scope and Field of Application

This standard describes 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.

This standard is intended for systems supporting information retrieval services for organizations such as information services, universities, libraries, and union catalogue centers. It addresses connection oriented, program - to - program communication. It does not specify a user interface.

1.2 Version

Z39.50-1995 specifies versions 2 and 3 for the Z39.50 service and protocol. This standard Z39.50-2001 also specifies versions 2 and 3, and additionally, incorporates many clarifications, amendments, defect corrections, and implementer agreements, all of which have been endorsed by the Z39.50 Implementers Group.

Z39.50-1992 specifies version 2 only. Version 2 of Z39.50 is assumed identical to version 1 of Z39.50; thus implementations that support version 2 automatically support version 1. Implementations that support version 3 are required to support version 2 (and thus version 1as well).

Certain procedures specified within the standard apply specifically to version 2 or version 3 and are noted as such.

1.3 References

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	Documentation - Format for Bibliographic Information Interchange on Magnetic Tape 1981.
ISO 4217	Codes for the representation of currencies and funds 1990.
ISO 8777	Information and Documentation - Commands for

	Interactive Text Searching.
ISO 8824:1990	Specification of Abstract Syntax Notation one (ASN.1).
ISO 8825:1990	Specification of basic Encoding Rules for Abstract Syntax Notation one (ASN.1/BER).
ISO 10160	Information and Documentation - Interlibrary Loan Application Service Definition for Open Systems Interconnection 1991.
ISO 10161	Information and Documentation - Interlibrary Loan Application Protocol Specification for Open Systems Interconnection 1991.

2. Definitions

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 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 description 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 alone or in combination with other access points) in a search for records. An access point may or may not correspond to one or more elements (defined by an abstract syntax),.
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 Protocol	The rules governing the format and exchange of information between a client and server.
Application Protocol Data Unit	A unit of information, transferred between client and server, whose format is specified by the Z39.50 protocol, consisting of application-protocol-information and possibly application-user-data.
Application ProtocolControl Information	Information conveyed by an application protocol data unit.
AppliedVariant	One of three usages for a variant specification. The applied variant is the variant specification that the server applied 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.
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 initiating application.
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 client or server) followed possibly by a response (from the peer). For example, Resource-control is a conditionally confirmed service, initiated by the server. See also Non-confirmed service and Confirmed service.
Confirmed service	A service that is defined in terms of a request (from the client or server) followed by a response (from the peer). For example, Search is a confirmed service, initiated by the client; Access-control is a confirmed service initiated by the server. 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 client and server 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).
Non-confirmed service	A service that is defined in terms of a request from the client or server, with no corresponding response. For example, Segment is a non-confirmed service initiated by the server. 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."
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.
Primitive Name	A name whose internal structure is not required to be understood or have significance to users of the name.
Record syntax	An abstract syntax requested by the client or used by the server to represent retrieval records. For a complete definition, see section 3.6.3.
Response record	A retrieval record or a 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 which 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	(Informal) 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	Reverse Polish Notation.
Schema	See Database schema.
Segment	A message that is sent (or is in preparation for transmission) by the server as part of an aggregate Present response, i.e. a Segment request or Present response.
Server	The application that includes the server; 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".
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 server 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.
Server	The entity that accepts a Z-association.
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.)
Variant	One of possibly several forms in which an element is available for retrieval. The client may request, or the server present, an element according to a specific variant. The server may indicate what variants are available for an element.
Variant list	A list provided by the server 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 client and either explicitly terminated by the client or server, or implicitly terminated by loss of connection. Communication between client and server is via a Z39.50-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. The specification is logically divided into procedures pertaining to the client and procedures pertaining to the server.

3.1 Model and Characteristics of the Information Retrieval Service

Communication between client and server is via a Z39.50-Association (Z-association). A Z-association is explicitly established by the client and may be explicitly terminated by either client or server, or implicitly terminated by loss of connection.

There may be multiple consecutive Z-associations for a connection. There may be multiple consecutive as well as concurrent operations (see 3.1.2) within a Z-association.

The roles of client and server 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 client and server. 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 client or server) followed by a response (from the peer). For example, Search is a confirmed service, initiated by the client; the Search service is defined in terms of a Search request from the client followed by a Search response from the server. Access-control is an example of a confirmed service initiated by the server.

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

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 client or server) followed possibly by a response (from the peer). For example, Resource-control is a conditionally-confirmed service, initiated by the server.

3.1.2 Z39.50 Operations

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

A request from the client 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 server. Only the client may initiate an operation, and not all client 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 client perspective, an operation begins when it issues the initiating request, and ends when it receives the terminating response. From the server 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 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. 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). A unique or non-unique key that can be specified (either alone or in combination with other access points) in a search for records. An access point may or may not correspond to one or more elements (defined by an abstract syntax),

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 "science fiction" find all records for which 'title' contains "galaxy" AND 'author' contains "adams". (" 'title' contains "galaxy" " is an access point clause, as is " 'author' contains "adams" ".

"AND" is a logical operator.)

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 "usage" 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 available at the server, for reference by the client, for subsequent searches or retrieval requests. When requesting the retrieval of a record from a result set, the client 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 client and server 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 result 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 server 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

Logically, a result set is an ordered list of items, each of which is a pointer to a database record; it is used as a selection mechanism for the transfer of database records identified by a query. A 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).

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 important to distinguish the physical implementation from the abstract model. How a server chooses to implement result sets is an implementation matter; a result set may be a copy of the database records, a table of pointers, or there may not even be a physical result set (the server might execute the query every time the result set is referenced, package up and send the requested records, and otherwise immediately discard the results; however, the result set model

does require that a result continue to refer to the same records in the same order). But from the client point of view (and the abstract model) the result set is a set of items, or vectors, where each includes a database name and a pointer to a record within the database.

3.1.6.1 Concurrency and Update Considerations

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.

The server could potentially modify a record pointed to by a result set. For example suppose the client requests record 1, and then subsequently requests record 1 again; the second time, the record may have changed. There is no direct mechanism provided by Z39.50 to prevent this (though Extended Services might be used to lock a record). When a client does modify a record referenced by a result set, then when the client subsequently requests the record, the server might refuse to supply it and instead supply a surrogate diagnostic, but there is no such requirement.

A successful search (see 3.2.2.1.10) results in the creation of a result set. However, a server may unilaterally delete a result set at any time for no specified reason; the next time that the client attempts to refer to that result set (for example in a Present request) the server might send a diagnostic to the effect "result set does not exist". (The server may instead send a diagnostic to the effect that "the result set no longer exists because it was unilaterally deleted", however, the server is not obliged to do so and may simply take the position that the result set never existed.) Therefore, although in the abstract a successful search results in the creation of a result set, as a practical matter, if a server does not actually create a result set, it is not violating the Z39.50 protocol.

3.1.6.2 Order of Result Set Records

The records in a result set are not necessarily ordered according to any specific or predictable scheme (although, whatever the order, it is assumed static). Thus assume for example there is a result set of 100 records, and the client wants the three most recent (i.e. based for example on 'date of publication'). There is no simple way to find those records, short of retrieving all the records in the result set. The Z39.50 Sort service however allows the client to request that a particular result set be sorted based on a specific key (e.g. ‘date of publication’, descending). If the server supports the sort service (and also supports sorting on the requested key, in this example, ‘date of publication’) then following the Sort, the client may subsequently retrieve the first three records, and they will be the three most recent.

3.1.6.2 Logical Structure of a Result Set 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 triples consisting of:

(a) An ordinal number corresponding to the position of the triple in the list

(b) A database name

(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 Services (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 client, 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 server in a special database, the "extended services database". The client 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 client 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 client 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 client may obtain details of the server implementation, including databases, attribute sets, diagnostic sets, record syntaxes, and element specifications supported. The client obtains these details through the Z39.50 Explain facility. The server maintains this information in a database that the client may access via the Z39.50 Search and Present facilities.

This "explain" database appears to the client as any other database supported by the server, but it has a well-known name and a pre-defined 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 client to initiate an Init operation.

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

Retrieval Facility — The Retrieval facility consists of two services:

- Present Service: A confirmed service initiated by the client to initiate a Present operation.
- Segment Service: A non-confirmed service initiated by the server, 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 client to initiate a Delete operation.

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

Sort Facility — The Sort facility consists of two services:

- Sort Service: A confirmed service initiated by the client to initiate a Sort operation.
- Duplicate Detection Service: A confirmed service initiated by the client to initiate a Duplicate Detection operation.

Access Control Facility — Access-control service: A confirmed service initiated by the server. 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:

- Resource-control Service: A conditionally-confirmed service initiated by the server. It does not initiate an operation, and it might or might not be part of an active operation.
- Trigger-resource-control Service: A non-confirmed service initiated by the client during an operation.
- Resource-report Service: A confirmed service initiated by the client 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 client to initiate an Extended-services operation.
Termination Facility	Close Service: A confirmed service initiated by the client or server. It does not initiate nor is it part of any operation. It allows a client or server to abruptly terminate all active operations and to initiate termination of the Z-Association. (Following termination of the Z-Association the client 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 enables the client to establish a Z-association. In the Init request, the client proposes values for initialization parameters. In the Init response, the server responds with values for the initialization parameters; those values, which may differ from the client-proposed values, are in effect for the Z-association.

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

Parameters of the Init Service

Parameter Name	Client Request	Server Response	Reference
Version	m	m	3.2.1.1.1
Id/authentication	o		3.2.1.1.2
Options	m	m	3.2.1.1.3
Preferred-message-size	m	m	3.2.1.1.4
Exceptional-record-size	m	m	3.2.1.1.4
Result		m	3.2.1.1.5
Implementation-id	o	o	3.2.1.1.6
Implementation-name	o	o	3.2.1.1.6
Implementation-version	o	o	3.2.1.1.6
User-information-field	o	o	3.2.1.1.7
Other-information	o	o	3.2.1.1.8
Reference-id	o	ia	3.4

Key:

- **m** Mandatory

- **o** Optional
- **ia** If applicable

3.2.1.1.1 Version

Both the client and server 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 server 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. (**Note:** version 1 is not defined by this standard, however there may be implementations that indicate support for version 1, based on other, obsolete standards.)

3.2.1.1.2 Id/Authentication

The client and server agree, outside the scope of the standard, whether or not this parameter is to be supplied by the client, and if so, to the value. This value is used by the server to determine if the client is authorized to enter into communication with the server.

3.2.1.1.3 Options

For each of the capabilities (represented by an “option bit”) in the table below, the client proposes either 'on' or 'off' (meaning 'in effect' or 'not in effect' respectively) and the server responds correspondingly for each. The response determines whether the capability is in effect.

