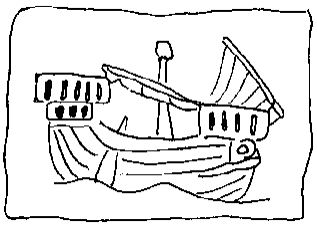
ARCA

Libraries Programme Project LIB-ARCA/2-3039



Title : ARCA Dictionary Schema Document

Document Reference : ARCA/T13/DSD

Version : Preliminary Final Version
11:45 June 4, 1997

Date : February, 1996

Author(s) : Alberto Catoni (CNR-CNUCE)
Sabrina Giuliano (CNR-IEI)
Mario Loffredo (CNR-CNUCE)
Oreste Signore (CNR-CNUCE)

Distribution : CNR-CNUCE/IEI
Fundacion Sancho El Sabio
Intecs Sistemi
Regione Toscana
SABINI
Università di Pisa

Abstract : This report presents the detailed description of the Dictionary component of the ARCA Target system. In particular, the document makes clear how the Z39.50 V3 EXPLAIN facility is implemented by using the Dictionary information.

Keywords : SR, Z39.50, OMT, EXPLAIN, ARCA Target Sytem, Dictionary, C++, YAZ

Reviewed by :

Approved by :

Document Status Sheet

Issue	Changes	Date	Reason
0.0	New document	14 December 1995	First draft
1.0	Editing	28 February 1996	Preliminary Final Version

Table of Contents

1 Introduction

Task 1.3 of the ARCA Project consists of producing a detailed description of each component of the SR Target (Dictionary, Kernel and OPAC Interfaces).

The results provided by this task are:

- The detailed design of the Kernel, described in the report ARCA/T13/DDD.
- The specifications of the API, contained in the report ARCA/T13/API, that for each API function gives the calling format, the meaning and the type of the input and output parameters, a brief free text description of the performed function.
- The detailed structure of the SR Target Dictionary, that constitutes the content of this report.

2 The Dictionary objects

2.1 Generalities

The Dictionary component plays a central role in the architecture of the whole ARCA project as it contains the information used both by the Target to enhance the services offered by the underlying OPAC and by the Origin to configure itself each time the user establishes a connection.

The first group of information are used by the Target to check the admissibility of a service request and, if possible, to arrange a query coming from an origin and make it processable by the OPAC native language. The second group of information are used by the Origin to generate only admissible search queries thus accelerating the user work. Obviously, also the underlying OPAC takes vantage from this situation as it appears more powerful than it really is.

The information populating the Dictionary are loaded at the ARCA system start up. Such information are extracted from a set of files whose content will be described in detail in the following of this report. The OPAC administrator is in charge of filling the files via a friendly interface. To report in the Dictionary any modification made on the files, the OPAC administrator has only to restart the Target. The files are:

1) *TargetFile*
   Containing general information about the target and about the capabilities of the underlying OPAC.

2) *DatbaseFile*
   Reporting information about the supported databases and the links between them and attribute sets, record syntaxes, element sets.

3) *TermListFile*
   Containing information about the term list supported by the databases and their own elements.

4) *AttributeSetFile*
   Including information about the supported attribute types and values, the allowable attribute combinations, the mapping between the Use values and the OPAC access points' tags, the rules according which the attribute combinations are translated in atomic queries processable by the OPAC.

5) *RecordSyntaxFile*
   Including information of the record syntax supported by the databases.

6) *ElementSetDetailsFile*
   Containing information about the presentation format available for each pair <database, record syntax> and their elements.

As it was exposed in [ARCA/T12/ADD], one of the enhancements to the SR protocol [ISO] offered by the ARCA Target is the support of the Z39.50 V3 protocol EXPLAIN service [ANSI]. Such a service is simulated by storing the necessary information in the Dictionary. Therefore, the Dictionary has been modelled according to the structure of the Explain database but neither it stores all the information concerning the Explain service nor it stores solely such information. As a matter of fact, in the analysis phase, we represented only the information we considered the most relevant for the functionality of the Target. Afterwards, we have augmented the Dictionary with a further set of information, useful for the Origin [ARCA/T22/ADD], which complete the presentation of both the Target and the databases. The supported Explain categories are the following:

1) *CategoryList*
   Lists the supported Explain categories.

2) *TargetInfo*
   Lists information about the connection with the target.

3) *DatabaseInfo*
   Lists information about the existing databases.

4) *TermListInfo*
   Reports information concerning the term lists supported.

5) *AttributeSetInfo*
   Includes information about the attributes for the databases.

6) *AttributeDetails*
   Includes for each database information about the supported attribute types and values and the supported attribute combinations.

7) *RecordSyntaxInfo*
   Lists information about the structure of the record in the databases.

8) *ElementSetDetails*
   Lists information about the available presentation formats.

As it can be seen later, some categories (2, 3, 5, 6, 7, 8) are important for the Target that uses their content in order to verify the service requests and to enhance the underlying OPAC capabilities while others (3, 4, 6, 7, 8) are needful for the user to understand the abstract structure of the available databases, to submit effective search queries and to choose a presentation format. Definitively, we believe that such a subset is quite complete.

Obviously, at the system start up, the information about the "virtual" Explain database must be loaded in the Dictionary by the Target. Therefore, the Dictionary will contain information about the real OPAC databases and about the Explain database that is implememted by the Dictionary itself.

Since we aimed to extend the services and the facilities of the underlying OPAC, we have to face the problem of distinguishing between the information about the OPAC basic services and those about the services available to an origin via ARCA. As a consequence, we have introduced some properties apart from the Explain database and, in case of need, we have modelled a property value as either not supported or OPAC natively supported or ARCA supported.

Reminding that we adopted the object oriented design methodology OMT [Rumbaugh et al.] for the Target development [ARCA/T12/ADD], we have modelled the Dictionary in terms of classes and associations between classes. This enables us to overcome the flat and redundat structure of the Explain records but , at the same time, it force us to navigate through the objects to retrieve some information. In Appendix A, the correspondence between Explain and Dictionary items is reported. Of course, being the Dictionary a collection of persistent read-only objects, only the Object Model is worthwhile to be presented.

For each class of the object model two different representations are given: a table showing the attributes with their types and the methods with the type of the returned values, and a C++ class skeleton produced by the CASE tool StP/OMT [StP/OMT]. Before going to describe the entities in the object model we introduce some necessary assumptions.

Firstly, we report in Appendix A all the types used to define the characteristics of the Dictionary objects except the C/C++ elementary types (int, void, char *). In some cases, we have used the data types defined in the YAZ toolkit library (which fully support the EXPLAIN facility). These data types are listed in Appendix B.

Secondly, although “Struct” and “Class” are equal in C++, we have used the “Class” mechanism to model the objects of the problem domain while the “Struct” clause has been adopted to describe particular sets of information returned by the methods, which contains data logically pertaining to various objects. In addition, we must outline that C++ allows to define a “class” type as the type of a characteristic or the type of a method returned value. We have used the "struct" clause also for the Attribute Type and Attribute Value objects because they are encapsulated in the associated Attribute Set objects.

Thirdly, we have choosen to implement each association in the Object Model as a pair of C++ pointers to the associated object in both the source and the target objects. Moreover, we remind that the charateristics of an object defined as “public” are visible by all the objects, the “private” ones are visible only inside the owner object while the “protected” ones are visible also from the derived objects.

Finally, we have missed the *constructors* and *destructors* methods since they are implicitly declared in the class definition. Among the input parameters of each method, we won’t include the type of the object the

method is acting on because it is implicit in the object-oriented design style. Anyway, the methods are fully specififed in the C++ classes.

For each class, the methods are grouped according five main categories:

- *Internal Methods*, including some special methods used only when loading the Dictionary files content (some of them could be protected or private);
- *Set Methods*, a set of simple methods used to set the value of the attributes of each created object;
- *Check Methods*, the methods invoked by the ARCA Target Core methods to play checks on the content of received APDUs' parameters;
- *Get Methods*, a set of methods used to return the value of the attributes of the created object or other less simple structures built starting from the attribute values;
- *Explain Methods*, the methods used to set and get information about the Explain database.

2.2 Class Dictionary

A “Dictionary” object has only one meaningfull attribute (Table 2.1 - Figure 2.1). The last modify date is used by the Origin to verify whether its own version of the Dictionary is consistent with the Target one. The other attributes are used when loading the Dictionary files content.

The methods implementing the objects creation reads data from the related file. It is worthwhile to remind that only one instance of the Dictionary and TargetInfo classes is created while the number of instances of the other classes depends upon the content of the files. In addition, after the objects have been created, some checks are performed to verify the correctness of each link.

The “GetExplainData” method returns information needed by the ARCA Target Core representing a Z39.50 V3 Explain category as defined by the YAZ toolkit [YAZ]. According to the requested category, the appropriate method of the related object is called. The first parameter of “GetExplainData” represents the requested Explain category, the second one the list of identifiers required for the retrieval, the third one the requested presentation format, the last one the buffer which the needed memory is kept from.

Attribute	Description
pvoCurrentObject	Pointer to a generic object
ptrTargetInfo	Pointer to the TargetInfo unique object
sllDatabaseList	List of pointers to DatabaseInfo objects
sllTermListList	List of pointers to TermList objects
sllAttributeSetList	List of pointers to AttributeSetInfo objects
sllElementSetList	List of pointers to ElementSetDetails objects
sllRecordSyntaxList	List of pointers to RecordSyntaxInfo objects
last_modify_date	The last modify date
Internal Method	
CreateRecordSyntaxObject	Creates a RecordSyntaxInfo object
CreateElementSetDetailsObject	Creates a ElementSetDetails object
CreateTermListObject	Creates a TermList object
CreateAttributeSetObject	Creates AttributeSetInfo, AttributeType, AttributeValue objects
CreateTargetObject	Creates the unique TargetInfo object
CreateDatabaseObject	Creates a DatabaseInfo object
CreateObjects	Creates the unique Dictionary object
CreateLinnks	Creates the links between the created objects
ResetNumberOfReferences	Resets the number of references to an object
Check Method	
CheckDatabaseExist	Checks if a database exists
CheckRecordSyntaxSupportedByTarget	Checks if the target supports a record syntax
CheckDatabaseAvailable	Checks if a database is available
CheckDatabaseIsExplain	Checks if a database is the Explain database
CheckQueryTypeSupported	Checks if a database supports a query type
CheckSearchableWith	Checks if two databases can be searched together
CheckAttributeSetSupported	Checks if an attribute set is supported by a database
CheckRecordSyntaxSupportedByDb	Checks if a record syntax is supported by a database
CheckAttributeTypeSupported	Checks if an attribute set supports an attribute type
CheckAttributeValueSupported	Checks if a type of an attribute set supports a value
CheckRPNOperatorSupported	Checks if adatabase supports a RPN operator
CheckRSetAsRPNOperandSupported	Checks if a database supports a result set name as a RPN query operand
CheckRPNSearchSupported	Checks if the OPAC supports queries in RPN structure
CheckElementSetsExist	Checks if some element set exists
CheckElementSetSupported	Checks if a database supports an element set according to a record syntax
CheckDefaultRecordSyntax	Checks if exists a default record syntax for the target
CheckAboutElementSets	Plays some checks on the element sets
Get Method	(used in Dictionary creation)
GetDatabase	Returns a pointer to a Database object with the given name
GetTermList	Returns a pointer to a Term List object with the given name
GetAttributeSet	Returns a pointer to a Attribute Set object with the given oid
GetRecordSyntax	Returns a pointer to a Record Syntax object with the given oid
GetElementSet	Returns a pointer to an Element Set object with the given name, database and record syntax
	(called by the Arca Target Core)
GetInitData	Returns the information about the Init service.
GetSearchCommonData	Returns the common information about the Search service.
GetDatabaseMaxRecSize	Returns the maximum record size of a database
GetNativeOperators	Returns the translation of RPN operators in the OPAC native query language
GetSupportedRecordSyntaxes	Returns as a string the oids of the supported record syntaxes for a database
GetDefaultDatabase	Returns the name of the database to be searched when the default Z39.50 database is specified
GetDefaultRecordSyntax	Returns the default record syntax for a list of databases
GetDefaultElementSetName	Returns the default format for a record syntax and a database
GetNativeForAttributesComb	Returns the native query corresponding to the attribute combination on the specified database and attribute set, otherwise returns an error message.
Explain Method	
AllocateExplainRecordList	Allocates memory for the Explain records
GetExplainData	Returns a list of records of the specified category formatted in the Explain record syntax and the specified element set name