Option Bit	Option	Description
0	Search	See Note 1
1	Present	See Note 1
2	Delete Result Set	See Note 1
3	Resource Report	See Note 1
4	Trigger Resource Control	See Note 2
5	Resource Control	See Note 3
6	Access Control	See Note 3
7	Scan	See Note 1
8	Sort	See Note 1
9	Unused	
10	Extended Services	See Note 1
11	Level 1 Segmentation	See Note 4
12	Level 2 Segmentation	See Note 4
13	Concurrent Operations	See Note 6
14	Named Result Sets	See Note 5
15	Encapsulation	See 4.2.4
16	resultCount parameter in Sort Response	See Note 8

Option Bit	Option	Description
17	Negotiation Model	See Note 9
18	Duplicate Detection	See Note 1
19	Query type 104	See Note 10
20	PeriodicQuery ES Correction	See Note 11
21	Use of string values for schema in compSpec	See Note 12

Notes:

1. For each Z39.50 operation type -- search, present, delete, resource-report, scan, sort, extended-services, duplicate detection -- the client indicates that it may choose to initiate operations of that type by setting the value for that type to ‘in effect’; if so, the server indicates whether it is willing to process an operation of that type. If the client proposes 'not in effect' for a particular operation type, the server must also specify 'not in effect'.
2. The client may propose to submit Trigger-resource-control requests; if so, the server indicates whether it will accept Trigger-resource-control requests. If the client proposes 'not in effect,' the server must also specify 'not in effect'. If the server specifies 'in effect' for Trigger-resource-control, but 'not in effect' for 'resource-control,' then the client may use only the Cancel function of Trigger-resource-control. The server may indicate unwillingness to accept Trigger-resource-control requests even if it specifies 'in effect' for 'resource-control’. The server's indication of willingness to accept Trigger-resource-control requests does not imply that the server will take any action as a result of a Trigger-resource-control request.
3. The client indicates whether it proposes to permit the server to invoke Resource-control and/or Access-control (i.e. send Resource-control and/or Access-control requests). The server specifies that it may choose to (or will not) invoke Resource-control and/or Access-control. If the server specifies 'not in effect' for resource-control (or access-control) then it will not invoke resource-control (or access control) even if the client has proposed 'in effect’. If the client proposes 'not in effect' for resource-control, and the server indicates 'in effect' for resource-control, indicating that it is not willing to suppress Resource-control requests, and if indeed the client cannot accept Resource-control requests, the client should terminate the Z-association. If the client proposes 'not in effect' for access-control, and if security requirements on the server 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 server 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 client proposes 'not in effect' for access-control, and the server normally invokes security (other than at the Z-association level), the server need not necessarily reject the Z-association. The server might wish to invoke a security challenge during an Init operation to determine whether the client is authorized to use a capability it has proposed. If the client has proposed 'not in effect' for access-control, the server may simply refuse the use of that particular operation via the Options parameter. If the client proposes 'not in effect' for access-control, and the server chooses to accept the Z-association, and if the client subsequently initiates an action that would precipitate an Access-control request (for example, if the client issues a Search specifying a database for which it has not yet established credentials), the server should suppress the Access-control request and instead respond with an error status indicating that a security

challenge was required but could not be issued.

4. The client proposes one of the following:
   1. "no segmentation", by specifying 'not in effect' for both level 1 and level 2 segmentation;
   2. "level 1 segmentation", by specifying 'in effect' for level 1 and 'not in effect' for level 2 segmentation; or
   3. "level 2 segmentation", by specifying 'in effect' for level 2 segmentation.

   If the client proposes 'in effect' for level 2 segmentation then it may also propose 'in effect' for level 1 segmentation to indicate that if the server is unable to support level 2 segmentation, the client wishes level 1 segmentation to be in effect." Segmentation" is said to be 'in effect' if either level 1 or level 2 segmentation is in effect. Segmentation may be in effect only when version 3 is in force. The server response indicates which, if either, form of segmentation it intends to perform. If the server specifies neither level 1 nor level 2 then 'no segmentation' is in effect, regardless or what the client has proposed. If the server specifies level 1 (but not level 2) segmentation, it will not perform level 2 segmentation, and the client must be prepared to accept level 1 segmentation, regardless of what the client has proposed. If the server specifies level 2 segmentation, the client must be prepared to accept level 2 segmentation regardless of what it has proposed (the server value for level 1 should be 'not in effect'). When 'no segmentation' is in effect, the server 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 server 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 server may respond to a Present request with multiple segments, and individual records may span segments. Segmentation procedures are detailed in section 3.3.

5. The client may propose to be permitted 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 server specifies whether it will support named-result-sets. If the client proposes 'not in effect,' the server must also specify 'not in effect'.
6. The client may propose to be permitted to initiate concurrent operations; if so, the server indicates whether it will accept concurrent operations. If the client proposes 'not in effect,' the server must also specify 'not in effect'.
7. Note 7 Deleted.
8. Rules for negotiation of resultCount parameter in Sort Response are as follows:
   1. If the client sets bit 8 (proposing that it wishes to initiate Sort operations) then it may also sets bit 16 indicating that it supports the resultCount parameter in the response.
   2. The client must not set bit 16 if it does not set bit 8.
   3. If the client does not set bit 16, then the server must not set bit 16 under any circumstance.
   4. If the client sets bit 8 and not bit 16, and if the server sets bit 8, accepting the proposal that Sort operations may be submitted (the server must not set bit 16 as noted above), the server must not include the resultCount parameter in a Sort response.

5. If the client sets bits 8 and 16, and the server also sets bits 8 and 16, the server may include the resultCount parameter in Sort responses. However, the server, by setting bit 16, does not obligate itself to supplying the resultCount parameter.
6. If the client sets bits 8 and 16, and the server sets bit 8 but not bit 16, the server indicates that it will accept Sort requests, but will not include the resultCount parameter in a Sort response.

9. When the client sets the "negotiation model" option bit, it signifies adherence to the negotiation model. If the client and server both set the option bit (in the InitRequest and Response respectively) both may assume that negotiation is carried out in accordance with the model.

   If the client sets this option bit and the server does not, the client should assume that negotiation has not been carried out in accordance with the model.

   If the client does not set this option bit, but the server requires that negotiation be carried out in accordance with this model, the server may reject the Z-association and supply diagnostic 1055: "negotiation option required".

   An option bit corresponding to the negotiation model is necessary for this reason. Suppose a server is unaware of the negotiation model and suppose further that that server routinely echos in the InitResponse all of the information supplied in the InitRequest. This behavior may lead the client to believe (falsely) that negotiation has been carried out. However, because the server is unaware of the negotiation model, it will not set that bit, and so the client will know that the negotiation model is not in effect.

10. When the client sets the "query type 104" bit it proposes to submit queries of type 104. If the server also sets this bit, then it will accept queries of type 104 -- this means the server will not consider it to be a protocol error if the client submits a type 104 query; it does not mean that the server agrees to support any specific external query-definition.
11. If this option is negotiated, then in the event that the PeriodicQuery Extended Service is used, the definition in this standard will be in effect; otherwise the definition in Z39.50-1995 will be in effect:
    - databaseNames must not occur in ClientPartToKeep if option bit 20 is set,
    - databaseNames must not occur in ClientPartNotToKeep unless option bit 20 is set,
    - additionalSearchInfo must not occur in ClientPartNotToKeep unless option bit 20 is set,
    - databaseNames must occur in ServerPart if bit 20 is set and must not occur if option bit 20 is not set,
    - lastQueryTime and lastResultNumber are optional if bit 20 is set and mandatory otherwise,
    - additionalSearchInfo must not occur in ServerPart unless bit 20 is set.
12. If this option is negotiated, then Schema within Specification (in the ASN.1 APDU definitions) is defined as in this version of the standard, that is it may take on the additional choice of a string. Otherwise it must be an object identifier.

3.2.1.1.4 Preferred-message-size and Exceptional-record-size

The Init request contains the client'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 server is going to use; these may be different from (and

override) the values proposed by the client. 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 client 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).

If the client supplies values of zero for both parameters, then the client is explicitly indicating "no preference". If the server responds with zero for both parameters (regardless of what the client proposed), the server is indicating that the client must be prepared to accept arbitrarily large records and arbitrarily large messages.

Note that "message size", as in Preferred-message-size, etc., means the sum of the sizes (in bytes) of the response records (retrieval and diagnostic records) in the Present or Search Response APDU. As a practical matter this standard does not prescribe that message sizes be strictly enforced; thus if the negotiated Preferred-message-size is 32767, the server does not need to ensure that the message size does not exceed that value, but rather that it not exceed it substantially. Moreover, message size is the sum of the sizes of the records not including protocol control information. However, this standard does not attempt to distinguish what is protocol control information versus record content. Thus, if the server thinks it sent 32,767 bytes, but the client thinks it sent 32,770 bytes (because the server considered something to be data that the client though was protocol control information) the server complies in spirit, and the client should be forgiving and not consider this to be a protocol error. It is recommended that the client propose a value of Preferred-message-size that is less than the actual limit that the client can support.

The use of these parameters is detailed in 3.3.

3.2.1.1.5 Result

The server 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 client 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 client or server implementation. These three implementation parameters are provided solely for the convenience of implementers, for the purpose of distinguishing implementations.

3.2.1.1.7 User-information-field

This parameter may be used by the client or server for additional information not specified by this standard.

3.2.1.1.8 Other-information

This parameter may be used by the client or server for additional information not specified by the standard. This parameter may be used only if version 3 is in force.

Note*:* The use of Other-information during initialization (i.e. within the Init request or response, but particularly in the request) is not recommended, because of the uncertainty of what protocol version, 2 or 3, is in force during initialization See Appendix USR (USR.2 Use of Init Parameters for User Information).

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 a client to specify a query to be applied to databases at a server system, and to receive information about the results of the query.

The search request allows the client to request that the server apply a query to a specified set of databases at the server, to identify records with the properties indicated by the query. The server creates a result set, which represents the set of records identified by the query. 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). The server 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. (This is referred to informally as "piggybacking". Retrieval is in general a function of the Present service but in special cases may be carried out as part of the Search operation.)

Parameters of the Search Service

Parameter Name	Client Request	Server Response	Reference
Query-type	m		3.2.2.1.1
Query	m		3.2.2.1.1
Database-names	m		3.2.2.1.2
Result-set-name	m		3.2.2.1.3
Replace-indicator	m		3.2.2.1.3
Small-set-element-set-names	o		3.2.2.1.4
Medium-set-element-set-names	o		3.2.2.1.4
Preferred-record-syntax	o		3.2.2.1.5
Small-set-upper-bound	m		3.2.2.1.6

Parameter Name	Client Request	Server Response	Reference
Large-set-lower-bound	m		3.2.2.1.6
Medium-set-present-number	m		3.2.2.1.6
Response-records		ia	3.2.2.1.7
Result-count		m	3.2.2.1.8
Number-of-records-returned		m	3.2.2.1.9
Next-result-set-position		m	3.2.2.1.10
Search-status		m	3.2.2.1.11
Result-set-status		ia	3.2.2.1.11
Present-status		ia	3.2.2.1.11
Additional-search-information	o	o	3.2.2.1.12
Other-information	o	o	3.2.2.1.13
Reference-id	o	ia	3.4

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 client and server have a priori agreement outside of the standard.
- Type-1 is the Reverse Polish Notation (RPN) query (so-called because RPN is sometimes used as an abstract representation; note however that RPN is not used as an encoding of the Type-1 query). It is 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.
- Type-104 is used to identify externally defined queries.

3.2.2.1.2 Database-names

The client indicates the set of databases to which the Query applies.

Notes:

1. **Database Combinations**. The server 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 server 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. **Multi-database Searching.** Z39.50 allows but does not require support for,

multi-database searching in a single Search request. A client may simply choose not to send Search requests that list more than a single database. A server, upon receipt of a multi-database search request may simply fail the search (diagnostic 111: "Too many databases specified"; with an addinfo "maximum value" 1 supplied). A server that chooses to support multi-database search request, may limit such support to any specific combinations it chooses. For example suppose a server provides 100 databases, named "1", "2", ... , "100". It may declare that each database may be searched singly, and databases "1" and "2" may be searched together. (It may declare this via Explain, or, more dynamically, when it receives a search with an unsupported combination of databases, it may fail the search with diagnostic 23: "Specified combination of databases not supported".) A server may define virtual databases corresponding to supported combinations rather than support multi-database search requests. For example, suppose there are 3 real databases (A, B, and C) to which a server will allow access in all possible combinations; the server might expose 7 virtual databases (A, B, C, AB, AC, BC, and ABC), thus in effect providing all possible combinations but allowing only a single (virtual) database to be specified in a Search request.

3. **No Default Database.** There is no defined concept in Z39.50 of a “default” database. If a server considers a particular database to be the default database, it could express that fact via Explain (using the “description” field of DatabaseInfo) however that information would be for human consumption only.
4. **Case Insensitive.** Each database name specified by the server is a string of characters, and the string is case-insensitive. Thus for any character that is a letter, the client may use either upper or lower case, regardless of how the server 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 server, 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 server, 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 server, 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 server need not support, in general, the naming of result sets by the client (see Notes below.). However, a server must support at least the result set whose name is "default." If the client 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
- It is unilaterally erased or deleted by the server

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 server (at any time)
- By termination of the Z-association

Notes:

1. A server need not support, in general, the naming of result sets by the client. The server may always choose to:

   (a) Fail the search, supplying diagnostic 22 (Result set naming not supported).