Table 2.1 - The properties of the Dictionary class

```
#ifndef _Dictionary_h_
#define _Dictionary_h_

class Dictionary; /* Necessary for include files */

#include "KeyDescTable.h"
#include "Info.h"
#include "Message.h"
#include "TargetInfo.h"
#include "DatabaseInfo.h"
#include "TermList.h"
#include "AttributeSetInfo.h"
#include "ElementSetDetails.h"
#include "RecordSyntaxInfo.h"


class Dictionary
{
public:

  /* Methods */

 Dictionary(ArcMessage& log, arc_ErrorCode_T& errorCode);
 ~Dictionary(void);

   /* Internal Methods */

 arc_Result_T CreateRecordSyntaxObject      (ArcInfo& InfoFile, ArcMessage& fiLog,
                                             arc_ErrorCode_T& eError);
 arc_Result_T CreateElementSetDetailsObject (ArcInfo& InfoFile, ArcMessage& fiLog,
                                             arc_ErrorCode_T& eError);
 arc_Result_T CreateTermListObject          (ArcInfo& InfoFile, ArcMessage& fiLog,
                                             arc_ErrorCode_T& eError);
 arc_Result_T CreateAttributeSetObject      (ArcInfo& InfoFile, ArcMessage& fiLog,
                                             arc_ErrorCode_T&  eError);
 arc_Result_T CreateTargetObject            (ArcInfo& InfoFile, ArcMessage& fiLog,
                                             arc_ErrorCode_T& eError);
 arc_Result_T CreateDatabaseObject          (ArcInfo& InfoFile, ArcMessage& fiLog,
                                             arc_ErrorCode_T& eError);

   /* Check Methods */

 arc_Boolean_T        CheckDatabaseExist(char *pchDatabaseName);
 arc_Boolean_T        CheckRecordSyntaxSupportedBytarget(char *pchSyntax);
 arc_Boolean_T        CheckDatabaseAvailable(char *pchDatabaseName);
 arc_PrivateBoolean_T CheckDatabaseIsExplain(char *pchDatabaseName);
 arc_Boolean_T        CheckQueryTypeSupported(char *pchDatabaseName,int iQueryType);
 arc_Boolean_T        CheckSearchableWith(char *pchDBa, char *pchDBb);
 arc_Boolean_T        CheckAttributeSetSupported(char *pchDBName,
                                                 char *pchAttributeSetID);
 arc_Boolean_T        CheckRecordSyntaxSupportedByDb(char *pchDatabase,
                                                     char *pchRecordSyntaxSupportedID);
 arc_Boolean_T        CheckAttributeTypeSupported(char *pchAttributeSetID,
                                                  int iAttributeType);
 arc_Boolean_T        CheckAttributeValueSupported(char *pchAttributeSetID,
                                                   int iAttributeType,
                                                   int iAttributeValue);
 arc_Boolean_T        CheckRPNOperatorSupported(int iOperator, char *pchDBName);
 arc_PrivateBoolean_T CheckRSetAsRPNOperandSupported(char *pchDBName);
 arc_Boolean_T        CheckRPNSearchSupported();
 arc_Boolean_T        CheckElementSetsExist();
 arc_Boolean_T        CheckElementSetSupported(char *pchElementSetName,
                                               char *pchRecordSyntax,
                                               char *pchDatabaseName);
```

```
../* Get Methods (used in Dictionary creation) */

 DatabaseInfo *        GetDatabase(arc_String_T strDatabaseName);
 TermList *            GetTermList(arc_String_T strTermListName);
 AttributeSetInfo *    GetAttributeSet(arc_String_T strAttributeSetOid);
 RecordSyntaxInfo *    GetRecordSyntax(arc_String_T strRecordSyntaxOid);
 ElementSetDetails *   GetElementSet(arc_String_T strElementSetName,
                                     arc_String_T strRecordSyntax,
                                     arc_String_T strDatabaseName);

../* Get Methods */

 arc_InitData_T         * GetInitData(void);
 arc_SearchCommonData_T * GetSearchCommonData(void);
 int                      GetDatabaseMaxRecSize(char *pchDatabaseName);
 arc_NativeOperators_T *  GetNativeOperators();
 char *                   GetSupportedRecordSyntaxes(char *pchDatabase);
 char *                   GetDefaultDatabase();
 char *                   GetDefaultRecordSyntax(int iLen, char** ppchDBName);
 char *                   GetDefaultElementSetName(const char *pchDatabase,
                                                   const char *pchRecSyntax);
 char *                   GetNativeForAttributesComb(arc_Boolean_T* bIsSelector,
                                                     char *pchDBName,
                                                    arc_String_T strAttributeSetOid,
                                 Z_AttributesPlusTerm *prAttributesPlusTerm,
                                                     arc_MsgArea_T** pprMsgArea);

   /* Explain Method */
  Z_ExplainRecord **      AllocateExplainRecordList(int iNumRecsToAllocate,
                                                    arc_Buffer_T prEncode);
  Z_ExplainRecord **      GetExplainData(arc_String_T Category_Info_Type,
                                         arc_ExplainIdList_T sllIDSList,
                                         arc_String_T  strElementSetName,
                                         arc_Buffer_T prEncode);

protected:

   /* Structures used during Dictionary creation  */

 void*         pvoCurrentObject;    /* pointer to the current object*/
 TargetInfo*   ptrTargetInfo; /* pointer to the only target object*/
 arc_DatabaseList_T sllDatabaseList; /* pointer to database objects*/
 arc_TermListList_T sllTermListList; /* pointer to term list objects*/
 arc_AttributeSetList_T sllAttributeSetList;/*pointer to attribute set objects*/
 arc_ElementSetList_T sllElementSetList; /*pointer to attribute set objects*/
 arc_RecordSyntaxList_T sllRecordSyntaxList; /*pointer to record syntax objects */

 private:
   arc_String_T   last_modify_date;

   arc_Result_T   CreateObjects(const char* szFileName, arc_KeyId_T eClassType,
                                KeyDescTable& ArcaDictLang, ArcMessage& fiLog,
                                arc_ErrorCode_T& eError);
   arc_Result_T   CreateLinks(ArcMessage& fiLog, arc_ErrorCode_T& eError);
   void           ResetNumberOfReferences();
   arc_Result_T   CheckDefaultRecordSyntax(ArcMessage& fiLog,
                                           arc_ErrorCode_T& eError);
   arc_Result_T   CheckAboutElementSets(ArcMessage& fiLog, arc_ErrorCode_T& eError);
};
#endif
```

Figure 2.1 - The C++ skeleton for the Dictionary class

As it is shown in Figure 2.2, the unique Dictionary object can be seen as composed of items belonging to ten subclasses.

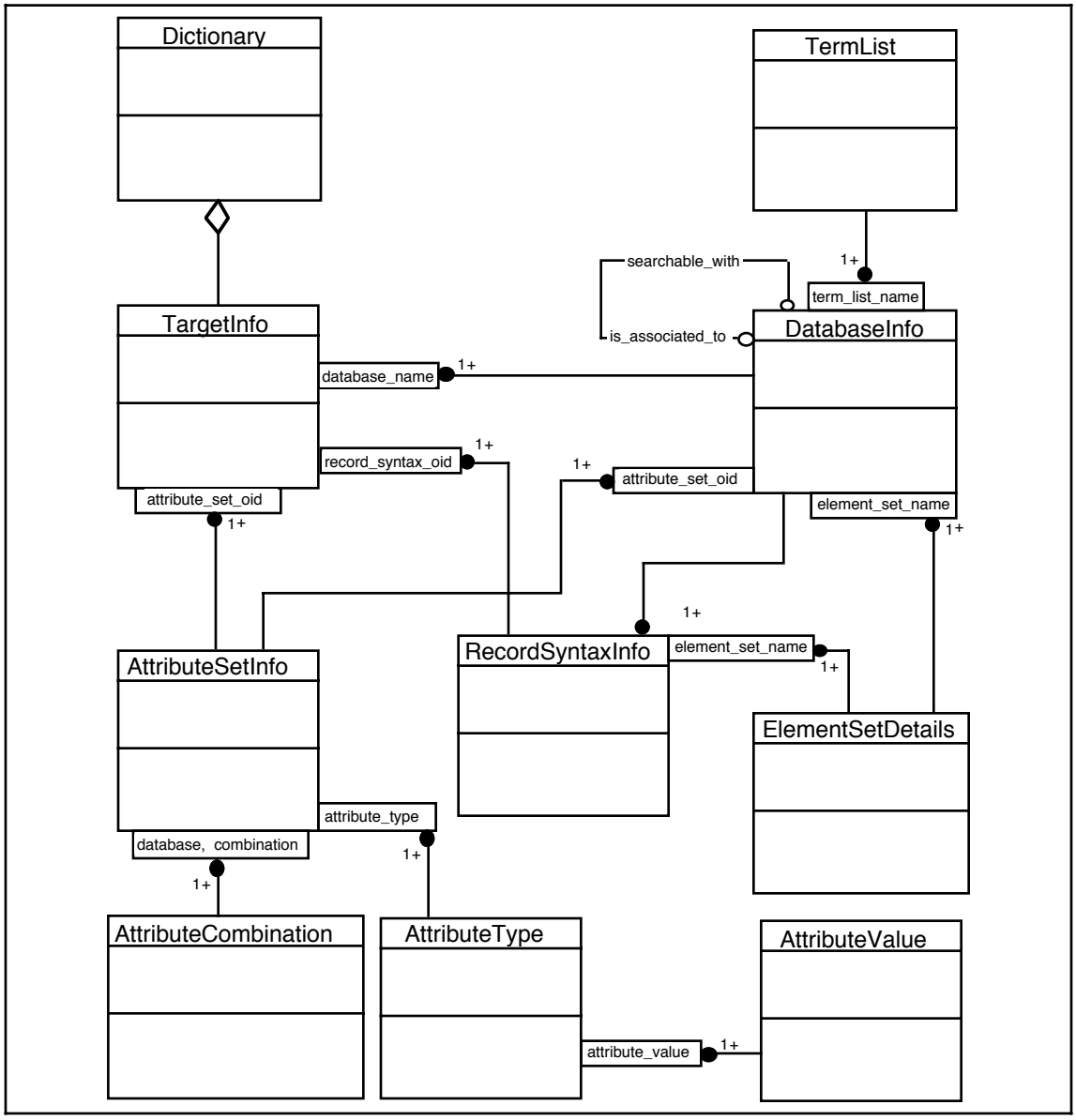


Figure 2.2 - The Dictionary context

2.3 Class TargetInfo

The class TargetInfo (Table 2.2 - Figure 2.3) is concerned with information about the whole Target. The information can be partitioned in two groups:

- A first group of items describing the Target to the user i.e. the items having type “arc_Text_T” or “arc_String_T” and the “diagnostic_set” property.