   Alternatively, in certain circumstances (detailed below) the server may (but need not):

   (b) Treat the condition as a protocol error: issues a Close, with Close-reason ‘protocol error’, if version 3 is in effect, or terminate the connection if version 2 is in effect.

   Result set naming is a negotiated feature in thus a server who doesn't support it should so inform the client during initialization (via option bit 14). If so, then a subsequent attempt by the client to name a result set (i.e. assign a name other than 'default') may be construed by the server to constitute a protocol error.

   However the mechanism, by which the server informs the client that result set naming is not suported, is not reliable unless either version 3 is in effect or if the client sets bit 14 during initialization (implicitly acknowledging that it knows the meaning of this option bit). Z39.50-1992 did not define this option bit, so by 1992 interpretation, the condition is not a protocol error. If version 2 is in force, the client may have implemented Z39.50-1992. If version 3 is in force, the server may be sure that the client has implemented Z39.50-1995 or later.

   In general, the server may always choose behavior (a) above, and more specifically: If version 2 is in force, and if the client did not attempt during initialization to negotiate result set naming (via option bit 14), the server should choose behavior (a). If version 2 is in force, but the client did attempt during initialization to negotiate result set naming, and the server rejected that option, the server may choose (a) but may alternatively choose (b).

   If version 3 is in force, and the server had set bit 14 off indicating that result set naming is not supported (regardless of whether the client set bit 14), then similarly, the server may choose (a) but may alternatively choose (b).

2. Result set names are *case sensitive*. This is in contrast to database names, which are case-*in*sensitive (see 3.2.2.1.2 Database-names note 4). Database names are passed around outside of the protocol, often by humans, and transcend Z-associations. Result set names have none of these characteristics; they are referred to only by the client (the mapping between user specified names and actual result set names used in the protocol are not within the scope of the Z39.50 protocol), within a Z-association, and the client is expected to use a given result set name consistently within a Z-association. (Element set names are case-*in*sensitive; see 3.6.2.)

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 (see 3.2.2.1.6). If the query results in a small-set then Small-set-element-set-names pertains. If the query results in a medium-set, then Medium-set-element-set-names pertains.

3.2.2.1.5 Preferred-record-syntax

The client may specify a preferred record syntax for retrieval records.

See 3.2.3.1.5.

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 client (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 less 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 client 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 server 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 server attempts to create retrieval records to be included in the Search response, corresponding 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
- 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 server returns a single non-surrogate diagnostic record. If version 3 is in force, the server 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. The database name appended to a record need not be one of the database names that was included in the original search request (the request that caused the creation of the result set from which the record is being presented).

When Server Does Not Support Response Records in a Search Response

If a server does not support “piggybacking”, i.e. supplying response records in a Search response, and when the Search request parameter values for small-set-upper-bound and large-set-lower-bound and the Search response parameter value for Result-count are such that the response is expected to include one or more records as specified in 3.2.2.1.7 (for example, if a search results in 5 records and small-set-upper-bound on the search request was 10, the server is expected to attempt to return all five records), the server should respond with a Search-status of 'success' and a Present-status of 'failure', and include a non-surrogate-diagnostic, for example General Diagnostic “1005: Response records in Search response not supported”, or “1006: Response records in Search response not possible for specified database or database combination”.

Note that Diagnostic 1006 is intended for the case of an intermediary providing access to multiple servers, some of which may support piggybacking and some which do not, and is for the intermediary to use in case the particular end server does not support piggybacking, to distinguish from the case where the intermediary itself does not support piggybacking.

When Query is not supported for a Database

When the query is not supported for one of the databases (that is, when there is at least one database listed by the Client in the Database-names parameter, 3.2.2.1.2, for which the query is not supported, even though the query may be supported for other listed databases), the server

should set 'search-status' to ‘failure’, and supply an appropriate diagnostic. (For example, when an attribute is not supported for a database, supply diagnostic 1056: "Attribute not supported for database". Multiple instances of diagnostic 1056 may be supplied for multiple attribute/database combinations.)

When a query is supported for one or more but not all of the databases, and if the server wishes to make the result set available to the client (the result set produced by applying the query to the subset of supported databases) the server may set Search-status to ‘failure’ and Result-set-status to ‘subset: partial, valid results available’. The reasoning for this prescribed behavior is as follows. Setting Search-status to 'success' would give the client the false impression that the search was executed across all of the databases. For those databases for which the query is not supported, the client would be led to believe instead that the query is indeed supported and that no records were found. But while a status of 'success' is misleading when part of the search fails, 'failure', too, is misleading when part of the search succeeds. Thus the parameter Result-set-status helps disambiguate a 'failure' status. Furthermore, Result-set-status may be supplied only when Search-status is 'failure'. Thus, setting status to 'failure' is prescribed, not because of any implied semantics of a failure status but (more pragmatically) because it provides the client with access to the result set.

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.

A negative value for resultCount in a Search Response is invalid and constitutes a protocol violation. A client that receives a negative value of Result-count may issue a Close Request with Close-reason 'protocol error'.

The parameter Number-of-records-returned is the total number of records returned in the Search response, including diagnostic records, surrogate or non-surrogate.

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 server 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 server returns a single non-surrogate diagnostic record. If version 3 is in force, the server 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. See note 1.
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 client request. See note 2.
Partial-4	Not all of the expected response records can be returned because the request was terminated by resource control, by the server. See Note 3.
Failure	None of the expected response records can be returned. One or more non-surrogate diagnostic records are returned (see note in 3.2.2.1.7).

Notes:

1. In the case where the Search Request specified Small-set-upper-bound = 0 and Large-set-lower-bound = 1 (i.e. don't present records under any circumstances) and where Search-status = 'success'; Present-status (which must be supplied, since Present-status must occur if Search-status is 'success') should be 'success', where in this case “success” may be interpreted to mean "you asked for no records, and no records were sent, so therefore consider the disposition of the Present part of the request to be successful".
2. If partial-3 is indicated, this implies that either (1) the server sent a resource control request to which the client responded "do not continue", or (2) the client issued a TriggerResourceControl request to terminate the operation. This means that either Resource Control or Trigger Resource control (for cases 1 and 2 respectively) must have been negotiated.
3. Partial-4 may be indicated when the server simply terminates the operation, unilaterally, perhaps because resources at the server are limited (but not as a result of a Resource-control response indicating "do not continue".) Partial-4 does not require that

resource control be negotiated.

Notes Pertaining to the Relationship among Search-Status, Present-Status, and Result-Set-Status

Informally, in a Z39.50 Search operation, the client supplies search criteria and requests the server to:

(a) Identify records that meet the criteria

(b) Report the number of records identified

(c) Establish a result set corresponding to those records (see note 1)

(d) (Conditionally) return some or all of the records (see note 3)

Notes:

1. If no records were identified (see note 2), the server might not physically create a result set, but in the abstract, an empty result set is assumed to have been created.
2. The qualification "no records were identified" means that the server performed the search and is stating that "there are no records meeting the criteria"; it does not mean the server was unable to determine the number (in which case the server should fail the search).
3. The number of records returned, which may be zero, depends on the result count and the values of the Search Request parameters Small-set-upper-bound, Large-set-lower-bound, and Medium-set-present-number.

The search operation consists of a search phase and a retrieval phase, where (a), (b), and (c) correspond to the search phase and (d) to the retrieval phase. The search is considered to succeed if and only if the server is able to perform (a), (b), and (c).

Thus if a search results in the identification of zero records (that is, the server determines that no records meet the criteria), this does not constitute failure. (In Z39.50, "search" does not so much connote "locate", as in the classical usage of the term "search", as it means "identify how many and which items meet specified criteria". "None" is a valid answer to "how many", and in that case, the "which" part does not apply.) The search-status values of 'success' and 'failure' correspond respectively to whether the search succeeds or not.

When the search succeeds, a result set is created. If the search fails, a result set may or may not be created, and its existence may be determined by the value of Result-set-status (see table below). Result-set-status occurs if and only if the search fails; if the search succeeds, the existence of the result set is implicitly known (it exists) and therefore the parameter is not necessary.

There is a retrieval phase if and only if the search succeeds, and if so, the response parameter Present-status occurs.

Therefore exactly one of the parameters Result-set-status and Present-status occurs in the response. Result-set-status corresponds to a search that fails, and Present-status to a search that succeeds.

Present-status has values of 'success' or 'failure', similar to Search-status (though, in contrast, Present-status has additional intermediate statuses as well). 'Success' means that the server presented all of the response records (retrieval or surrogate diagnostic records) that it attempted to present; 'failure' means that it was unable to present any of the records (cases where the server presents some, but not all, are covered by the intermediate statuses).

Whenever either Search-status or Present-status is 'failure', the server must provide one or more non-surrogate diagnostics, supplying diagnostic information associated with the failure.

The following table summarizes the relationship among the statuses, the presence of non-surrogate diagnostics, and the existence of a result set.

If Search Status is:	And Present Status is:	Then Result Set Status:	And a Non-surrogate Diagnostic:	And a Result Set:
'success'	'success' (or 'partial')	Must not occur	Must not be supplied	Exists
'success'	'failure'	Must not occur	Must be supplied (at least one)	Exists
'failure'	*(Present Status must not occur)*	Must occur	Must be supplied (at least one)	Exists if result-set-status is 'subset' or 'interim'; Does not exist if 'none'.

3.2.2.1.12 Additional-search-information

On the response the server may use this parameter to convey information, which 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 client may use this parameter to indicate the preferred format or content of that information. User Information format SearchResponse-1 is defined in Appendix 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 client or server 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 client sends a Present request-to-request response records according to position within a result set maintained by the server. The server 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 server 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 client 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 client specifies a range and may follow with subsequent requests specifying different ranges.

Note: If version 3 is in force, a single request may include more than one range.

The client may request, for example, records one through five and follow with a request for records four through six.

Note: In this section, "record N" means "the response record corresponding to the database record identified by result set entry N."

Parameters of the Present Service

Parameter Name	Client Request	Server Response	Reference
Number-of-records- requested	m		3.2.3.1.1
Result-set-start-position	m		3.2.3.1.1
Additional-ranges	o		3.2.3.1.2
Result-set-id	m		3.2.3.1.3
Element-set-names	o		3.2.3.1.4
Preferred-record-syntax	o		3.2.3.1.5
Comp-spec	o		3.2.3.1.6
Max-segment-count	o		3.2.3.1.7
Max-segment-size	o		3.2.3.1.7
Max-record-size	o		3.2.3.1.7
Response-records		ia	3.2.3.1.8
Number-of-records- returned		m	3.2.3.1.9
Next-result-set-position		m	3.2.3.1.9
Present-status		m	3.2.3.1.10
Other-information	o	o	3.2.3.1.11
Reference-id	o	ia	3.4

3.2.3.1.1 Number-of-records-requested and Result-set-start-position

The client requests a range of records: N records beginning at record M. M = Result-set-start-position, N = Number-of-records-requested.

Note: The 1995 version of this standard assumed that the client always know the size of (i.e. number of records in) the result set, and therefore considered it to be a protocol error if N is greater than (Result-count - M) + 1; that is, if the range requested includes records whose ordinal result set position would exceed the number of items in the result set (for example, requesting result set records 7 through 10 when there are only 8 records in the result set). However, if version 3 is in effect, this need not be considered a protocol error, because the client might not

always be expected to know the size of the result set (for example when the result set is sorted, or de-duplication is performed, or the result set is a re-activated persistent result set). The server may chose to treat it as a protocol error (and may issue a non-surrogate diagnostic, such as diagnostic 13) or may honor the request and supply a surrogate diagnostics for each of the non-existent records (and similarly, diagnostic 13 may be used as a surrogate diagnostic). Note however, if version 2 is in effect, then this condition must be treated as a protocol error.

3.2.3.1.2 Additional-ranges

The client 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 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.

If the client includes this parameter and the server does not support Additional-ranges, it should fail the Present (Present-status of 'failure' with a non-surrogate diagnostic 243).

3.2.3.1.3 Result-set-id

The client 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 client may indicate the desired composition of the retrieved records. See 3.6.2.

3.2.3.1.5 Preferred-record-syntax

The client may specify a preferred record syntax for retrieval records.