- A second group containing items about the facilities provided by the underlying OPAC natively or by means of ARCA procedures i.e. the items having type “arc_PrivateBoolean_T” or “selector_position” property which is used by the ARCA Target Core to put the incoming query selectors in the allowed position.

Attribute	Description
common_info	General information about each instance of this class.
target_name	Target name.
news	News about the Target.
welcome_message	A welcome message from the Target.
target_descr	Description of the Target system.
usage_restriction	Description of any usage restrictions applied to the target.
payment_address	Payment address for the supporting organization.
hours	Hours of operation.
result_set_naming	Wheher or not the result set naming is supported.
multi_database_search	Whether or not the search on multiple databases is supported.
max_result_set	Maximum number of result sets handled by the target.
max_num_term	Maximum number of terms allowed in a query
max_set_size	Maximum number of records in a result set
contact_info	Information about the person or institution to call.
preferred_message_size	The preferred target message size.
protocol_version	The target protocol version
diagnostic_set	Diagnostic set supported (BIB-1).
selector_position	The allowed selector position in a query.
supported_authentication	Whether or not the authentication is supported.
rpn_natively_supported	Whether or not rpn querys are nativeli supported
prNativeOperators	The OPAC native notation for AND, OR, AND_NOT, PROXIMITY operators
Internal Method	
CreateLinks	Creates the links between the target object and the other objects
GetDatabase	Returns the pointer to the database object with the specified name
Set Method	Sets the attribute values reading from file "TargetFile"
SetDateAdded, SetDateChanged, SetExpiry, SetHumanStringLanguage, SetTargetDescr, SetPaymentAddress, SetTargetName, SetNews, SetWelcomeMessage, SetTargetTxt, SetUsageRestriction, SetHours, SetMaxResultSet, SetMaxNumTerm, SetMaxSetSize, SetPreferredMessageSize, SetContactInfo, SetDiagnosticSet, SetSelectorPosition, SetResultSetNaming, SetMultiDatabaseSearch, SetSuppAuthentication, SetDatabases, SetRecordSyntaxes, SetRPNNativelySupported, SetRPNToStringConversion, SetNativeAnd, SetNativeOr, SetNativeAndNot, SetNativeProximity	
Check Method	
CheckDatabaseExist	Checks if a database exist
CheckDatabaseAvailable	Checks if a database is available
CheckDatabaseIsExplain	Checks if a database is the Explain database
CheckQueryTypeSupported	Checks if a database supports a query type
CheckSearchableWith	Checks if two databases are searchable together
CheckRecordSyntaxSupported	Checks if the target supports a record syntax
CheckRPNSearchSupported	Checks if the OPAC supports the RPN query (in the Z39.50 RPN notation) natively
Get Method	
GetInitData	Returns data used to fill Init Response
GetSearchCommonData	Returns data used to satisfy a search request
GetDatabaseMaxRecSize	Returns the maximum record size of a database
GetDefaultRecordSyntax	Returns the default record syntax if exists
GetNativeOperators	Returns the OPAC native version of RPN operators
Explain Method	
SetExplainDatabase	Sets the properties of the Explain database and the link with the target
GetAccessInfoData	Returns target data of the Explain AccessInfo structure
GetExplainData	Returns Explain data dealing with TargetInfo Category

Table 2.2 - The properties of the TargetInfo class

```
#ifndef _TargetInfo_h_
#define _TargetInfo_h_

class TargetInfo;

#include "Message.h"
#include "Info.h"
#include "Dictionary.h"
#include "DatabaseInfo.h"
#include "RecordSyntaxInfo.h"

class TargetInfo
{
  public:

   /* Methods */

   TargetInfo(ArcMessage& fiLog, arc_ErrorCode_T& eError);
   ~TargetInfo(void);
   arc_Result_T CreateLinks(Dictionary *const ptrDictionary,
                            ArcMessage& fiLog,
                            arc_ErrorCode_T& eError);

   /* Set Methods */

 void SetDateAdded               (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDateChanged             (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetExpiry                  (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetHumanStringLanguage     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetTargetDescr             (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetPaymentAddress          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetTargetName              (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetNews                    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetWelcomeMessage          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetTargetTxt               (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetUsageRestriction        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetHours                   (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetMaxResultSet            (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetMaxNumTerm              (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetMaxSetSize              (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetPreferredMessageSize    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetContactInfo             (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDiagnosticSet           (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetSelectorPosition        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetResultSetNaming         (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetMultiDatabaseSearch     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetSupportedAuthentication (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDatabases               (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRecordSyntaxes          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRPNNativelySupported    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRPNToStringConversion   (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetNativeAnd               (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetNativeOr                (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetNativeAndNot            (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetNativeProximity         (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);

   /* Check methods */

 arc_Boolean_T           CheckDatabaseExist(char *);
 arc_Boolean_T           CheckDatabaseAvailable(char *);
 arc_PrivateBoolean_T    CheckDatabaseIsExplain(char *);
 arc_Boolean_T           CheckQueryTypeSupported(char *, int);
 arc_Boolean_T           CheckSearchableWith(char *, char *);
 arc_Boolean_T           CheckRecordSyntaxSupported(char *);
 arc_Boolean_T           CheckRPNSearchSupported(){return(rpn_natively_supported);};
```

```
   /* Get methods */

 arc_InitData_T *        GetInitData(void);
 arc_SearchCommonData_T *GetSearchCommonData(void);
 int                     GetDatabaseMaxRecSize(char *);
 char *                  GetDefaultRecordSyntax(void);
 arc_NativeOperators_T * GetNativeOperators();

   /* Explain methods */

 void                    SetExplainDatabase();
 Z_AccessInfo *          GetAccessInfoData(arc_Buffer_T);
 Z_ExplainRecord *       GetExplainData(arc_String_T, arc_Buffer_T);

protected:
 arc_StringList_T             sllDatabaseNameLink;
 arc_StringList_T             sllRecordSyntaxIDLink;
 SLList <DatabaseInfo*>       sllDatabaseObjLink;
 arc_RecordSyntaxObjsList_T   sllRecordSyntaxObjLink;
 DatabaseInfo*                GetDatabase(arc_String_T strDatabaseName);

private:
 Z_CommonInfo        *common_info;
 arc_String_T         target_name;
 arc_Text_T           news;
 arc_Text_T           welcome_message;
 arc_Text_T           target_txt;
 arc_Text_T           target_descr;
 arc_Text_T           usage_restriction;
 arc_Text_T           payment_address;
 arc_String_T         hours;
 int                  max_result_set;
 int                  max_num_term;
 int                  max_set_size;
 arc_Text_T           contact_info;
 arc_Boolean_T        result_set_naming;
 arc_PrivateBoolean_T multi_database_search;
 int                  preferred_message_size;
 int                  selector_position;
 int                  protocol_version;
 arc_Boolean_T        rpn_natively_supported;
 arc_NativeOperators_T *prNativeOperators;
 arc_PrivateBoolean_T supported_authentication;
 arc_String_T         diagnostic_set;
};
#endif
```

Figure 2.3 - The C++ skeleton for the TargetInfo class

2.4 Class DatabaseInfo

The Target receives the requests directed to one or more databases of the underlying OPAC. The class "DatabaseInfo" holds information about each OPAC database and the virtual Explain database. Most of the attributes (Table 2.3 - Figure 2.4) are seen by the user to have as much as possible notices about the database he/she is going to search on. A particular relevance is assumed by the "RPN_query_syntax" and "query_type_supported" properties as they are used both by the Origin to make the user generates only admissible OPAC queries and by the Target to verify the incoming query.

Attribute	Description
common_info	General information about each instance of this class.
database_name	Full database name.
database_desc	Description of the database.
database_txt	An extended description of the database
explain_database	Whether this is an Explain database.
user_fee	Whether there is a fee to pay for searching.
available	Whether the database is available.
disclaimers	Any disclaimer concerning the database.
news	News about the database.
record_count	Record count for the database.
def_order	Default order in which records are presented.
average_rec_zize	Average size of the records.
max_rec_zize	Maximum size of the records.
hours	Hours of operation.
best_time	The best time to access the database.
last_update	The time of last update of the database.
update_interval	The update interval for this database.
coverage	The coverage dates of this database
proprietary	Whether the database contains proprietary information.
copyright_text	Description of copyright relating to this database.
copyright_notice	Notice about copyright to be displayed to the user if possible.
producer_contact_info	Description and contact info for the database producer.
supplier_contact_info	Description and contact info for the database supplier.
submission_contact_info	Description useful to submit material for the inclusion in the database.
title_string	The official long name of the database.
database_keywords	List of keywords for the database.
RPN_query_syntax	Allowed RPN query syntax.
query_types_supported	The query types supported for the database.
searchable_with	The list of databases the database is searchable with
attribute_sets	The list of attribute sets supported by the database
record_syntaxes	The record syntaxes supported by the database
element_set_details	The element set supported by the database
term_list	The term lists of the database
Set Method	Sets the attribute values reading from file "DatabaseFile"
SetDateAdded, SetDateChanged, SetExpiry, SetHumanStringLanguage, SetDatabaseDesc, SetDatabaseName, SetDatabaseTxt, SetDisclaimers, SetNews, SetDefOrder , SetHours, SetBestTime, SetLastUpdate, SetUpdateInterval, SetCoverage, SetCopyrightText, SetCopyrightNotice, SetProducerContactInfo, SetSupplierContactInfo, SetSubmissionContactInfo, SetTitleString, SetDatabaseKeywords, SetRPNQuerySyntax, SetAverageRecSize, SetMaxRecSize, SetRecordCount, SetExplainDatabase, SetUserFee, SetAvailable, SetProprietary, SetPrivateCapabilities, SetPrivateOperators, SetPrivateSearchKeys, SetRpnCapabilities, SetRpnOperators, SetResultSetAsOperandSupported, SetRestrictionOperandSupported, SetProximity, SetIso8777Capabilities, SetIso8777SearchKeys, SetSearchableWithDatabases, SetAttributeSet, SetTermList, SetElementDetails, SetRecordSyntaxes	
Check Method	
CheckAvailable	Checks if the database is available
CheckIsExplain	Checks if the database is the Explain database
CheckQueryTypeSupported	Checks if the database supports a query type
CheckSearchableWith	Checks if the database can be searched with another database
CheckElementSetSupported	Checks if the database supports an element set
CheckAttributeSetSupported	Checks if the database supports an attribute set
CheckRecordSyntaxSupported	Checks if the database supports a record syntax
CheckRPNOperatorSupported	Checks if a database supports a RPN operator
CheckRSetAsRPNOperandSupported	Checks if a database supports a result set name as a RPN operand
Get Method	
GetName	Returns the database name
GetRecordSyntax	Returns the record syntax OID with the given position in the list of the supported syntaxes
GetSupportedRecordSyntaxesAsSingleString	Returns all the supported syntaxes as a single string
GetSupportedRecordSyntaxesAsStringList	Returns all the supported syntaxes as a list of strings
GetDefaultElementSet	Returns the default element set for the database
GetMaxRecSize	Returns the database maximum record size
GetIsExplain	Returns the value specifyng if this is the Explain database
GetSearchableWithList	Returns the list of databases the database is searchable with