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 server 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 client has received the aggregate Present response, the result (if all of the segments are re-assembled, and segmented response records re-assembled from their fragments) will be one of the following:

- N response records, where N = Number-of-records-requested
- A number of response records, which is less than N (reason specified by Present-status)
- 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 server returns a single non-surrogate diagnostic record. If version 3 is in force, the server 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 client or server 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 server 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 modeled as a request, even though, logically, the server 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 modeling constraints: the Segment service cannot be modeled as a response (because if it were, it would necessarily respond to a segment request, and it is a non-confirmed service); and the present operation cannot be modeled 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 server 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 server does not send any Segment requests and the aggregate Present response consists of a simple Present response.

Parameters of the Segment Service

Parameter Name	Server Request	Reference
Segment-records	m	3.2.3.2.1
Number-of-records-returned	m	3.2.3.2.2
Other-information	o	3.2.3.2.3
Reference-id	ia	3.4

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 server 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 a client to request that the server delete specified result sets, or all result sets, created during the Z-association. The server responds by reporting information pertaining to the result of the operation.

Although a result set is deleted automatically after the Z-association is closed. A client may wish to delete a result set explicitly during the Z-association. A server might have a limit to the number of result sets it can maintain, and might unilaterally delete one or more result sets when that limit is approached. Therefore by deleting a result set it no longer needs, a client might forestall the possibility that a result set that it does still need may be unilaterally deleted. Moreover, some servers charge to maintain results, so deleting a result set when it is no longer needed may help reduce cost.

Parameters of the Delete Service

Parameter Name	Client Request	Server Response	Reference
Delete-function	m		3.2.4.1.1
Result-set-list	ia		3.2.4.1.2
Delete-operation-status		m	3.2.4.1.3
Delete-list-status		ia	3.2.4.1.4
Number-not-deleted		ia	3.2.4.1.5

Parameter Name	Client Request	Server Response	Reference
Bulk-statuses		ia	3.2.4.1.5
Delete-msg		ia	3.2.4.1.6
Other-information	O	o	3.2.4.1.7
Reference-id	O	ia	3.4

3.2.4.1.1 Delete-function

The client 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 server 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 the table below.

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'.

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

failure-10	Result-set in use. See Note.2.

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.

3.2.4.1.5 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 server is not obligated to provide statuses for each result set not deleted on a bulk delete. For example, a server 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 server 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 client or server 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 server to challenge a client. 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.

A client must be prepared to accept and respond to Access-control requests from the server if access control is in effect. A server 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 server 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 client 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 server 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 client response is acceptable to the server, the operation proceeds as if the challenge has never taken place. If the client fails to respond correctly to the challenge then the server's terminating response to the interrupted operation may indicate that the operation was terminated due to an Access-control failure.
2. If the client fails to respond correctly to a challenge during an Init operation, the server 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 server need not necessarily reject the Z-association. For example the server might wish to invoke a security challenge during an Init operation to determine whether the client is authorized to use a capability it has proposed. If the client fails to respond properly, the server may simply refuse the use of that particular operation (via the Options parameter).
3. During a Search or Present operation, while the server is preparing records for presentation, it might send an Access-control request pertaining to a particular record. If the client fails to respond correctly to the challenge, the server 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 client must be prepared at any time during the Z-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 server 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 client response is acceptable to the server, the suspended operations proceed as if the challenge had never taken place.
3. If the client fails to respond correctly to the challenge, the server might decide to terminate one or more operations but to leave open the Z-association. In that case, the server's terminating response to any such operations may indicate that the operation was terminated because of an Access control failure. Alternatively, the server may close the Z-association.

Parameters of the Access Control Service

Parameter Name	Server Request	Client Response	Reference

Security-challenge	m		3.2.5.1.1
Security-challenge-response		m	3.2.5.1.1
Other-information	o	o	3.2.5.1.2
Reference-id	ia	ia	3.4

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 in Appendix ACC. Alternatively, the contents of these two parameters may be established by prior agreement between a given server/client pair.

3.2.5.1.2 Other-information

This parameter may be used by either the client or server 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:

- The Resource-control service, invoked by the server, either as part of an active operation (of any type) or pertaining to the Z-association
- The Trigger-resource-control service, invoked by the client as part of an active operation (of any type except Init)
- The Resource-report service, invoked by the client to initiate a Resource-report operation

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

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

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

3.2.6.1 Resource-control Service

A client must be prepared to accept and respond to Resource-control requests from the server if resource control is in effect. A server 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 server 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 client 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 server awaits the Resource-control response, and subsequently issues a terminating response after processing of the operation is concluded.
- If not, the client must not issue a Resource-control response. If the Resource-control request was part of an operation: the server subsequently issues the terminating response, after processing of the operation is concluded.

A client 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 client 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 client request. However, the response might alternatively indicate that the operation completed, since the operation at the server may continue to execute and subsequently complete before the Resource-control response reaches the server.

Parameters of the Resource Control Service

Parameter Name	Server Request	Client Response	Reference
Resource-report	o		3.2.6.1.1
Partial-results-available	ia		3.2.6.1.2
Suspended-flag	ia		3.2.6.1.3
Response-required	m		3.2.6.1.4
Triggered-request-flag	o		3.2.6.1.5
Continue-flag		m	3.2.6.1.6
Result-set-wanted		ia	3.2.6.1.7
Other-information	o	o	3.2.5.1.8

Reference-id	ia	ia	3.4

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 Appendix RSC.

3.2.6.1.2 Partial-results-available

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

Value of Partial-results-available	Description
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 server will accept subsequent Present requests against the result set if the client 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 server need not accept subsequent Present requests in the event that the client 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 treated as the authoritative information.

Access to Incomplete Results

This parameter Partial-results-available may be used not only for post-search information, but during a search as well. The server may notify the client of the availability of partial (i.e. incomplete) results by sending a resource report via that may include the name of a result set (different from the result set specified in the Search Request) that contains the partial results, and the client may begin retrieving records, if concurrent operations is in effect. The client may send a trigger-resource-control request during the search, specifying the searchResult-1 format (see Appendix USR, USR.1), and so indicate that it wants partial results.

3.2.6.1.3 Suspended-flag

This parameter is valid only when the request pertains to an operation. The server 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 server indicates whether or not a response (from the client) to this request is required.

3.6.2.1.5 Triggered-request-flag

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

3.2.6.1.6 Continue-flag

This parameter is valid only when the request pertains to an operation. The client indicates to the server 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 server will maintain the (possibly partial) result set for subsequent Present operations. If the value of the flag is 'no,' the server may delete the result set. A result set status of 'none' on the subsequent Search response indicates that the server 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 server and the way in which the search terminated.

3.2.6.1.8 Other-information

This parameter may be used by either the client or server 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

A client 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 server that the client wishes the server to:

a) Simply send a Resource-report, i.e. issue a Resource-control request with Response-required 'off';

b) Invoke full resource control, i.e. issue a Resource-control request with Response-required 'on'; or

c) Cancel the operation.

The server 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 server wishes to issue a Resource-control request it does so unilaterally. (If the client issues a Trigger-resource-control request and subsequently receives a Resource-control request as part of the same operation, the client cannot necessarily determine whether the latter resulted from

the Trigger-resource-control request. However, the server may include Triggered-request-flag in the Resource-control-request to so indicate.)

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

Although a client may issue a Trigger-resource-control request as part of an active operation, the server might receive the request after the operation terminates. In that case, the server will ignore the Trigger-resource-control request. Furthermore, the server 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 server should ignore the Trigger-resource-control request. (Note that in general, the server may ignore any Trigger-resource-control request.)

Parameters of the Trigger-resource-control Service

Parameter Name	Client Request	Reference
Requested-action	m	3.2.6.2.1
Preferred-resource-report-format	ia	3.2.6.2.2
Result-set-wanted	ia	3.2.6.2.3
Other-information	o	3.2.3.2.4
Reference-id	ia	3.4

3.2.6.2.1 Requested-action

The client 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 client 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 client requests that the server 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 client 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 a client 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 server is obliged to respond, although the server 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).

Parameters of the Resource Report Service

Parameter Name	Client Request	Server Response	Reference
Preferred-Resource-report-format	o		3.2.6.3.1
Op-id	o		3.2.6.3.2
Resource-report-status		m	3.2.6.3.3
Resource-report		o	3.2.6.3.4
Other-information	o	o	3.2.5.3.5
Reference-id	o	ia	3.4

3.2.6.3.1 Preferred-resource-report-format

The client may indicate a resource report format that it prefers.

3.2.6.3.2 Op-id

This parameter may be supplied by the client to identify a completed operation for which the client 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 client anticipates that it will subsequently issue a Resource-report request pertaining to that operation, it is the client's responsibility to ensure that the Reference-id is not re-used before doing so.
2. The client 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 client 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 client does so, the server may (but need not) consider the request in error (see failure-6 of Resource-report-status).

3. If the client 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 server receives the Trigger-resource-control request, the client 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 client requests a resource report pertaining to the Z-association.

3.2.6.3.3 Resource-report-status

The server supplies one of following status values:

Status	Meaning
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	Server unable to supply resource report
failure-2	Operation terminated by server 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 client or server 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 two services, Sort and Duplicate-detection.

3.2.7.1 Sort Service

The Sort service allows a client to request that the server sort a result set (or merge multiple result sets and then sort). The client 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 server to apply to the result set as so ordered.

Parameters of the Sort Service

Parameter Name	Client Request	Server Response	Reference
Input-result-sets	m		3.2.7.1.1
Sorted-result-set	m		3.2.7.1.2
Sort-sequence	m		3.2.7.1.3
Sort-status		m	3.2.7.1.4
Result-set-status		ia	3.2.7.1.5
Diagnostics		ia	3.2.7.1.6
Result-count		o	3.2.7.1.7
Other-information	o	o	3.2.7.1.8
Reference-id	o	ia	3.4

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 server, 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 server action if an element is missing from a record in the result set to be sorted.

Each sort element includes a sort key (that the server has designated either via the Explain facility, or through some mechanism outside of the standard) that takes one of three possible forms:

(1) field-in-record

(2) abstract access point

(3) private sort key

(1) and (2) correspond respectively to a "retrieval" and "search" view of the record.

The "field-in-record", corresponds to a retrieval element of the record, as defined perhaps by a schema, and specified by an element specification, such as eSpec-2, or by an element set name. The element specification or name should resolve to a single element; if it resolves to several elements, the sort key is not well defined.

- "abstract access point" corresponds to a search attribute, or several; however if several are supplied, then similarly, they should resolve to a single abstract access point.
- "private sort key" is used when the client and server have a prior agreement about what the supplied key means. For example the client may supply the string 'author', where that string doesn't denote any particular field in the record as defined by the schema, and it isn't an attribute (no attribute set id) but the server knows (because of prior agreement) that that string, when supplied as a private sort key, and when applied to the records in question, refers to a specific field.

3.2.7.1.4 Sort-status

The parameter Sort-status, returned by the server, assumes one of the following values:

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

3.2.7.1.5 Result-set-status

The server 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:

Result-set Status	Meaning
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 server includes this parameter if the value of Sort-status is 'failure'. It includes one or more diagnostic records.

3.2.7.1.7 Result-Count

The server may use this parameter to report the size of the output result set. The server is never obligated to supply this parameter, there is no default value, and the client should not draw any conclusion from its omission.

3.2.7.1.8 Other-information

This parameter may be used by the client or server for additional information not specified by the standard.

3.2.7.1.8 Reference-id

See 3.4.

3.2.7.2 Duplicate Detection Service

The Duplicate Detection service allows the client to request that the server analyze one or more result sets in terms of potential duplicates and to construct a new result set according to client-specified criteria for detecting, retaining, grouping, and ordering the records including duplicates.

The following notation is used in the "Client Request" and "Server Response" column:

- [0,1] means parameter is optional, not repeatable; i.e. zero or one.
- 0+ means parameter is optional, repeatable; i.e. zero or more.
- 1 means parameter is mandatory, not repeatable; i.e exactly one.
- 1+ means parameter is mandatory, repeatable; i.e. one or more.

Parameters of the Duplicate Detection Service

Parameter Name	Client Request	Server Response	Condition	Reference
Input Result Set Id	1+			3.2.7.2.1
Output Result Set Name	1			3.2.7.2.1
Applicable Portion of Record	[0,1]			3.2.7.2.2
Duplicate-detection Criterion	0+			3.2.7.2.3
Clustering	[0,1]		May be omitted if representative record only is to be retained (if Retention Criterion is 'number of entries' and its value is 1). Otherwise must be supplied.	3.2.7.2.4
Retention Criterion	1+			3.2.7.2.5
Sort Criterion	0+			3.2.7.2.6

Parameter Name	Client Request	Server Response	Condition	Reference
Status		1		3.2.7.2.7
Result count		[0,1]	Must occur if Status is 'success'.	3.2.7.2.8
Diagnostic		0+	Must occur if Status is 'failure'.	3.2.7.2.9
Other-information	[0,1]	[0,1]		3.2.7.2.10
Reference-id	[0,1]	[0,1]	See 3.4	3.2.7.2.11

3.2.7.2.1 Input Result Set Id and Output Result Set Name

The client identifies one or more transient result sets belonging to the current Z-association. The server is to logically merge the sets (removing duplicates and ordering equivalence classes according to the parameters below) into a single result set, specified by the parameter Output Result Set Name.

3.2.7.2.2 Applicable Portion of Record

The client may specify what portion of the record is subject to matching (for example, one or more fields) for purposes of duplicate-detection. If this parameter is omitted, the server decides what portion of the record is subject to matching.

3.2.7.2.3 Duplicate-detection Criterion

For modeling purposes, a temporary, intermediate result set (not the output result set) is assumed to be created, which includes all of the result set items from all of the input result sets (including duplicate result set items). The server applies duplicate-detection criteria supplied in this parameter (or if the client omits this parameter, the server applies whatever duplicate detection criteria it chooses) to partition the intermediate result set into one or more *equivalence classes* where two result set items are considered *equivalent* if they are duplicate. That is, the partitioning has the following properties:

- Every result set item from one of the input result sets is in exactly one class
- Any two result set items are in the same class if and only of they are duplicates

The server distinguishes a single result-set item within each equivalence class as the *representative record* for that class. The selection of representative record might be based on the value of the parameter Sort Criterion.

The client may specify one or more criteria for detecting duplicates. These include the following (the list is subject to extension):

level of match	If this criterion is included, the client specifies a level of match in terms of a percentage. For example, fingerprints might be duplicates based on a 60% match; 100% might mean that records are duplicates only if they are identical.
Case sensitive	
Punctuation sensitive	
Regular expression	If this criterion is included the client supplies a regular expression to govern matching.
result-set duplicates	Two result set items are *result-set duplicates* if they point to the same database record.

3.2.7.2.4 Clustering

The client indicates one of the following:

Clusters	The output result set is to contain one item for each equivalence class. For each equivalence class, create a result set item for the representative record only and maintain duplicates as metadata. (Records may subsequently be presented either as (a) representative record with duplicates attached as metadata, using, for example, GRS; or (b) as a cluster record, using an appropriate cluster syntax.)
Individual Entries	Create individual result set items for representative records as well as duplicates that are to be retained (according to Retention Criterion). Order the output result set such that records within an equivalence are grouped together. The parameter Sort Criterion may be supplied, to indicate how the records within a class are to be ordered.

This parameter may be omitted only if 'Number of entries' is supplied as a retention criterion (parameter Retention Criterion) and the value supplied is 1.

3.2.7.2.5 Retention Criterion

The client specifies one or more criteria for how records are to be selected for inclusion in or exclusion from each equivalence class. These include the following (the list is subject to extension):

1. Number of entries	If this criterion is selected, the client supplies a number, N>0, meaning retain (up to) N entries in each equivalence class. N=1 means retain the representative record only. This value may be used in combination with (3) and/or (4), but not (2).

2. Percent of entries	If this criterion is selected, the client supplies a percentage, xx, meaning retain xx percent of the entries in each equivalence class. xx=100 means retain all entries. This value may be used in combination with (3) and/or (4), but not (1).
3. Duplicates only	Discard representative record. This value should not be specified unless the value of parameter Clustering is 'Individual Entries'. This value may be used in combination with (1) or (2), and/or (4).
4. Discard result-set duplicates	This value may be used in combination with (1) or (2), and/or (3). If used with (1) or (2) the result-set duplicates should be discarded first (before entries are selected).

3.2.7.2.6 Sort Criterion

The client may provide one or more sort criteria for selecting the representative record as well as for ordering records within an equivalence class.

This parameter will affect the ordering of result set items only within an equivalence class (it does not affect the ordering of equivalence classes). If the value of parameter Clustering is 'Clusters' then this parameter will have no effect whatever on the result set order (though it may be supplied anyway, to govern the selection of representative records as well as the order in which duplicates are presented within a single cluster record).

More than a single sort criterion may be supplied; if so, the order in which they are supplied is from major to minor, and only the first criterion supplied is used to govern selection of a representative record. The sort criteria include the following (the list is subject to extension):

Most Comprehensive	Select the longest (more comprehensive) record as the representative record; order duplicates within an equivalence class by descending comprehensiveness.
Least Comprehensive	Select the shortest (least comprehensive) record as the representative record; order duplicates within an equivalence class by ascending comprehensiveness.
Most Recent	Select the most recent record as the representative record; order duplicates within an equivalence class by ascending age.
Oldest	Select the oldest record as the representative record; order duplicates within an equivalence class by descending age.
Least Cost	Select the least expensive record as the representative record; order duplicates within an equivalence class by ascending cost.
Preferred Database	Select a record from the most preferred database as the representative record; order duplicates within an equivalence class corresponding to order of preference of databases. When this criterion is supplied the client includes a list of databases in order of preference.

3.2.7.2.7 Status

The server indicates a status of 'success' or 'failure'.

3.2.7.2.8 Result Count

If the value of parameter Status is 'success' then the value of this parameter is the size of the output result set.

3.2.7.2.9 Diagnostic

The server may always include one or more diagnostics in the response. If the value of parameter Status is 'failure', at least one diagnostic must be included.

3.2.7.2.10 Other-Information

This parameter may be used by the client or server for additional information, not specified by the standard.

3.2.7.2.11 Reference-id

See section 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 server defined. The client 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.

Note: The Z39.50 term list abstraction is intended as a generalization of the concept of an index corresponding to a search access point. A term list may but need not necessarily be an index, or correspond to a search access point.

Parameters of the Scan Service

Parameter Name	Client Request	Server Response	Reference

Parameter Name	Client Request	Server Response	Reference
Database Names	m		3.2.8.1.1
Term-list-and-start-point	m		3.2.8.1.2
Step-size	o	ia	3.2.81.3
Number-of-entries	m	m	3.2.8.1.4
Position-in-response	o	o	3.2.8.1.5
Scan-status		m	3.2.8.1.6
Entries		o	3.2.8.1.7
Other-information	o	o	3.2.8.1.8
Reference-id	o	ia	3.4

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 client 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 client 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 server 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 client omits this parameter, the step size is selected by the server, and the server includes the selected step size in the response.

3.2.8.1.4 Number-of-entries

The client indicates the proposed number of entries to be returned. The server 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 client 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 client requests terms immediately preceding the starting point term.

The server 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 client 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 client 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 client 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.

Example Illustrating the Semantics of the Position-in-response Parameter

Consider a term list that includes consecutive terms A, B, C, D, E, and F; the Scan Request specifies C as the starting term, and 2 for Number-of-entries. If the value of Position-in-response is 1, terms C and D are to be supplied. If the value of Position-in-response is 2, terms B and C are to be supplied. If the value of Position-in- response is 3 then terms A and B are to be supplied. If the value of Position-in-response is zero, Terms D and E are to be supplied. If the value of Position-in-response is minus 1, then terms E and F are to be supplied; etc.

3.2.8.1.6 Scan-status

The server indicates the result of the operation. The defined values are:

Value of Scan-status	Meaning
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 client request.
partial-4	Not all of the expected entries can be returned because the operation was terminated by resource-control, by server.
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 server:

- 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 that is less than N, and may be zero (reason specified by Scan-status)
- And 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 server 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.
  Note: A “Count” is a number of records. There is no mechanism provided by this standard to indicate the count of occurrences of the term.
- Other information: additional information concerning the entry

3.2.8.1.8 Other-information

This parameter may be used by the client or server 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 a client to create, modify, or delete a task package at the server. The server 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 server is outside the scope of Z39.50. The extended services defined by this standard are listed in section 3.2.9.1.2. Definitions of those services are included in Appendix EXT.

The client sends an ES Request to the server requesting execution of a task. The request includes parameters that the server uses to construct the task package. The server checks the request for validity, for consistency with the user's access privileges, and possibly for other

server-dependent limitations. The server 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 client. The ES operation consists of a request from the client and a response from the server, 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 server 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 server 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 a client, 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 a client 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 that are specific to the particular extended service. Among the common parameters (indicated in the table below, listed under "task package parameter" in the right column), some are supplied by the client as parameters in the ES request, and are used by the server to form the task package; some of those supplied by the client may be overridden by the server. Others are supplied by the server. The specific parameters are derived from the parameter Task-specific-parameters of the ES request (see Appendix 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."

Parameters of the Extended Services Service

Parameter Name	Client Request	Server Response	Task-Package Parameter	Reference
Function	m			3.2.9.1.1
Package-type	m			3.2.9.1.2
Package-name	o		o	3.2.9.1.3
User-id	o		o	3.2.9.1.4
Retention-time	o		o	3.2.9.1.5
Permissions	o		o	3.2.9.1.6
Description	o		o	3.2.9.1.7
Server-reference			o	3.2.9.1.8
Creation-date-time			o	3.2.9.1.9
Task-status			o	3.2.9.1.10

Parameter Name	Client Request	Server Response	Task-Package Parameter	Reference
Package-diagnostics			o	3.2.9.1.11
Task-specific-parameters	m		(Task-specific)	3.2.9.1.12
Wait-action	m			3.2.9.1.13
Elements	ia			3.2.9.1.14
Operation-status		m		3.2.9.1.15
Operation-diagnostics		ia		3.2.9.1.16
Task-package		ia	o	3.2.9.1.17
Other-information	o	o		3.2.7.1.18
Reference-id	o	ia		3.4

3.2.9.1.1 Function

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

If the function is Delete or Modify, the server is to delete or modify the task package specified by the parameter Package-name. A server 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 client requests that if the specified task has not been acted on, it should not be started. If the task is active, the server should either terminate the task or refuse the request.

If the function is Modify, the client 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 server does not modify that parameter within the task package (thus to return a parameter to its default value, a client 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 Appendix 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 client 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 client 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 server may or may not allow a client to supply a user id different from its own.

3.2.9.1.5 Retention-time

The client may optionally specify a retention period (e.g. 2 hours, 3 days, 1 week), which may be overridden by the server. When the retention time has passed, the server 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 client may indicate who may access the task package. If the client does not supply this parameter, only the creating user may do so. See 3.2.9.3.

3.2.9.1.7 Description

The client 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 Server-reference

The server may supply a unique identifier for the task package.

3.2.9.1.9 Creation-date-time

The server supplies the date and time that the task package was created.

3.2.9.1.10 Task-status

The server indicates the status of the task. Values are 'pending,' 'active,' 'complete,' and 'aborted'. See also 3.2.9.5.

3.2.9.1.11 Package-diagnostics

The server 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 client indicates whether the server 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:

Value of Wait-action	Meaning
wait	The server must perform the task before issuing the ES response (unless the operation aborts; see section 3.2.9.4). If the tar-get 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 server accepts the request, it includes the parameter Task-package in the response.
wait-if-possible	The client requests that, if possible, the server perform the task before issuing the ES response and include the task package in the response. If not possible, the server should proceed as though the value were 'do not wait'.
do-not-wait	The client does not request that the server attempt to perform the task before issuing the ES response. However, if the server does perform the task before issuing the response, then the response may include the task package.
Do-not-send-task-package	The server 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 client may optionally include this parameter if Wait-action is other than 'do-not-sent-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:

Value of Operation-Status	Meaning
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).