Explain Method	
SetExplainDbFields	Sets the simulated Explain (“IR-Explain-1”) database fields
GetPrivateQueryInfo	Returns data about the private query supported
GetRpnQueryInfo	Returns data about the RPN query supported
GetIso8777QueryInfo	Returns data about the ISO8777 query supported
GetQueryTypesInfoData	Returns data about the supported query types
GetAccessInfoData	Returns database data of the Explain AccessInfo structure
GetExplainData	Returns Explain data dealing with DatabaseInfo Category
GetTermListData	Returns Explain data dealing with TermList Category
GetAttributeDetailsData	Returns Explain data dealing with AttributeDetails category

Table 2.3 - The properties of the DatabaseInfo class

```
#ifndef _DatabaseInfo_h_
#define _DatabaseInfo_h_

class DatabaseInfo;

#include "Message.h"
#include "Info.h"
#include "Dictionary.h"
#include "TermList.h"
#include "AttributeSetInfo.h"
#include "ElementSetDetails.h"
#include "RecordSyntaxInfo.h"

class DatabaseInfo
{
public:
 DatabaseInfo(ArcMessage& fiLog, arc_ErrorCode_T& eError);
 ~DatabaseInfo();
 arc_Result_T CreateLinks(Dictionary *const ptrDictionary, ArcMessage& fiLog,
                          arc_ErrorCode_T& eError);
   /* Set Methods */
 void SetDateAdded          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDateChanged        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetExpiry             (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetHumanStringLanguage(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDatabaseName       (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDatabaseDesc       (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDatabaseTxt        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDisclaimers        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetNews               (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDefOrder           (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetHours              (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetBestTime           (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetLastUpdate         (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetUpdateInterval     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetCoverage           (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetCopyrightText      (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetCopyrightNotice    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetProducerContactInfo(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetSupplierContactInfo(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetSubmissionContactInfo(ArcInfo&,ArcMessage&,arc_ErrorCode_T&);
 void SetTitleString        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDatabaseKeywords   (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetPrivateOperators   (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRpnOperators       (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetResultSetAsOperandSupported(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRestrictionOperandSupported(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetProximity          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRPNQuerySyntax     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetAverageRecSize     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetMaxRecSize         (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetPrivateSearchKeys  (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRecordCount        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetExplainDatabase    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetUserFee            (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetAvailable          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetProprietary        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetIso8777SearchKeys  (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetPrivateCapabilities(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRpnCapabilities    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetIso8777Capabilities(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetSearchableWithDatabases(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetAttributeSets      (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetTermList           (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetElementDetails     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRecordSyntaxes     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
```

```
   /* Check Methods */
 arc_Boolean_T        CheckAvailable() { return(available);}
 int                  CheckIsExplain() {if(GetIsExplain()<2)return(1);return(0);}
 arc_Boolean_T        CheckQueryTypeSupported(int ) ;
 arc_Boolean_T        CheckSearchableWith(const char *) ;
 arc_Boolean_T        CheckElementSetSupported(const char *,const char *);
 arc_Boolean_T        CheckAttributeSetSupported(const char *);
 arc_Boolean_T        CheckRecordSyntaxSupported(const char *);
 arc_Boolean_T        CheckRPNOperatorSupported(int);
 arc_PrivateBoolean_T CheckRSetAsRPNOperandSupported();
   /* Get Methods */
 arc_String_T         GetName()  { return(database_name); }
 arc_String_T         GetRecordSyntax(int);
 arc_String_T         GetSupportedRecordSyntaxesAsSingleString();
 arc_StringList_T     GetSupportedRecordSyntaxesAsStringList();
 arc_String_T         GetDefaultElementSet(const char *);
 int                  GetMaxRecSize(){return(max_rec_size); }
 arc_PrivateBoolean_T GetIsExplain() {return(explain_database); }
   /* Explain Methods */
 void                    SetExplainDbFields(int);
 Z_DatabaseList  *       GetSearchableWithList(arc_Buffer_T);
 Z_PrivateCapabilities * GetPrivateQueryInfo(arc_Buffer_T, Pix);
 Z_RpnCapabilities     * GetRpnQueryInfo(arc_Buffer_T, Pix);
 Z_Iso8777Capabilities * GetIso8777QueryInfo(arc_Buffer_T, Pix);
 Z_QueryTypeDetails **   GetQueryTypesInfoData(arc_Buffer_T);
 Z_AccessInfo *          GetAccessInfoData(arc_Buffer_T);
 Z_ExplainRecord *       GetExplainData(arc_String_T, arc_Buffer_T);
 Z_ExplainRecord *       GetTermListData(arc_String_T,arc_Buffer_T);
 Z_ExplainRecord *       GetAttributeDetailsData(arc_String_T,arc_Buffer_T);
private:
 Z_CommonInfo     *common_info;
 arc_String_T     database_name;
 arc_String_T     database_desc;
 arc_Text_T       database_txt;
 arc_ArcaBoolean_T explain_database;
 arc_Boolean_T    user_fee;
 arc_Boolean_T    available;
 arc_Text_T       disclaimers;
 arc_Text_T       news;
 int              record_count;
 arc_String_T     def_order;
 int              average_rec_size;
 int              max_rec_size;
 arc_String_T     hours;
 arc_String_T     best_time;
 arc_String_T     last_update;
 arc_String_T     update_interval;
 arc_String_T     coverage;
 arc_Boolean_T    proprietary;
 arc_Text_T       copyright_text;
 arc_Text_T       copyright_notice;
 arc_Text_T       producer_contact_info;
 arc_Text_T       supplier_contact_info;
 arc_Text_T       submission_contact_info;
 arc_Text_T       title_string;
 arc_StringList_T database_keywords;
 arc_Text_T       RPN_query_syntax;
 arc_QueryTypeDetailsList_T query_types_supported;
 arc_StringList_T searchable_with;
 arc_StringList_T attribute_sets;
 arc_StringList_T element_set_details;
 arc_StringList_T record_syntaxes;
 arc_StringList_T term_list;
 arc_TermListObjsList_T sllTermListObjs;
 arc_AttributeSetObjsList_T sllAttributeSetObjs;
 arc_ElementSetObjsList_T sllElementSetObjs;
 arc_RecordSyntaxObjsList_T sllRecordSyntaxObjs;
};
#endif
```

Figure 2.4 - The C++ skeleton for the DatabaseInfo class

2.5 Class TermList

This class holds information a user can exploit to better formulate the queries. In fact, the search terms could be associated to attributes or attribute combinations with different search cost indexes (Table 2.5 - Figure 2.6). Other information belonging to this class is related to the Z39.50 V3 SCAN service. Even if the first release of the ARCA system will not support it, we decided to model the related information to enhance the system easily. So, the TermList objects will always have the "scannable" property set to "ARC_FALSE" and "narrower" and "broader" will be meaningless.

Attribute	Description
term_list_name	The name of the term list.
title	The title of the term list.
searchCost	An indication of how expensive it is to search using the associated attributes.
scannable	Whether thet term-list is scannable.
broader	A list of alternative, broader term lists.
narrower	A list of alternative, narrower term lists.
Set Method	Sets the attribute values reading from file "TermListFile"
SetTermListName, SetTitle, SetSearchCost, SetScannable, SetBroader , SetNarrower	
Get Method	
GetName	Returns the term list name
Explain Method	
SetExplainTermListElement	Sets a term list element for the TermListInfo Explain category
GetExplainData	Returns Explain data dealing with TermListInfo category

Table 2.4 - The properties of the TermList class

```
#define _TermList_h_
class TermList;
#include "DatabaseInfo.h"

class TermList
{
public:
TermList(ArcMessage& fiLog, arc_ErrorCode_T& eError);
~TermList(void);

   /* Set Methods */
    void SetTermListName (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
    void SetTitle        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
    void SetSearchCost   (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
    void SetScannable    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
    void SetBroader      (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
    void SetNarrower     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
    /* Get Methods */
    arc_String_T        GetName() { return(term_list_name); }
    /* Explain Methods */
    void                SetExplainTermListElement(char *);
    Z_TermListElement * GetExplainData(arc_Buffer_T);

protected:
    DatabaseInfo* ptrDatabaseInfo;
private:
    arc_String_T term_list_name;
    arc_String_T title;
    arc_String_T search_cost;
    arc_PrivateBoolean_T scanable;
    arc_StringList_T broader;
    arc_StringList_T narrower;
};
#endif
```

Figure 2.5 - The C++ skeleton for the TermList class

2.6 Classes about the attribute sets

A database recognizes one or more attribute sets (bib-1 is mandatory) each one composed by a specific collection of attributes which may be only a part of the entire attribute set (Table 2.6 - Figure 2.7) as well as each supported attribute type may supports a subset of the available values. Moreover, a database could support only some attribute combinations of all those could be secified with the attribute types and values supported.

The "AttributeSetInfo" class reports the pairs <type,value> of an attribue set allowed for a specific database and the association between the allowable attribute combinations in the search query and the OPAC access points [ARCA/T12/ABD]. Each pair <attribute type, attribute value> identifies an acces point plus a term (i.e. Title) and some properties to be applied in the search (i.e. truncation, word list). Therefore, we have built some special types to represent this situation. The information about each single attribute type is maintained in the "AttributeType" structure, those pertaining the allowed attribute values for each type in the AttributeType structure, the combinations supported by each database in the AttributeCombination structure.