See also 3.2.9.5.

3.2.9.1.16 Operation-diagnostics

The server may supply additional diagnostic information if Operation-status is 'failure'.

3.2.9.1.17 Task-package

If Operation-status is 'done,' the server 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 client or server 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

Servers 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").

Note: Thus if a server claims to support Extended Services, it supports the ES database to the extent, at minimum, that if a client searches IR-Extend-1 the server will not fail the search because there is no such database. In the case where the server does not create task packages (see next note) it may always respond to searches that zero records were identified.

Records in the extended services database are task packages constructed from the Request-parameter-package parameter in ES requests (the server 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 server may (but need not) retain a task package until the requested task has completed; it may retain the task package until the client requests that it be deleted. A server may unilaterally delete a task package from the ES Database at any time.

Note: This means, as a practical matter, the server need not actually create a task package for a given task, in particular, 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 server receives an ES request it may immediately create a task package, with status 'pending,' before completely validating the request. The client 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 client may be able to determine that the request for that operation was received.

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

An extended services database will appear to the client as any other database supported by the server (records may be searched and retrieved by the Z39.50 Search and Retrieval facilities; search processing is defined locally by the server; the server may impose access control or exclude records to which the client 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 in Appendix ATR. The task package structures are defined in Appendix EXT.

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

Element Set Name	Meaning
Identification	The creating user's identification, the client-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 server 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, a client 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 server 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 server 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 server would do so only if that user has 'invoke' privilege for that persistent query. As another example, a server 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.

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

3.2.9.4. Aborted Operations

A client 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 connection is lost), the client 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 client may search the ES database (possibly in a subsequent Z-association) for information that would otherwise have been returned in the response.

3.2.9.5 Description of Status Parameters

Task-Status and Operation-Status (3.2.9.1.10 and 3.2.9.1.17) both apply to Extended Services in general; individual Extended Services may define specific additional statuses (for example the Update Extended Service defines Update-status and Record-status) Operation-status and Task-status distinguish the ES operation from the ES task. The "ES operation" is the ES request followed by the ES response, the result of which is the spawning of an ES task, which may be monitored by Task-status. Thus Operation-status is set only once, after the operation is complete, in contrast to Task-status, a dynamic status that changes as the status of the operation changes. Neither of these two parameters conveys any status information specific to the ES type.

Task-status

Task-status is a general ES parameter, and it exists only in the task package (i.e. it is not an explicit parameter of the ES response). Values are:

- 'pending',
- 'active',
- 'complete', and
- 'aborted'

Its purpose is to allow the client, by repetitively retrieving the package, to monitor the progress of its execution, from initiation of the ES operation until completion of the task. But it is not intended to provide status specific to the type of task, and in particular, it is not intended to convey whether the task was completed successfully, only that it completed.

In the abstract, when the server receives the ES request, if it passes preliminary inspection and the task is queued, Task-status is ‘pending’; if it does not pass preliminary inspection then at the server discretion there may not even be a task package created (see Operation status) but if there is, its status is set to 'aborted'. Once the task starts (which may or may not be before the server sends back the ES response) the status is set to 'active'. If it subsequently aborts, it is set to 'aborted' and if it subsequently completes, it is set to 'complete’. Note that the ‘pending’ state need not occur; the server might start the task immediately upon receipt of the task package.

Operation status

Operation-status, in contrast to Task-status, is always included in the ES response. (Task-status exists only in the task package, and the task package might not be included in the ES response.) Its values are:

- 'done',
- 'accepted', and
- 'failure'.

The operation status of 'done' means the task package is included in the ES response (in that case this status is redundant); 'accepted' corresponds to Task-status of 'pending' or 'active', and 'failure' corresponds to the case where the task package was not even set up because the task did not pass preliminary inspection.

3.2.10 Explain Facility

The Explain facility allows a client to obtain details of the implementation of a server, including databases available for searching, attribute sets and diagnostic sets used by the server, and schema, record syntax and element specification definitions supported for retrieval. Servers 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 Appendix ATR (which defines a set of Use attributes and imports bib-1 non-Use attributes); and
- Support the Explain syntax, which is defined in Appendix 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 client as any other database supported by the server. 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 client 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 server 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 server 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 server 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 server should also support one or more of the following relation attributes: 'less than', 'less than or equal', 'greater than', and 'greater or equal'.

Clients should not in general expect that the explain database is searchable using the bib-1 truncation attribute, completeness attribute nor any of the alternative values of the relation, position and structure attributes defined in bib-1. However, servers 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 client 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 client 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) are given in the table below, as well as the keys for each category. In 3.2.10.3 each category is described in detail.

A client should adhere to the following rules when searching an Explain database by the predefined information categories.

- To search for information about the server, 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.

Category	**An Explain record in this category describes:**	**Key(s)**
TargetInfo	The server, including search constraints imposed by the server.	server name
DatabaseInfo	A database. 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
ExtendedServicesInfo	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
RetrievalRecordDetails	The elements of a retrieval record (for a particular record syntax, defined by a particular schema).	databaseName, schema oid, recordSyntax oid

Category	An Explain record in this category describes:	Key(s)
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, processing-context, name,oid
VariantSetInfo	A variant set definition; classes, types, and values, for a specific variant set definition supported by the server. Support by the server 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 server.	unit system name
CategoryList	Explain categories that the server supports.	(no key)

3.2.10.1.2 Searching for Information in a Particular Language

Elements intended to be presented to the user by the client 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 expiry 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 that are currently available. The use of these attributes by a client 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 Appendix REC, REC.1).

Explain records include key elements that serve to uniquely identify each record. Each Explain category is defined in term 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, HTML, or PDF. 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.

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 client may request summary information about all of the databases supported by a server 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
- Includes an element indicating the language of the "human readable text" elements of the record

These are logical descriptions, which do not reflect the possibility 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 TargetInfo

TargetInfo is information about the server. There is one such Explain record in the Explain database.

Brief elements:

- A name for the server (only one), in human readable text
- Recent news of interest to people using this server, in human readable text
- An icon used to represent this server (in machine presentable form)
- Whether named results sets are supported (result set names other that "default")
- Whether multiple databases can be searched in one search request
- The maximum number of concurrent result sets supported, for a given Z-association. Thus for example, if the value is 2, say a client creates (via a search) result set A, then B; a subsequent attempt to create C will exceed the maximum (note however that the server action when the maximum is exceeded is not specified; for instance, the server might unilaterally delete A in order to create C, or it might return an error), however, if the client first deletes B, then it could create result set C without exceeding the maximum.
- The maximum size (in records) of a result set
- The maximum number of terms allowed in one search request. This is subject to server interpretation. It could mean maximum number of operands in the search, or operands of the form "attributesPlusTerm". Or, suppose for example result set A is the result of the search "cat" AND "dog"; and result set B is the result of the search (Result set A) AND "moon". The latter search has only one "Term", it has two "Operand"s, and all together the search involves three terms.
- A timeout interval after which the server will trigger an event if no activity has occurred.
- A "welcome" message from the server to be displayed by the client.
- Contact information for the organization supporting this server
- A description of the server, in human readable text
- A set of nicknames or alternate names by which the server is known

- Restrictions pertaining to this server, in human readable text
- A payment address (e.g. business office) for the organization supporting this server
- Hours of operation
- A list of supported database combinations
- Internet address and Port number
- 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

Database-Info is a 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 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 server 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 that the server expects the client to display to the user if possible, in human readable text
- Description and contact information for the database producer, database supplier, and for 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

Schema-Info is descriptive information about a database schema. There is one Explain record for each schema supported by the server.

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

TagSetInfo is included as an Explain category to allow a client to retrieve (and in turn allows the end-user to discover) information about a tagSet supported on the server. The server can convey, for a supported tagSet, the identifier of the tagSet (an Object Identifier), and for each supported element, its name, tag, datatype, and description.

This Explain category might support collaboration between the end-user and client, where the client retrieves the TagSetInfo information and conveys names and descriptions to the end-user, who then decides which elements are of interest and requests the client to retrieve those specific elements. Thus the semantics of an individual element are conveyed transparently to the end-user, the client never needs to understand their meaning, only their datatype and tag; and the end-user never knows the tag or datatype.

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

Record-Syntax-Info is descriptive information about a record syntax. There is one Explain record for each abstract record syntax supported by the server.

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

Attribute-Set-Info is 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.
  - Its name
  - Description
  - Its value
  - Names of equivalent attributes. Equivalences are derived from the attribute set definition (not from the servers behavior)
- Description of the attribute set

3.2.10.3.7 TermList-Info

TermList-Info is 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 server 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

Extended-Services-Info is 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

Attribute-Details contain 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

Term-list-Details is 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) 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

Element-Set-Details is 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

Retrieval-Record-Details is 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

Sort-Details is a description of the sorting capabilities supported by the server. 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, structured
  - Whether the key is case-sensitive

3.2.10.3.14 Processing-Info

Instructions, representing how the server believes the data should be processed by the client 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 server 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

Variant-set-info is descriptive information about a variant set definition supported by the server; 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

Unit-info is descriptive information about a unit system definition supported by the server.

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

Category-list is a list of the Explain categories supported by the server. 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 server is supporting a category not defined in this standard.

- The original search term. (This is for information categories where the server 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 a client or server to abruptly terminate all active operations and to initiate termination of the Z-association.

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 client or server:

- 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.

Parameters of the Close Service

Parameter Name	Request	Response	Note	Reference
Close-reason	m	m		3.2.11.1.1
Diagnostic-Information	o	o	server only	3.2.11.1.2
Resource-report-format	o		client only	3.2.11.1.3
Resource-report	o	o	server only	3.2.11.1.3
Other-information	o	o		3.2.11.1.4
Reference-id	ia	ia		3.4

3.2.11.1.1 Close-reason

This parameter indicates the reason why the client or server 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 effect 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 server may include an optional text message, providing additional diagnostic information.

3.2.11.1.3 Resource-report-format and Resource-report

When the client issues a Close request: the client may include the parameter resource-report-format to request that the server include a resource report (see 3.2.6.1.1) in the response. The server'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 client included the parameter resource-report-format.

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

3.2.11.1.4 Other-information

This parameter may be used by the client or server 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 client.

The server should omit Reference-id on a Close request. On a Close response, if the server is responding to a Close request that included Reference-id, the server may either include Reference-id using the identical value, or it may omit the parameter. If the server is responding to a Close request that did not include a Reference-id, the server 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 server as part of an aggregate Present response, i.e. a Segment request or Present response.

Throughout Section 3.3, the term "record" is used as follows:

- Unless otherwise qualified, it indicates a "response record," i.e., retrieval record or surrogate diagnostic.
- Except within Section 3.3.3, it refers to a "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 Section 3.3.3, a set of records is said to "fit into a segment" if the sum of their sizes, not including protocol control information, does not exceed Preferred-message-size. For the Present operation, the server might be unable to fit the requested records in a single segment, because of record or message size limitations. In that case, the server 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 server 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 server 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 server 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 server 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 server is not able to return all of the records requested, because of message size limitations, the server should fit as many records as possible.

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

Illustration

Assume that the server 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 server returns records 1 through 6. In case (b), except as noted below (see "Exception"), the server substitutes a diagnostic record for retrieval record 7, indicating that the

record exceeds Preferred-message-size. In case (c) the server 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 server 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 server 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 server 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 client 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 client to continually allocate and hold local buffer space for worst-case records. Note also that this intended purpose is defeated if the client routinely requests a single record.

3.3.2 Level 1 Segmentation

When level 1 segmentation is in effect, the server 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 server 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.

Assume that the client 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 server 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 client 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 server 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 server 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) apply 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 sub string 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 client cannot necessarily predict how a record will be segmentation into fragments by the server 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 server 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 client 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 client 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 server 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 server fits a starting fragment of the following record (if possible), and the segment is sent as a Segment request. The server then fits the remainder of that record into the next segment (if possible; and if not possible, 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 server 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 stated more formally as follows:

Assume that the client 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 server 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
   - R=P+1
3. **Note:** Having reached this step, record R will not fit in the current segment.
   If the Present request has included Max-segment-count and the server 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 client 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: 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:** 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 client 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 server is able to segment record 5, but only at element boundaries; the server will not let the elements span fragments.
   Note: this means that the server 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 server 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 server 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 server can segment the smaller records into 100 byte fragments (or multiples).