Attribute	Description
common_info	General information about each instance of this class.
attribute_set_oid	Object identifier of the attribute set.
attribute_set_name	Name of the attribute set.
sllAttributeTypesList	The list of supported attribute types for the attribute set
sllAttributeCombinationsList	The list of supported attribute combination
	(used in AttributeType definition)
checked	If the attribute type has been checked (only for internal use)
attribute_type	Attribute type.
attribute_name	Attribute type name.
attribute_description	Attribute type description.
default_value	Attribute default value.
sllAttributeValuesList	The list of supported attribute values for the attribute type
	(used in AttributeValue definition)
checked	If the attribute type has been checked (only for internal use)
name	Attribute value name.
value	Attribute value.
description	Attribute value description.
is_selector	Whether this attribute value corresponds to a selector.
	(used in AttributeCombination definition)
iNumAttributes	The number of attribute types in the combination
database	The database supporting the combination
combination	The combination as a string (only for internal use)
translation	The associated translation rule
infix_word_list_operator	The native OPAC operator for the word list
infix_tags_operator	The native OPAC operator for the identification of the tag
sllUseAttributesList	The list of pairs <type, value> of the combination
Set Method	Sets the attribute values reading from file "AttributeSetFile"
SetDateAdded, SetDateChanged, SetExpiry, SetHumanStringLanguage, SetAttributeSetName, SetAttributeSetOid, SetAttributeCombinations , SetUseAttributes	
	(used for AttributeType)
SetAttributeTypes, SetAttributeType, SetAttributeName, SetAttributeDescr, SetDefaultValue	
	(used for AttributeValue)
SetAttributeDescriptions, SetName, SetValue, SetDescr, SetIsSelector	
	(used for AttributeCombination)
SetCombination, SetCombinationDatabaseName, SetTranslation, SetInfixWordListOperator, SetInfixTagsOperator , SetTags, SetUseValue	
Check Method	
CheckUseAttributeValueIsSelector	Checks if the attribute valus is a selector
CheckAttributeTypeSupported	Checks if a type is supported for a given attribute set
CheckAttributeValueSupported	Checks if a value is supported for the given type and set
CheckObjectDefinitionComplete	Checks if the definition of objects and structures is complete
Get Method	
GetOid	Returns the oid of the attribute set
GetNumCombinations	Returns the number of combinations supported by a database
GetNativeForAttributesComb	Returns the OPAC native version of the given combination
Explain Method	
SetExp1Attributes	Sets the Explain database supported types and combinations
GetAttributeSetDetails	Returns the data aboutthe Explain AttributeDetails category
GetAttributeSetCombinations	Returns the Explain data dealing with the combinations
GetExplainData	Returns the data dealing with AttributeSetInfo category

Table 2.6 - The properties of the AttributeSetInfo class

```
#ifndef _AttributeSetInfo_h_
#define _AttributeSetInfo_h_

class AttributeSetInfo;

#include "DatabaseInfo.h"
#include "Dictionary.h"

typedef struct arc_AttributeValue_T{
    arc_Boolean_T checked;
    arc_String_T name;
    int value;
    arc_String_T description;
    arc_Boolean_T is_selector;
};
typedef SLList <arc_AttributeValue_T*> arc_AttributeValuesList_T;

typedef struct arc_AttributeType_T{
    arc_Boolean_T checked;
    arc_String_T attribute_name;                        //
    int default_value;
    int attribute_type;                             //
    arc_String_T attribute_description;                 //
    arc_AttributeValuesList_T sllAttributeValuesList;
};

typedef SLList <arc_AttributeType_T*> arc_AttributeTypesList_T;

typedef struct arc_UseAttribute_T{
    int  use_value;
    arc_StringList_T tags;
};

typedef SLList <arc_UseAttribute_T*> arc_UseAttributesList_T;
typedef struct arc_AttributeCombination_T{
    int    iNumAttributes;
    arc_String_T database;
    arc_String_T combination;
    arc_String_T translation;
    arc_String_T infix_word_list_operator;
    arc_String_T infix_tags_operator;
    arc_UseAttributesList_T sllUseAttributesList;
};
typedef SLList <arc_AttributeCombination_T*> arc_AttributeCombinationsList_T;


class AttributeSetInfo
{
public:
   AttributeSetInfo(ArcMessage& fiLog, arc_ErrorCode_T& eError);
    ~AttributeSetInfo();
   arc_Result_T CreateLinks(Dictionary *const ptrDictionary,
                            ArcMessage& fiLog, arc_ErrorCode_T& eError);

   /* Set Methods */
 void SetDateAdded          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDateChanged        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetExpiry             (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetHumanStringLanguage(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetAttributeSetName   (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetAttributeSetOid    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetAttributeCombinations(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetUseAttributes      (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetCombination        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetCombinationDatabaseName(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetTranslation        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetInfixWordListOperator(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetInfixTagsOperator  (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetTags               (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetUseValue           (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
```

```
   //Used for AttributeType
 void SetAttributeTypes     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetAttributeType      (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetAttributeName      (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetAttributeDescr     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDefaultValue       (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);

   //Used for Attribute Description
 void SetAttributeDescriptions(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetName               (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetValue              (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDescr              (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetIsSelector         (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);

   /* Check Methods */
   arc_Boolean_T CheckUseAttributeValueIsSelector(int iAttValue);
   arc_Boolean_T CheckAttributeTypeSupported(int iAttributeType);
   arc_Boolean_T CheckAttributeValueSupported(int iAttributeType,
                                              int iAttributeValue);
   void  CheckObjectDefinitionComplete(ArcMessage&, arc_ErrorCode_T&);

   /* Get Methods */
   arc_String_T           GetOid() { return(attribute_set_oid); }
   int                    GetNumCombinations(arc_String_T strDBName);
   char *GetNativeForAttributesComb(arc_Boolean_T* bIsSelector, char *pchDBName,
                                    Z_AttributesPlusTerm *prAttPTerm,
                                    arc_MsgArea_T **pprMsgArea);

   /* Explain Methods */
   void                   SetExplAttributes      ();
   Z_ExplainRecord *      GetExplainData(arc_String_T,arc_Buffer_T);
   Z_AttributeSetDetails *GetAttributeSetDetails(arc_Buffer_T);
   void                   GetAttributeSetCombinations(arc_Buffer_T, arc_String_T,
                                                      Z_AttributeCombination ***,
                                                      int *);

protected:
    DatabaseInfo* ptrDatabaseInfo;
private:
  Z_CommonInfo *common_info;
  arc_String_T attribute_set_name;
  arc_String_T attribute_set_oid;
  arc_AttributeTypesList_T       sllAttributeTypesList;
  arc_AttributeCombinationsList_T sllAttributeCombinationsList;
  arc_Result_T  CheckAttributeCombination(const char *szAttrComb,
                                          const char* szExtAttrComb);
  arc_Boolean_T StringToInt(arc_String_T strX, int& iX);
  int           GetDefaultValue(int iAttributeType);
};

#endif
```

Figure 2.7 - The C++ skeleton for the AttributeSetInfo class

2.7 Class RecordSyntaxInfo

The database records are stored according to a specific record syntax (i.e. USMARC, UNIMARC). The property "record_structure" models the set of elements composing the record as a list of C++ structures implementing the Z39.50 V3 "ElementInfo" type (Table 2.7 - Figure 2.8).

Attribute	Description
common_info	General information about each instance of this class.
record_syntax_name	Name by which this syntax is known.
record_syntax_oid	Object identifier of the abstract record syntax.
record_syntax_descr	Description of the abstract record syntax.
ASN1_syntax	ASN.1 module describing the syntax.
is_default	Whether the record syntax is the default one for the target
Set Method	Sets the attribute values reading from file "RecordSyntaxFile"
SetDateAdded, SetDateChanged, SetExpiry, SetHumanStringLanguage, SetRecordSyntaxName, SetRecordSyntaxOid, SetRecordSyntaxDescr, SetASN1Syntax, SetIsDefault	
Get Method	
GetOid	Returns the oid of the record syntax
Check Method	
CheckIsDefault	Checks if the record syntax is the default one for the target
Explain Method	
SetExplainRecordSyntax	Sets the record syntaxes supported by the Explain databases (Explain, SUTRS)
GetExplainData	Returns data dealing with RecordSyntaxInfo category

Table 2.7 - The properties of the RecordSyntaxInfo class

```
#ifndef _RecordSyntaxInfo_h_
#define _RecordSyntaxInfo_h_

class RecordSyntaxInfo;

#include "Message.h"
#include "Info.h"      // The Set Methods need to read Info File
#include "Dictionary.h"
#include "ElementSetDetails.h"

class RecordSyntaxInfo
{
public:
RecordSyntaxInfo(ArcMessage&, arc_ErrorCode_T&);
~RecordSyntaxInfo();

   /* Set Methods */
 void SetDateAdded          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDateChanged        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetExpiry             (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetHumanStringLanguage(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRecordSyntaxName   (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRecordSyntaxOid    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRecordSyntaxDescr  (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetASN1Syntax         (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetIsDefault          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);

       /* Get Methods */
 arc_String_T         GetOid()      { return(record_syntax_oid);}
       /* Check Methods */
 arc_Boolean_T        CheckIsDefault()   { return(is_default); }

       /* Explain Methods */
 void SetExplainRecordSyntax    (const char *);
 Z_ExplainRecord *    GetExplainData(arc_String_T,arc_Buffer_T);

private:
    Z_CommonInfo *common_info;
    arc_Boolean_T is_default;
    arc_String_T record_syntax_name;
    arc_String_T record_syntax_oid;          // Key Element
    arc_String_T record_syntax_descr;
    arc_String_T ASN_1_syntax;
};
#endif
```

Figure 2.8 - The C++ skeleton for the RecordSyntaxInfo class

2.8 Class ElermentSetDetails

For each database a set of presentation format can be defined each one according a specified record syntax (Table 2.8 - Figure 2.9). The class "ElementSetDetails" contains information about the general properties of these element set and the list of elements composing the set which is reported in the "element_details" attribute.

Attribute	Description
common_info	General information about each instance of this class.
element_set_name	The element set name.
database_name	The associated database name
record_syntax_oid	The associated record syntax oid
element_set_descr	The description of the element set.
sllElementDetailsList	The elements composing the element set.
is_default	Whether the element set is the default one for the pair <database,record syntax>
Set Method	Sets the attribute values reading from file "ElementSetDetailsFile"
SetRecordSyntaxOid, SetDatabaseName, SetDateAdded, SetDateChanged, SetExpiry, SetHumanStringLanguage, SetElementSetName, SetElementSetDescr, SetPerElementDetails, SetElementName, SetRepeatable, SetRequired, SetIsDefault	
Check Method	
CheckRecordSyntaxOid	Checks if the record syntax is the one specified
CheckDatabaseName	Checks if the database is the one specified
CheckObjectDefinitionComplete	
CheckIsDefault	Checks if the element set is the default one
Get Method	
GetRecordSyntax	Returns the record syntax oid of the element set
GetDatabaseName	Returns the database name of the element set
GetName	Returns the name of the element set
GetKey	Returns the concatenation of database name, record syntax and element set name
Explain Method	
SetExplainElementSet	Sets the element sets ("B","F") for the Explain database and the record syntaxes Explain and SUTRS
GetExplainData	Returns Explain data dealing with ElementSetDetailsInfo Category

Table 2.8 - The properties of the ElementSetDetails class

```
#ifndef _ElementSetDetails_h_
#define _ElementSetDetails_h_

class ElementSetDetails;

#include "Message.h"
#include "Info.h"      // The Set Methods need to read Info File
#include "Dictionary.h"  /* for CreateLinks */

class ElementSetDetails
{
public:
ElementSetDetails(ArcMessage&, arc_ErrorCode_T&);
~ElementSetDetails();
arc_Result_T CreateLinks(Dictionary *const ptrDictionary,
                         ArcMessage& fiLog, arc_ErrorCode_T& eError);
void         ErrorKey(ArcMessage& fiLog)

       /* Set Methods */

 void SetDatabaseName       (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRecordSyntaxOid    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDateAdded          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetDateChanged        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetExpiry             (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetHumanStringLanguage(ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetElementSetName     (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetElementSetDescr    (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetPerElementDetails  (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetElementName        (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRepeatable         (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetRequired           (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);
 void SetIsDefault          (ArcInfo&, ArcMessage&, arc_ErrorCode_T&);

/* Check Methods */

   arc_Boolean_T CheckRecordSyntaxOid(const char *pchRecordSyntaxOid);
   arc_Boolean_T CheckDatabaseName(const char *pchDatabaseName);
   void          CheckObjectDefinitionComplete (ArcMessage&, arc_ErrorCode_T&);
   arc_Boolean_T CheckIsDefault() { return(is_default);}

   /* Get Methods */

   arc_String_T      GetRecordSyntax() { return(record_syntax_oid);}
   arc_String_T      GetDatabaseName() { return(database_name);}
   arc_String_T      GetName() { return(element_set_name); }
   arc_String_T      GetKey();

   /* Explain Methods */

   void              SetExplainElementSet(const char *, const char *);
   Z_ExplainRecord * GetExplainData(arc_String_T,arc_Buffer_T);

private:
   Z_CommonInfo                *common_info;
   arc_String_T                database_name;
   arc_String_T                record_syntax_oid;
   arc_String_T                element_set_name;
   arc_String_T                element_set_descr;
   arc_Boolean_T               is_default;
   arc_PerElementDetailsList_T sllElementDetailsList;
};
#endif
```

Figure 2.9 - The C++ skeleton for the ElementSetDetails class

3 Explain-Dictionary mapping

To better understand how the EXPLAIN service will be implented by using the Dictionary information, the detail matching between the items of the supported Explain catergories and the characteristics of the Dictionary objects is shown below. The used notation is shown in table 3.1.

Notation	Description
<object>.attribute	The value of the specified attribute of the specified object.
object1=object2-[(association_name)]->class	The object1 of the specified class obtained by following the association from the object2 towards the class. When is not clear which association to follow, the association name is reported.
derived	The value is the result of a computation.

Table 3.1 - The Explain-Dictionary mapping notation

We assume the existence of the generic objects: Dictionary, Target, Database, AttributeSet, RecordSyntax, ElementSet.

CategoryList item	Implementation in the Dictionary
categories	derived

Table 3.2 - The information returned by "Get_CategoryList_Data"

TargetInfo item	Implementation in the Dictionary
commonInfo	Target.common_info
name	Target.target_name
recentNews	Target.news
namedResultSets	Target.result_set_naming
multipleDBsearch	Target.multi_database_search
maxResultSets	Target.max_result_set
maxResultSize	Target.max_set_size
maxTerms	Target.max_num_term
welcomeMessage	Target.welcome_message
contactInfo	Target.contact_info
description	Target.target_descr
usageRest	Target.usage_restriction
paymentAddr	Target.payment_address
hours	Target.hours
dbCombinations	for each (db1=Target -> DatabaseInfo) for each (db2=db1-(searchable_with)->DatabaseInfo) <db1.database_name, db2.database_name>
accessInfo.diagnosticsSets	Target.diagnostic_set
accessinfo.recordSyntaxes	for each (rs=Target -> RecordSyntaxInfo) rs.record_syntax_oid

Table 3.3 - The information about the TargetInfo catergory

DatabaseInfo item	Implementation in the Dictionary
commonInfo	Database.common_info
name	Database.database_name
explainDatabase	Database.explain_database
userFee	Database.user_fee
available	Database.available
titleString	Database.title_string
keywords	Database.database_keywords
description	Database.database_descr
associatedDbs	for each (db=Database-(searchable_with)-> DatabaseInfo) db.database_name
disclaimers	Database.disclaimers
news	Database.news
recordCount	Database.record_count
defaultOrder	Database.def_order
avRecordSize	Database.average_rec_size
maxRecordSize	Database.max_rec_size
hours	Database.hours
bestTime	Database.best_time
lastUpdate	Database.last_update
updateInterval	Database.update_interval
coverage	Database.coverage
proprietary	Database.proprietary
copyrightText	Database.copyright_text
copyrightNotice	Database.copyright_notice
producerContactInfo	Database.producer_contact_info
supplierContactInfo	Database.supplier_contact_info
submissionContactInfo	Database.submission_contact_info
accessInfo.queryTypesSupported	Database.query_types_supported
accessInfo.attributeSetIds	Database.attribute_sets
accessInfo.recordSyntaxes	Database.record_syntaxes
accessInfo.elementSetNames	Database.element_set_details

Table 3.4 - The information about the DatbaseInfo category

RecordSyntaxInfo item	Implementation in the Dictionary
commonInfo	RecordSyntax.common_info
recordSyntax	RecordSyntax.record_syntax_oid
name	RecordSyntax.record_syntax_name
description	RecordSyntax.record_syntax_descr
asn1Module	RecordSyntax.ASN1_syntax

Table 3.6 - The information about The RecordSyntaxInfo category

ElementSetDetails item	Implementation in the Dictionary
commonInfo	ElementSet.common_info
databaseName	ElementSet.database_name
elementSetname	ElementSet.element_set_name
reordSyntax	ElementSet.record_syntax_oid
description	ElementSet.element_set_descr
detailsPerElement	ElementSet.sllElementDetailsList

Table 3.9 - The information about the ElementSetDetails category

AttributeSetInfo item	Implementation in the Dictionary
commonInfo	AttributeSet.common_info
attributeSet	AttributeSet.attribute_set_oid
name	AttributeSet.attribute_set_name
attributes	for each (at=AttributeSet -> sllAttributeTypesList) at.attribute_name at.attribute_type at.attribute_descr for each(av=at->sllAttributeValuesList) av.name av.description av.value

Table 3.7 - The information about the AttributeSetInfo category

AttributeDetails item	Implementation in the Dictionary
commonInfo	Database.common_info
databaseName	Database.database_name
attributesBySet	for each (at=Database->AttributeSet -> sllAttributeTypesList) at.attribute_name at.attribute_type at.attribute_descr for each(av=at->sllAttributeValuesList) av.name av.description av.value
attributeCombinations	for each (ac=AttributeSet -> sllAttributeCombinationsList) ac.sllUseAttributesList

Table 3.7 - The information about the AttributeDetails categoru

TermListInfo item	Implementation in the Dictionary
databaseName	Database.database_name
termLists	for each (tl=Database->TemList) tl.tem_list_name tl.title tl.search_cost tl.scannable tl.broader tl.narrower

Table 3.8 - The information about the TermListInfo category

4 Creating the Dictionary

As we have previously argued the OPAC administrator should put in a set of files those information populating the Dictionary before the ARCA Target starting up. For those properties the Explain service deals with, the problem of distinguishing between the OPAC and the Target values arises.

For such reason, we have adopted a particular syntax that foresees the use of the "value" and, in case of need, "mode" clauses in the description of a property.

The information is partitioned in a set of file each one containing the information related to each supported Explain category.

The files are loaded by using a friendly interface implemented in Tcl/Tk.

4.1 File "TargetFile"

```
TargetInfo:
common_info:
  dateAdded:
    value=String_Value;
  dateChanged:
    value=String_Value;
  expiry:
    value=String_Value;
  humanStringLanguage:
    value=String_Value;
target_name:                          // Mandatory
  value=ARCA;
news:
  value=Text_Value;
welcome_message:
  value=Text_Value;
target_descr:
  value=Text_Value;
usage_restriction:
  value=Text_Value;
payment_address:
  value=Text_Value;
hours:
  value=String_Value;
result_set_naming:                    // Mandatory
  value=(Y|N) ;
multi_database_search:                // Mandatory
  support :
      value=(Y|N) ;
  mode:
    value=(ARCA|native);
max_result_set:                       // Mandatory
  value=Int_Value;
max_num_term:                         // Mandatory
  value=Int_Value;
max_set_size:                         // Mandatory
  value=Int_Value;
preferred_message_size:               // Mandatory
  value=Int_Value;
```

```
contact_info:
  value=Text_Value;
diagnostic_set:
  value=(1.2.840.10003.4.1);
selector_position:                    // Mandatory
  value=(RIGHT|LEFT|ANY);
supported_authentication:             // Mandatory
  support:
         value=(Y|N);
  mode:
         value=(ARCA|native);
databases:                            // Mandatory
  value=String_Value,
.....................,
.....................,
  value=String_Value.
record_syntaxes:                      // Mandatory
  value=Oid_Value,
.....................,
.....................,
  value=Oid_Value.
```

4.2 File "DatabaseFile"

```
// The following description is repeated for each database supported
DatabaseInfo:                         // Mandatory
common_info:
  dateAdded:
    value=String_Value;
  dateChanged:
    value=String_Value;
  expiry:
    value=String_Value;
  humanStringLanguage:
    value=String_Value;
database_name:                        // Mandatory
  value=String_Value;
database_descr:
  value=Text_Value;
explain_database:
  support:
       value=(Y|N),
  mode:
       value=(ARCA|native);
user_fee:                             // Mandatory
  value=(Y|N);
available:                            // Mandatory
  value=(Y|N);
disclaimers:
  value=Text_Value;
news:
  value=Text_Value;
record_count:
  value=Int_Value;
def_order:
  value=Text_Value;
average_rec_size:
  value=Int_Value;
max_rec_size:
  value=Int_Value;
hours:
  value=String_Value;
```

```
best_time:
  value=String_Value;
last_update:
  value=String_Value;
update_interval:
  value=String_Value;
coverage:
  value=String_Value;
proprietary;
  value=(Y|N);
copyright_text;
  value=Text_Value;
copyright_notice:
  value=Text_Value;
producer_contact_info:
  value=Text_Value;
supplier_contact_info:
  value=Text_Value;
submission_contact_info:
  value=Text_Value;
title_string:
  value=Text_Value;
database_keywords:
  value=String_Value,
.....................,
.....................,
  value=String_Value.
RPN_query_Syntax:
  value=File;
query_types_supported:           // list of the query type
     private_capabilities:
       private_operators:
         operator:               //list of operators and descr
           value=String_Value;
         description:
           value=String_value;
.....................;
.....................;
         operator:
           value=String_Value;
         description:
           value=String_value.
       private_search_keys:
         search:                 //list of search keys and descr
           value=String_Value;
         description:
           value=String_value;
.....................;
.....................;
         search:
           value=String_Value;
         description:
           value=String_value.
     Rpn_capabilities:
       operators:
         value=Int_Value,
         ..............,
         value=Int_Value.
       result_set_as_operand_supported:
         support:
           value=(Y|N),
         mode:
           value=(ARCA|native);
```

```
       restriction_operand_supported:
         value=N;
       proximity:
         value=(Y|N),
         mode=(native)
     iso8777_capabilities:
       search_keys:              //list of search keys and descr
         search:
           value=String_Value;
         description:
           value=String_value;
.....................;
.....................;
         search:
           value=String_Value;
         description:
attribute_sets:                        // Mandatory
  value=Oid_Value,
  ..............,
  value=Oid_Value.
term_lists:                            // Mandatory
  value=String_Value,
  ..............,
  value=String_Value.
element_set_details:                   // Mandatory
  value=String_Value,
  ..............,
  value=String_Value.
searchable_with_databases:
  value=String_Value,
  ..............,
  value=String_Value.
record_syntaxes:                       // Mandatory
  value=Oid_Value,
  ..............,
  value=Oid_Value.
```

4.3 File "TermListFile"

```
// The following description is repeated for each tem list supported
TermList:
term_list_name:                        // Mandatory
  value=String_Value;
title:
  value=String_Value;
search_cost:
  value=(optimized|normal|expensive|filter);
scannable:                             // Mandatory
  support:
       value=(Y|N),
  mode:
       value=(ARCA|native);
broader:
  value=String_Value,
  ..............,
  value=String_Value.
narrower:
  value=String_Value,
  ..............,
  value=String_Value.
```