Segments 1 through 3 are as in case 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 server 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

3.3.3.4 Segmentation and Access-or Resource-Control

A segmentation sequence may be interrupted by Access-control or Resource-control. A server might send one or more segments and follow (prior to sending the Present Response) with either an Access-Control request (for example because the next set of segments requires authorization) or a Resource-Control request (for example indicating that the next set of segments will cost more).

3.4 Operations and Reference-id

A request from the client of a particular operation type initiates an operation, which is terminated by the respective response from the server. The following operation types are defined: Init, Search, Present, Delete, Resource-report, Sort, Scan, Extended-services, and Duplicate-detection. (Thus each client 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 client, the client 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 client to server 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 the following 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 client and have meaning only within the client 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.)

Even though the client and server should supply reference-ids as described above, in the case where an incorrect reference-id is supplied (and so long as ‘serial operations’ is in effect), the receiver of the incorrect reference-id may simply ignore it, or may declare a protocol error (i.e. issue a Close with close reason 'protocol error'), and this standard takes no position on which choice is better. An example of an incorrect reference-id would be when the client sends a Search Request with a reference-id and the server responds with a reference-id of a different value, or no reference-id at all. Another example: the client sends a Search Request, the server sends an Access-control Request (with the correct reference-id) and the client sends an Access-control Response with a different reference-id.

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 client 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 server; the server may process concurrent operations in any manner it chooses.

Example:

The client 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 client 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 client might then issue a Present request (against the result set created by the second operation), initiating another operation. In that case, the client 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 server 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 client should not initiate concurrent Search operations with the same value of Result-set-id.

Other than the restriction cited above (that when the client 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 re-use or management of Reference-ids by the client. The client might re-cycle Reference-ids randomly among users, or it may manage local threads by assigning different Reference-ids to end-users. The server is not required to know how the client manages Reference-ids, or in particular, that the client is using Reference-ids to distinguish different users. There is no requirement for the server to have any knowledge of multiple end users at the client, the server interacts only with the (single) client.

3.6 Composition Specification

For each database supported the server defines one or more schemas (see 3.1.5), and designates one as the default schema. For each schema, the server 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 server designates one as the default element set name for the database.

Note: The server 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 server applies an abstract record structure (defined by a schema for the database to which that record belongs) to form an abstract database record, to which the server applies an element specification to form another instance of the abstract database record (the latter might be a null transformation), to which the server applies a record syntax, to form a retrieval record.

If the client 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 server 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 server 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 server 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 server 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 server 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 server applies that element set name.
- If not, the server 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 server applies that element set name; if not, the server applies the default element set name for the database.

A server 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 server 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 client knows the server's

definition of "brief" for a given schema, it should not assume that any particular elements are included.

Element set names are case-insensitive. (See the discussion of case-sensitivity of database names and result set names in 3.2.2.1.3. Element set names, as database names, are often passed around outside of the protocol, for example, in profiles.)

The client may specify a "preferred-record-syntax," which the server applies (to the abstract database record formed by the application of the element set name) to form a retrieval record. If the client does not specify a preferred-record-syntax, the server 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 server applies a record syntax, indicated as described above. The term "record syntax" has the following meaning:

- When specified by the client (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 server, with a transfer syntax) that the client requests the server use for retrieval records.
- When specified by the server, 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.

When Server cannot Supply a Record According to Requested Syntax

When Preferred-record-syntax is supplied (and Comp-spec is not supplied, or if it is supplied, no record syntax ids are included within) and a particular record is not available in the requested syntax , the server should return a surrogate diagnostic such as 238: "Record not available in requested syntax", or a variation such as 1070: "user not authorized to receive record(s) in requested syntax" (used as a surrogate). The server may fail the request in the case where *none* of the requested records is available in the requested syntax (227: "No data available in requested record syntax "), or when the syntax is not supported (239: "Record syntax not supported"), or user is not authorized (1070, used as a non-surrogate). In any case, whenever preferredRecordSyntax is supplied, the server should not supply any records in any other syntax.

When Requested Syntax is not Supplied

When Preferred-record-syntax is not supplied (and Comp-spec is not supplied, or if it is supplied, no record syntax ids are included within) the server may interpret the omission to mean that the client wishes the server to select an appropriate syntax (which may be different for different record). However the server is free to treat this case as it sees fit; it may:

- Fail the request (with diagnostic 1071: "preferredRecordSyntax not supplied" used as a non-surrogate, diagnostic; or 1069: "No syntaxes available for this request");
- Select a syntax for some records and decline to select a syntax for others (with diagnostic 1071, used as a surrogate diagnostic); or
- Select a syntax for each record.

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

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

If the server claims support for the Prox operator, the server 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 server 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 client, 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 client traverses the tree according to a left post-order traversal, to produce a sequence of (simple) operands and operators, which is transmitted to the server.

At the server, 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. Each is one of the following:

(a) AttributeList+term -- In which case it represents 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 -- In which case it represents the set of database records represented by the transient result set identified by ResultSetId.

(c) Restriction operand (ResultSetId+AttributeList): In which case it represents 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 server 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) -- In which case it represents the records identified by that result set.

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 server 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) there will be one object remaining on the stack (otherwise the query is in error), representing a set of database records, which is the result of the query.

Example

Suppose databaseName = A; resultSetName = 1; query = " 'dog' ". Subsequently, a follow-on query specifies

databaseName = B; resultSetName = 2; query = " 'dog' OR 'resultSet 1' ". That is, the second query asks for records from database B containing the word "dog" or records from result set 1 (result set 1 contains records from database A). The latter search, though it pertains only to database B (and there are no records from result set 1 that belong to database B) should nonetheless include records from A. (See 3.7.1 (a) and (b). Note that when an operand is of the form AttributesPlusTerm, it is evaluated against the specified databases, but when an operand is of the form resultSetName, it is not.)

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 server 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 server 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 server maintains this information is not prescribed by the standard. Appendix ERS (non-normative) provides examples.

An implementation may support proximity without supporting the extended result set model for proximity. For example, it might support "cat near hat" and not support "R1 near R2" (where R1 and R2 are result sets representing 'cat' and 'hat' respectively).

3.7.3 Restriction and the Extended Result Set Mode l

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 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 server maintains information associated with each record represented by the result set, that 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 server maintains this information is not prescribed by the standard. Appendix ERS (non-normative) provides examples.

If an implementation supports result set restriction, then it implicitly supports the extended result set model for restriction (the model is implicit in the semantics). However, supporting the extended model for restriction is an abstract concept for which there is no conformance requirement. It simply means that the server maintains some information (necessary to carry out the operation) and the manner in which the information is maintained is not prescribed by this standard.

4. Protocol Specification

This section (4) specifies the formats, procedures, and conformance requirements for the Z39.50 protocol, governing the transfer of information between a Z39.50 client/server pair. Sections 4.1 and 4.2 respectively describe the formats and rules for exchange of Z39.50 application protocol data units (APDUs). An APDU is a unit of information, transferred between client and server, 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 (Application Protocol Data Units). An APDU is a unit of information transferred between a client and server. The abstract syntax is is described using the ASN.1 notation defined in ISO 8824.

The comments included within the ASN.1 specification are part of the standard.

See Appendix ASN1, ASN1.1

4.2 Protocol Errors

Syntactical errors in received APDUs are considered to be protocol error (with the following exceptions: Unknown data elements, and unknown options within the Options data element, will be ignored on received Init APDUs). Incorrectly formatted APDUs or APDUs with invalid data are also considered to be protocol errors, as are incorrectly sequenced messages. Additional conditions that may be treated as protocol errors are described in 4.4.2.2.

This standard does not specify the actions to be taken upon detection of protocol errors. A system detecting a protocol error may:

- issue a Close request, with close-reason ‘protocol error’ (when version 3 is in force);
- Terminate the connection; or
- Ignore the error.

4.3 Encapsulation

Z39.50 APDUs may be included, or *encapsulated,* within other APDUs. The rules for encapsulation are as follows.

1. Encapsulation is negotiated during initialization. Option bit 15 is designated. If the client sets option bit 15 in the Init request, then:
   - The client proposes that "encapsulation be in effect" for the Z-association (or limited

to the Init operation if version 2 is proposed; see rule 5 below).

- The Init request may also (but need not) include one or more APDUs as described in 2.
- If the server also sets option bit 15, then encapsulation is in effect for the Z-association (or limited to the Init operation, if version 2 is negotiated). In that case, if the Init request had included encapsulated APDUs, the server's behavior (as concerns the Init response) is as described below.
- If the server does not set option bit 15, encapsulation is not in effect. (If the server does not even recognize option bit 15, it will ignore it and not set the bit in the response, as per the general rules for negotiation.) In this case if the server includes information that appears to be encapsulated APDUs in the Init response, the client should not assume that the information is encapsulated APDUs, and this specification does not prescribe client behavior in this situation.

2. If encapsulation is in effect, then within an operation-initiating APDU the client may include, within otherInfo (or, if the APDU is Init, within the userInfo parameter using UserInfo-1 (see Appendix USR) to simulate OtherInfo; (see Use of Init Parameters for User Information in Appendix USR), one or more APDUs, including any operation-initiating APDU (excluding Init), optionally followed by Close. These are referred to as "encapsulated APDUs".
   The first APDU in this chain of nested APDUs (i.e. the one that is not encapsulated in any other) is called the “base APDU”.
   The first encapsulated PDU is included in the otherInfo parameter of the base PDU (or in the userInfo parameter, if the base APDU is Init). The next encapsulated APDU is included in the otherInfo parameter of the first encapsulated APDU, and so on. Each encapsulated APDU is included in otherInfo as CHOICE externallyDefinedInfo, where the oid for the external is 1.2.840.10003.2.1 (the oid for Z39.50 APDUs).
3. There may be at most a single occurrence of an encapsulated APDU within the base APDU, and at most a single occurrence of an encapsulated APDU within an encapsulated APDU (thus arbitrary nesting is permitted, but multiple threads are not). When this is not the case, the prescribed behavior is not defined by the protocol.
4. The use of the otherInfo and/or userInfo parameter is specified in order to support encapsulation within version 2 or 3. In a future version of the protocol, there might be explicit encapsulation parameters in certain APDUs.
5. For version 2, encapsulation may be supported only within the Init operation. Thus, when version 2 is negotiated, encapsulation may also be negotiated, but for use only within Init.
6. When the client includes encapsulated APDUs the intent is that the server execute the APDUs serially in the order that they are nested (from shallowest to most deeply nested), and include APDU responses similarly nested in the APDU response to the base APDU, and formatted in the manner described in (2), that is, with each encapsulated APDU included in otherInfo as CHOICE externallyDefinedInfo, where the oid for the external is 1.2.840.10003.2.1.
7. If encapsulation is in effect, the server may not simply ignore encapsulated APDUs. The server may choose not to execute encapsulated APDUs, but if so, must include in the response to the base APDU an appropriate diagnostic (see 10), for example: "this specific sequence of APDUs is not supported".
8. Based on pre-screening analysis, the server may decide to execute neither the base APDU nor any encapsulated APDUs, for example in the case where in the server's opinion the client intent was that if the server did not expect to be able to execute the full sequence of APDUs, then it should not attempt to execute any of them. Similarly, in this case the server should include in the response to the base APDU an appropriate

diagnostic (see 10).

9. If encapsulation is in effect, the server may choose to execute some but not all of a sequence of nested APDUs. In that case it should not execute any encapsulated APDU following one that it chose not to execute (and similarly should include an appropriate diagnostic in the base-APDU response, see 10). Thus if there are N encapsulated APDUs, the server will always execute either none, or the first M APDUs (M less than or equal to N). The server should also not execute any APDUs following a APDU that it attempted to execute but execution failed.
10. The server may use the General Diagnostic Container format (see Appendix DIAG) to supply diagnostics in these cases described in 7, 8, and 9.
11. Access-control and resource-control apply to the operation initiated by the base APDU, from a modeling perspective. However, as a practical matter, Access-control and Resource-control formats may be developed to allow a server to indicate to what specific encapsulated APDU the Access-control or Resource-control request pertains. Similarly, for trigger-resource-control: the client might send a base APDU with several APDUs nested, and later send a trigger-resource-control request. From a modeling perspective it applies to the base APDU. But the request could actually be asking (based on the report-id requested) "how far into the sequence are you?" and subsequently the client might issue a second trigger request specific to the embedded request, which (again, from an operation model perspective) would still apply to the base APDU.
12. Encapsulation is not intended to support segmentation.