4.4 File "AttributeSetFile"

```
// The following description is repeated for each
// attribute set supported

AttributeSetInfo:
common_info:
  dateAdded:
    value=String_Value;
  dateChanged:
    value=String_Value;
  expiry:
    value=String_Value;
  humanStringLanguage:
    value=String_Value;
attribute_set_oid:                    // Mandatory
  value=Oid_Value;
attribute_set_name:
  value=String_Value;
// The following description is repeated for each
// attribute combination
attribute_combinations:          //begin repeatable item
  infix_word_list_operator:
      value=(String_Value, native);
infix_tags_operator:
  value=(String_Value, native);
combination:
  value=(attribute_type_Int_Value, attribute_value_Int_Value) ...
         (attribute_type_Int_Value, attribute_value_Int_Value);
translation:
  value=String_Value;
usable_with_database:
  value=String_Value;
use_attributes:
  use_value:
          value=Int_Value;
  tags:
          value=String_Value,
          ...................
          value=String_Value.
  ........................
  use_value:
          value=Int_Value;
  tags:
          value=String_Value,
          ...................
          value=String_Value.        //end repeatable item
.........................
attribute_combinations:              //begin repeatable item
.........................
.........................            //end repeatable item
attribute_types:
  attribute_type:                    //begin repeatable item
        value=Int_Value;
  attribute_name:
        value=String_Value;
  attribute_descr:
        value=String_Value;
  default_value:
        value=Int_Value;
  attribute_values:                  //begin repeatable item
        name:
          value=String_Value;
```

```
          value:
            value=Int_Value;
          is_selector:
            value=(Y|N);              //end repeatable item
   ......................
   attribute_values:                  //begin repeatable item
          name:
            value=String_Value;
          value:
            value=Int_Value;
          is_selector:
            value=(Y|N);              //end repeatable items
.........................
attribute_types:
   attribute_type:                    //begin repeatable item
          value=Int_Value;
   attribute_name:
          value=String_Value;
   attribute_descr:
          value=String_Value;
   default_value:
          value=Int_Value;
   attribute_values:                  //begin repeatable item
          name:
            value=String_Value;
          value:
            value=Int_Value;
          is_selector:
            value=(Y|N);              //end repeatable item
   ......................
   attribute_values:                  //begin repeatable item
          name:
            value=String_Value;
          value:
            value=Int_Value;
          is_selector:
            value=(Y|N);              //end repeatable items
```

4.5 File “RecordSyntaxFile”

```
// The following description is repeated for each record syntax
// supported
RecordSyntaxInfo:
common_info:
   dateAdded:
     value=String_Value;
   dateChanged:
     value=String_Value;
   expiry:
     value=String_Value;
   humanStringLanguage:
     value=String_Value;
recordsyntax_name:
   value=String_Value;
recordsyntax_oid:                     // Mandatory
   value=Oid_Value;
recordsyntax_descr:
   value=Text_Value;
is_default:
   value=(Y|N);
ASN_1_syntax:
   value=Text_Value;
```

4.6 File "ElementSetDetailsFile"

```
// The following description is repeated for each element set
// supported
ElementSetDetails:                    // Mandatory
common_info:
  dateAdded:
    value=String_Value;
  dateChanged:
    value=String_Value;
  expiry:
    value=String_Value;
  humanStringLanguage:
    value=String_Value;
element_set_name:                     // Mandatory
  value=String_Value;
database_name:                        // Mandatory
  value=String_Value;
record_syntax_oid:                    // Mandatory
  value=Oid_Value;
elementset_descr:
    value=Text_Value;
is_default:
  value=(Y|N);
element_set_details:
    perElementDetails:          //begin repeatable item
perElementDetails
      name:
        value=String_Value;
      repeatable:
        value=(Y|N);
      required:
        value=(Y|N);  //end rpeatable item perElementDetails
    perElementDetails:          //begin rpeatable item
    ................;          // end rpeatable item
```

5 References

[ANSI] *Information Retrieval: Application Service Definition and Protocol Specification*, ANSI/NISO Z39.50-1995, April 1995.

[ARCA/T12/ABD] *Analysis of BIB-1 Mapping*, Report ARCA/T12/ABD, September 1995

[ARCA/T12/ADD] *ARCA Architectural Design Document*, Report ARCA/T12/ADD, November 1995.

[ARCA/T22/ADD] *User Interface Application Design Document*, Report ARCA/T22/ADD, November 1995

[ARCA/T13/DDD] *ARCA Detailed Design Document*, Report ARCA/T13/DDD, January 1996

[ISO] International Organization for Standardization (ISO): *Information and Documentation — Open Systems Interconnection —Search and Retrieve Application Service Definition*, International Standard ISO 10162 (1993)

[Rumbaugh et al.] Rumbaugh J., Blaha M., Premerlani W., Eddy F. and Lorensen W.: *Object-Oriented Modelling and Design*, Prentice-Hall International, 1991.

[StP/OMT] *Object Modelling Technique. Creating OMT Models*, Software through Pictures Release 2, IDE (Interactive Development Environment)

[YAZ] *YAZ User's Guide and Reference*, Index Data

6 Definitions and acronyms

ANSI	American National Standards Institute
APDU	Application Protocol Data Unit
ARCA	Access to Remore Catalogues
ASN.1	Abstract Syntax Notation One
CASE	Computer Aided Software Engineering
ISO	International Organization for Standardization
OMT	Object Modelling Technique
OPAC	Online Public Access Catalogue
RPN	Reverse Polish Notation
SR	Search and Retrieve
StP/OMT	Software through Pictures/OMT
UNIMARC	Universal MARC
USMARC	United States MARC
YAZ	Yet Another Z39.50 Toolkit

Appendix A: Dictionary data types

This appendix reports the structure of the data types used in the detailed design of the Dictionary.

```
arc_String_T                Alphanumeric string
arc_Text_T                  Free text
arc_Boolean_T               enum{   ARC_FALSE,
                                    ARC_TRUE}
arc_PrivateBoolean_T        enum{   ARC_TRUE_ARCA,
                                    ARC_TRUE_NATIVE,
                                    ARC_FALSE}
arc_Result_T                enum{   ARC_FAILURE,
                                    ARC_SUCCESS }
ArcInfo&                    Reference to a file including the information to load
ArcMessage&                 Refercence to the log File
arc_ErrorCode_T&            Refererence to possible error found when reading files
content
arc_StringList_T            List of arc_String_T typed elements
arc_ExplainId_T             struct {
                                    int iExp1UseValue;
                                    arc_String_T strID;}
arc_ExplainIdList_T         List of arc_ExplainId_T typed elements
arc_InitData_T              struct {
                                    int             iPrefMsgSize;
                                    int             iExcRecSize;
                                    int             iProtocolVersion;
                                    arc_Boolean_T   eResultSetNaming;
                                    arc_PrivateBoolean_T
                            eSupportedAuthentication;}
arc_SearchCommonData_T      struct {
                                    arc_PrivateBoolean_T
                            eMultipleDatabaseSearch;
                                    int             iMaxResultSets;
                                    int             iMaxNumTerms;
                                    int             iMaxSetSize;
                                    int             iSelectorPosition;
                                    int             iMaxNumDatabases;}
arc_NonRPNType_T            enum{                   ARC_INFIX,
                                ARC_ISO8777,
                                ARC_PRIVATE,
                                ARC_RPNSTRING}
arc_NativeOperators_T       struct {
                                    arc_String_T    strNativeAND;
                                    arc_String_T    strNativeOR;
                                    arc_String_T    strNativeANDNOT;
                                    arc_String_T    strNativePROX;
                                    arc_NonRPNType_T    eConvertType;}

arc_DatabaseObjsList_T      List of pointers to Database objects
arc_TermListObjsList_T      List of pointers to TermList objects
arc_AttributeSetObjsList_T  List of pointers to AttributeSetInfo objects
arc_PerElementDetailsList_T List of pointers to Z_PerElementDetails typed elements
arc_ElementSetObjsList_T    List of pointers to ElementSetdetails objects
arc_RecordSyntaxObjsList_T  List of pointers to RecordSyntaxInfo objects
arc_QueryTypeDetailsList_T  List of pointers to Z_QueryTypeDetails typed elements
```

```
TermListReference           struct {
                                TermList*       ptrTermListObj;
                                int             iNumberOfReference;}
arc_TermListList_T          List of pointers to TermListReference typed elements
AttributeSetReference       struct {
                                AttributeSetInfo* ptrAttributeSetObj;
                                int             iNumberOfReference;}
arc_AttributeSetList_T      List of pointers to AttributeSetReference typed elements
ElementSetReference         struct {
                                ElementSetDetails*      ptrElementSetObj;
                                int             iNumberOfReference;}
arc_ElementSetList_T        List of pointers to ElementSetReference typed elements
RecordSyntaxReference       struct {
                                RecordSyntaxInfo*       ptrRecordSyntaxObj;
                                int             iNumberOfReference;}
arc_RecordSyntaxList_T      List of pointers to RecordSyntaxReference typed elements
DatabaseReference           struct {
                                DatabaseInfo*   ptrDatabaseObj;
                                int             iNumberOfReference;}
arc_DatabaseList_T          List of pointers to DatabaseReference typed elements
```