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 client or server
(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 and 4
(g) Assign values to APDU data elements according to the procedures of sections 3 and 4.1

4.4.2 Specific Conformance Requirements

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

The following table of 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.

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
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
21A	String identifier for schema	V3 (2001 only)	Note 10	4.4.2.2.30
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

Item	Feature	Version	Reference	Conformance
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
34.1	Result-count parameter of Sort Response	V2 and V3	3.2.7.1.7	4.4.2.2.26
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
44	Duplicate Detection Service	V2 and V3	3.2.12	4.4.2.2.13
45	Negotiation Model	V2 and V3	Appendix NEGO	4.4.2.2.27

Item	Feature	Version	Reference	Conformance
46	Query type 104	V2 and V3	3.2.2.1.1	4.4.2.2.28
47	Encapsulation	V2 and V3	4.2.4	4.4.2.2.29

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 Appendix DIAG.
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.
10. The Z39.50 Present request may include a schema identifier allowing the client to request that records be supplied according to a specific schema. In Z39.50-1995 the identifier must be an ISO object identifer (OID), and it is assumed that the schema is a GRS-1 compatible schema (i.e. that the record syntax will be GRS-1). Z39.50-2001 allows the identifier to be a string (or an OID).

4.4.2.2 Detailed Requirements

4.4.2.2.1 Init, Search, and Present Services

(See items 1, 2 and 14 above.)

A system must support the Init, Search, and Present services.

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

A client 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, a client indicates that it does intend to utilize the Present service, and the server refuses, this does constitute non-conformance on the part of the server.

This requirement is independent of version.

4.4.2.2.2 Type-1 Query

(See item 3 above.)

A client must be capable of formulating a type-1 query within a Search request, and a server should expect to receive a type-1 query.

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

This requirement does not mean that any specific feature of the type-1 query must be supported. A server 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 condition 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 above.)

For version 2, the client may not use any of these features in a type-1 query. If server receives a type-1 query with any of these features, it may treat this condition as a protocol error.

For version 3, the client may but is not required to use any of these features in a type-1 query. The server 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 server receives a type-1 query which includes any of these feature 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 above.)

A client is not required to support queries of any of these types. A server should expect to receive, but need not support queries of these types. If a server 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 above.)

For version 2, a client may not use the type-102 query. If a server receives a type-102 query it may treat this condition as a protocol error.

For version 3, a client may, but need not support the type-102 query. A server 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: Z39.50 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, Extended Services, or Duplicate Detection

(See items 12 and 39 above.)

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 the any of these parameter.

4.4.2.2.7 Additional-ranges and Comp-spec Parameters on Present Request

(See item 15 above.)

For version 2, the client may not use these parameters. If the server receives one of these parameters it may treat this condition as a protocol error.

For version 3, the client is not required to, but may use either of these parameters. The server should expect to receive, but need not support either of these parameters. If the server 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 above.)

For version 2, as well as for version 3 when segmentation is not in effect, the client may not use these parameters; if the server 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 client may but is not required to support Max-segment-count. The server should

expect to receive, but need not support Max-segment-count. If the server 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 client may not use Max-segment-size or Max-record-size. If server receives either it may treat this condition as a protocol error.

- If level-2 segmentation is in effect:
  - The client may but is not required to support any of these three parameters. The server should expect to receive, but need not support any of these parameters. If the server 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 above.)

For version 2, the server may send diagnostics in a Search or Present response using the default form only. If the client 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 server may send diagnostics using the default or external form. The client should expect to receive diagnostics in either form.

4.4.2.2.10 Addinfo of Default Diagnostic Format

(See items 19 and 20 above.)

For version 2, when the server 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 client 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 above.)

For version 2, the server must not include multiple non-surrogate diagnostics in a Search or Present response; if it does so, the client 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 server may (but is not required to) include multiple non-surrogate diagnostics in a Search or Present response and if it does, the client must not treat this condition as a protocol error.

4.4.2.2.12 Segmentation

(See items 22, 23, and 24 above.)

For version 2, as well as for version 3 when segmentation is not in effect, the server may not send a Segment request, and if it does, the client may treat this condition as a protocol error.

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

4.4.2.2.13 Delete Service, Trigger-resource-control Service, Resource-report Service, Sort Service, Scan Service, Extended-Services Service, and Duplicate Detection Service

(See items 25, 30, 31, 34, 35, 36, and 44 above.)

A system is not required to support any of these services. They are independently negotiable. If the server 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 above.)

A system is not required to support either of these services. They are independently negotiable. If the client receives an Access-control or Resource-control request and the respective service is not in effect (or if the request occurs while the client is awaiting an Init response and the client 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 above.)

For version 2, the server may not return this value; if it does the client may treat this condition as a protocol error.

For version 3, the server may return this value.

4.4.2.2.16 Security-challenge-response and Diagnostic in Access-control Response

(See item 28 above.)

For version 2, the client must include in the Access-control response the parameter Security-challenge-response, and may not include a diagnostic. If the server receives an Access-control response that violates this rule it may treat this condition as a protocol error.

For version 3, the client 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 above.)

For version 2, the client may not use this parameter; if the server receives this parameter it may treat this condition as a protocol error.

For version 3, the client may, but is not required to include this parameter. The server should expect to receive, but need not support the parameter. If the server 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 above.)

For version 2, the server may not return either value for this status; if it does the client may treat this condition as a protocol error.

For version 3, the server may return either value.

4.4.2.2.19 Close Service

(See item 37 above.)

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 above.)

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, Extended Services, and Duplicate Detection Request

(See item 40 above.)

The parameter Other-information may occur in a Scan, Sort, Extended Services or Duplicate Detection 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 above.)

For version 2, as well as for version 3 when concurrent operations is not in effect, if a client attempts to initiate concurrent operations (i.e. attempts to initiate an operation when an operation is already active), the server 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 above.)

A system may choose to support or not support named result sets.

If the server receives a Search request where the value of the parameter Result-set-id is other than 'default' and the server does not support named result sets:

- If version 2 is in effect, the server should not treat this condition as a protocol error but should instead return an appropriate diagnostic.
- If version 3 is in effect, the server may treat this condition as a protocol error or may instead return an appropriate diagnostic.

4.4.2.2.24 InternationalString Definition

(See item 42 above.)

For version 2, a value of a parameter of ASN.1 type InternationalString must conform to the VisibleString definition. A system which 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 which 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 above.)

For both version 2 and version 3, a client may choose to support or not support the Reference-id parameter; a server must support the Reference-id parameter. Note, however, for version 3, client support of concurrent operations (see 4.4.2.2.23) implies support for the reference-id parameter.

4.4.2.2.26 Result-count Parameter of Sort Response

(See item 34.1 above.)

A client may support the Sort service and still not recognize this parameter (because it is newly defined in Z39.50-2001), as long as option bit 16 is not negotiated. Thus if option bit 16 is not negotiated and the client receives this parameter it may consider this to be a protocol error. If option bit 16 is negotiated, the client must recognize this parameter. The server is never required to support this parameter (thus even if option bit 16 is negotiated the server is never obligated to send this parameter).

This requirement is independent of version.

4.4.2.2.27 Negotiation Model

(See item 44 above.)

Neither the client nor server is required to support the negotiation model. However, if the client and server both set the "negotiation model" option bit both signify adherence to the model and may assume that negotiation is carried out in accordance with the model.

This requirement is independent of version.

4.4.2.2.28 Query Type 104

(See item 46 above.)

Neither the client nor server is required to support the type-104 query. If the type-104 option bit is negotiated, the client may send type-104 queries, and the server must recognize type-104 queries but is not required to support any specific external query definition.

This requirement is independent of version.

4.4.2.2.29 Encapsulation

(See item 47 above.)

Neither the client nor server is required to support encapsulation, it is a negotiated feature. Rule for negotiation of encapsulation are supplied in 4.2.4.

4.4.2.2.30 String Indentifier for Schema

(See item 21A above.)

Neither the client nor server is required to support this, it is a negotiated feature. If option bit 21 is negotiated, the client may send string identifiers for schemas, and the server must recognize them as valid identifiers but is not required to support any specific schemas.

4.4.3 Z39.50 Version 3 Baseline Requirements

This section details the minimum requirements, beyond version 2, for a Z39.50 implementation to claim conformance to version 3, that is, to indicate in an Init request or response that it supports version 3.

The Version 3 Baseline Requirements include core and conditional requirements. Core requirements apply to all version 3 implementations. Conditional requirements pertain to features that are optional in version 2. These are requirements that are applicable in version 3 only if a particular, optional feature is implemented, and only if that feature was part of version 2. Only the Delete, Access-control, and Resource-report Services are affected.

Requirements are listed separately for client and server.

In the rules below, the terms *accept*, *recognize* and *support* have the following meaning:

- **Accept**: Rules stating that a system must accept a particular object mean that the system must not declare a protocol error due to receipt of that object; the system is not required to support the function (see “support” below) associated with that object
- **Recognize**: Rules stating that a system must recognize a particular object mean that the system must not declare a protocol error due to receipt of that object, and must either support the function associated with the object (see below) or must respond appropriately (e.g. with a diagnostic) that it does not support the function.
- **Support**: Rules stating that a system must support a particular function mean that the system must implement the function and exhibit behavior prescribed in the standard pertaining to that function.

4.4.3.1 Core Requirements

4.4.3.1.1 V3 Core Requirements for Client

1. The client must accept the parameter additionalSearchInfo in a Search response
2. The client must accept both the VisibleString and InternationalString form of the addInfo parameter within a diagnostic record
3. The client must accept diagnostics in both the default and external forms
4. The client must accept multiple non-surrogate diagnostic records in a Search or Present response
5. The client must accept the parameter otherInfo in any APDU
6. The client must support receipt of a Close request
7. In the absence of character-set negotiation, the client must accept all values conforming to the GeneralString definition for parameters of ASN.1 type InternationalString

4.4.3.1.2 V3 Core Requirements for Server

1. The server must recognize search terms in a type-1 query of type OCTET STRING, INTEGER, InternationalString, OBJECT IDENTIFIER, GeneralizedTime, EXTERNAL, IntUnit, or NULL.
2. The server must recognize within a type-1 query (in addition to the global attribute set id) an attribute set id qualifying any individual attribute within the query

3. The server must recognize the operand 'resultAttr' within a type-1 query
4. The server must recognize the operator 'prox' within a type-1 query
5. The server must recognize (in addition to numeric attribute values) attribute values of form 'complex'(as defined in the ASN.1 for APDUs) within a type-1 query
6. The server must recognize a type-102 query
7. The server must accept the parameter additionalSearchInformation in a Search request
8. The server must recognize the parameter AdditionalRanges on a Present request
9. The server must recognize the 'complex' (in addition to the 'simple') form of the parameter recordComposition on a Present request
10. The server must accept the parameter otherInfo in any APDU
11. The server must support receipt of a Close request
12. In the absence of character-set negotiation, the server must accept all values conforming to the GeneralString definition for parameters of ASN.1 type InternationalString

4.4.3.2 Conditional Requirements

4.4.3.2.1 V3 Conditional Requirements for Client

If the client implements the Delete service as well as concurrent operations: the client must accept a value of 'failure-10' for the Delete-list-status on Delete response.

If the client implements the Resource-report service, the client must accept a value of failure-5 or failure-6 for Resource-report-status in Resource-report response.

4.4.3.2.2 V3 Conditional Requirements for Server

If the server implements access control, the server must accept an Access-control response where the parameter Security-challenge-response is omitted and which includes a dagnostic.

If the server implements resource control, the server must recognize the parameter OpId of the Resource-report request.