Appendix B: YAZ data types

This appendix reports the structure of the YAZ toolkit library data types used in the detailed design of the Dictionary.

```
#ifndef PRT_EXP_H
#define PRT_EXP_H

#include <yconfig.h>

#define multipleDbSearch multipleDBsearch


typedef struct Z_CommonInfo
{
    char *dateAdded;           /* OPTIONAL */
    char *dateChanged;         /* OPTIONAL */
    char *expiry;              /* OPTIONAL */
    char *humanStringLanguage;    /* OPTIONAL */
    Z_OtherInformation *otherInfo;          /* OPTIONAL */
} Z_CommonInfo;

typedef struct Z_HumanStringUnit
{
    char *language;               /* OPTIONAL */
    char *text;
} Z_HumanStringUnit;

typedef struct Z_HumanString
{
    int num_strings;
    Z_HumanStringUnit **strings;
} Z_HumanString;

typedef struct Z_CategoryInfo
{
    char *category;
    char *originalCategory;                 /* OPTIONAL */
    Z_HumanString *description;             /* OPTIONAL */
    char *asn1Module;                       /* OPTIONAL */
} Z_CategoryInfo;


typedef struct Z_IconObjectUnit
{
    int which;
#define Z_IconObject_ianaType 0
#define Z_IconObject_z3950type 1
#define Z_IconObject_otherType 2
    char *bodyType;
    Odr_oct *content;
} Z_IconObjectUnit;

typedef struct Z_IconObject
{
    int num_iconUnits;
    Z_IconObjectUnit **iconUnits;
} Z_IconObject;
```

```
typedef struct Z_ContactInfo
{
    char *name;                                 /* OPTIONAL */
    Z_HumanString *description;                 /* OPTIONAL */
    Z_HumanString *address;                     /* OPTIONAL */
    char *email;                                /* OPTIONAL */
    char *phone;                                /* OPTIONAL */
} Z_ContactInfo;

typedef struct Z_NetworkAddressIA
{
    char *hostAddress;
    int *port;
} Z_NetworkAddressIA;

typedef struct Z_NetworkAddressOPA
{
    char *pSel;
    char *sSel;                    /* OPTIONAL */
    char *tSel;                    /* OPTIONAL */
    char *nSap;
} Z_NetworkAddressOPA;

typedef struct Z_NetworkAddressOther
{
    char *type;
    char *address;
} Z_NetworkAddressOther;

typedef struct Z_NetworkAddress
{
    int which;
#define Z_NetworkAddress_iA 0
#define Z_NetworkAddress_oPA 1
#define Z_NetworkAddress_other 2
    union
    {
       Z_NetworkAddressIA *internetAddress;
  Z_NetworkAddressOPA *osiPresentationAddress;
  Z_NetworkAddressOther *other;
    } u;
} Z_NetworkAddress;

typedef struct Z_PrivateCapOperator
{
    char *roperator;
    Z_HumanString *description;                 /* OPTIONAL */
} Z_PrivateCapOperator;

typedef struct Z_SearchKey
{
    char *searchKey;
    Z_HumanString *description;                 /* OPTIONAL */
} Z_SearchKey;

typedef struct Z_PrivateCapabilities
{
    int num_operators;
    Z_PrivateCapOperator **operators;          /* OPTIONAL */
    int num_searchKeys;
    Z_SearchKey **searchKeys;                   /* OPTIONAL */
    int num_description;
    Z_HumanString **description;                /* OPTIONAL */
```

```
} Z_PrivateCapabilities;

typedef struct Z_ProxSupportPrivate
{
    int *unit;
    Z_HumanString *description;             /* OPTIONAL */
} Z_ProxSupportPrivate;

typedef struct Z_ProxSupportUnit
{
    int which;
#define Z_ProxSupportUnit_known 0
#define Z_ProxSupportUnit_private 1
    union
    {
       int known;
  Z_ProxSupportPrivate *private;
    } u;
} Z_ProxSupportUnit;

typedef struct Z_ProximitySupport
{
    bool_t *anySupport;
    int num_unitsSupported;
    Z_ProxSupportUnit **unitsSupported;     /* OPTIONAL */
} Z_ProximitySupport;

typedef struct Z_RpnCapabilities
{
    int num_operators;
    int **operators;                        /* OPTIONAL */
    bool_t *resultSetAsOperandSupported;
    bool_t *restrictionOperandSupported;
    Z_ProximitySupport *proximity;          /* OPTIONAL */
} Z_RpnCapabilities;

typedef struct Z_Iso8777Capabilities
{
    int num_searchKeys;
    Z_SearchKey **searchKeys;
    Z_HumanString *restrictions;            /* OPTIONAL */
} Z_Iso8777Capabilities;

typedef struct Z_QueryTypeDetails
{
    int which;
#define Z_QueryTypeDetails_private 0
#define Z_QueryTypeDetails_rpn 1
#define Z_QueryTypeDetails_iso8777 2
#define Z_QueryTypeDetails_z3958 3
#define Z_QueryTypeDetails_erpn 4
#define Z_QueryTypeDetails_rankedList 5
    union
    {
  Z_PrivateCapabilities *private;
  Z_RpnCapabilities *rpn;
  Z_Iso8777Capabilities *iso8777;
  Z_HumanString *z3958;
  Z_RpnCapabilities *erpn;
  Z_HumanString *rankedList;
    } u;
} Z_QueryTypeDetails;

typedef struct Z_AccessRestrictionsUnit
```

```
{
    int *accessType;
#define Z_AccessRestrictions_any                 0
#define Z_AccessRestrictions_search              1
#define Z_AccessRestrictions_present             2
#define Z_AccessRestrictions_specific_elements   3
#define Z_AccessRestrictions_extended_services   4
#define Z_AccessRestrictions_by_database         5
    Z_HumanString *accessText;              /* OPTIONAL */
    int num_accessChallenges;
    Odr_oid **accessChallenges;             /* OPTIONAL */
} Z_AccessRestrictionsUnit;

typedef struct Z_AccessRestrictions
{
    int num_restrictions;
    Z_AccessRestrictionsUnit **restrictions;
} Z_AccessRestrictions;

typedef struct Z_Charge
{
    Z_IntUnit *cost;
    Z_Unit *perWhat;                        /* OPTIONAL */
    Z_HumanString *text;                    /* OPTIONAL */
} Z_Charge;

typedef struct Z_CostsOtherCharge
{
    Z_HumanString *forWhat;
    Z_Charge *charge;
} Z_CostsOtherCharge;

typedef struct Z_Costs
{
    Z_Charge *connectCharge;                /* OPTIONAL */
    Z_Charge *connectTime;                  /* OPTIONAL */
    Z_Charge *displayCharge;                /* OPTIONAL */
    Z_Charge *searchCharge;                 /* OPTIONAL */
    Z_Charge *subscriptCharge;              /* OPTIONAL */
    int num_otherCharges;
    Z_CostsOtherCharge **otherCharges;      /* OPTIONAL */
} Z_Costs;

typedef struct Z_AccessInfo
{
    int num_queryTypesSupported;
    Z_QueryTypeDetails **queryTypesSupported;  /* OPTIONAL */
    int num_diagnosticsSets;
    Odr_oid **diagnosticsSets;              /* OPTIONAL */
    int num_attributeSetIds;
    Odr_oid **attributeSetIds;     /* OPTIONAL */
       int num_schemas;
    Odr_oid **schemas;                      /* OPTIONAL */
       int num_recordSyntaxes;
    Odr_oid **recordSyntaxes;               /* OPTIONAL */
    int num_resourceChallenges;
    Odr_oid **resourceChallenges;           /* OPTIONAL */
    Z_AccessRestrictions *restrictedAccess;  /* OPTIONAL */
    Z_Costs *costInfo;                      /* OPTIONAL */
    int num_variantSets;
    Odr_oid **variantSets;                  /* OPTIONAL */
    int num_elementSetNames;
    char **elementSetNames;     /* OPTIONAL */
    int num_unitSystems;
```

```
    char **unitSystems;                     /* OPTIONAL */
} Z_AccessInfo;

typedef struct Z_DatabaseList
{
    int num_databases;
    Z_DatabaseName **databases;
} Z_DatabaseList;

typedef struct Z_AttributeValueList
{
    int num_attributes;
    Z_StringOrNumeric **attributes;
} Z_AttributeValueList;

typedef struct Z_AttributeOccurrence
{
    Odr_oid *attributeSet;         /* OPTIONAL */
    int *attributeType;
    Odr_null *mustBeSupplied;               /* OPTIONAL */
    int which;
#define Z_AttributeOcc_anyOrNone 0
#define Z_AttributeOcc_specific 1
    union
    {
   Odr_null *anyOrNone;
   Z_AttributeValueList *specific;
    } *attributeValues;
} Z_AttributeOccurrence;

typedef struct Z_AttributeCombination
{
    int num_occurrences;
    Z_AttributeOccurrence **occurrences;
} Z_AttributeCombination;

typedef struct Z_AttributeCombinations
{
    Odr_oid *defaultAttributeSet;
    int num_legalCombinations;
    Z_AttributeCombination **legalCombinations;
} Z_AttributeCombinations;

typedef struct Z_AttributeValue
{
    Z_StringOrNumeric *value;
    Z_HumanString *description;             /* OPTIONAL */
    int num_subAttributes;
    Z_StringOrNumeric **subAttributes;      /* OPTIONAL */
    int num_superAttributes;
    Z_StringOrNumeric **superAttributes;    /* OPTIONAL */
    Odr_null *partialSupport;               /* OPTIONAL */
} Z_AttributeValue;

typedef struct Z_TargetInfo
{
    Z_CommonInfo *commonInfo;               /* OPTIONAL */
    /*
     * key elements
     */
    char *name;
    /*
     * non-key brief elements
     */
```

```
    Z_HumanString *recentNews;              /* OPTIONAL */
    Z_IconObject *icon;                      /* OPTIONAL */
    bool_t *namedResultSets;
    bool_t *multipleDbSearch;
    int *maxResultSets;                      /* OPTIONAL */
    int *maxResultSize;                      /* OPTIONAL */
    int *maxTerms;                           /* OPTIONAL */
    Z_IntUnit *timeoutInterval;              /* OPTIONAL */
    Z_HumanString *welcomeMessage;           /* OPTIONAL */
    /*
     * non-brief elements
     */
    Z_ContactInfo *contactInfo;              /* OPTIONAL */
    Z_HumanString *description;              /* OPTIONAL */
    int num_nicknames;
    char **nicknames;
    Z_HumanString *usageRest;               /* OPTIONAL */
    Z_HumanString *paymentAddr;              /* OPTIONAL */
    Z_HumanString *hours;                    /* OPTIONAL */
    int num_dbCombinations;
    Z_DatabaseList **dbCombinations;         /* OPTIONAL */
    int num_addresses;
    Z_NetworkAddress **addresses;            /* OPTIONAL */
    Z_AccessInfo *commonAccessInfo;          /* OPTIONAL */
} Z_TargetInfo;

typedef struct Z_DatabaseInfo
{
    Z_CommonInfo *commonInfo;                /* OPTIONAL */
    /*
     * Key elements
     */
    Z_DatabaseName *name;
    /*
     * Non-key elements.
     */
    Odr_null *explainDatabase;               /* OPTIONAL */
    int num_nicknames;
    Z_DatabaseName **nicknames;              /* OPTIONAL */
    Z_IconObject *icon;                      /* OPTIONAL */
    bool_t *userFee;
    bool_t *available;
    Z_HumanString *titleString;              /* OPTIONAL */
    /*
     * Non-brief elements.
     */
    int num_keywords;
    Z_HumanString **keywords;                /* OPTIONAL */
    Z_HumanString *description;              /* OPTIONAL */
    Z_DatabaseList *associatedDbs;           /* OPTIONAL */
    Z_DatabaseList *subDbs;                  /* OPTIONAL */
    Z_HumanString *disclaimers;              /* OPTIONAL */
    Z_HumanString *news;                     /* OPTIONAL */
    int recordCount_which;
#define Z_Exp_RecordCount_actualNumber 0
#define Z_Exp_RecordCount_approxNumber 1
    int *recordCount;                        /* OPTIONAL */
    Z_HumanString *defaultOrder;             /* OPTIONAL */
    int *avRecordSize;                       /* OPTIONAL */
    int *maxRecordSize;                      /* OPTIONAL */
    Z_HumanString *hours;                    /* OPTIONAL */
    Z_HumanString *bestTime;                 /* OPTIONAL */
    char *lastUpdate;          /* OPTIONAL */
    Z_IntUnit *updateInterval;               /* OPTIONAL */
```

```
    Z_HumanString *coverage;                /* OPTIONAL */
    bool_t *proprietary;                    /* OPTIONAL */
    Z_HumanString *copyrightText;           /* OPTIONAL */
    Z_HumanString *copyrightNotice;         /* OPTIONAL */
    Z_ContactInfo *producerContactInfo;     /* OPTIONAL */
    Z_ContactInfo *supplierContactInfo;     /* OPTIONAL */
    Z_ContactInfo *submissionContactInfo;   /* OPTIONAL */
    Z_AccessInfo *accessInfo;               /* OPTIONAL */
} Z_DatabaseInfo;

typedef struct Z_TagTypeMapping
{
    int *tagType;
    Odr_oid *tagSet;                        /* OPTIONAL */
    Odr_null *defaultTagType;               /* OPTIONAL */
} Z_TagTypeMapping;

typedef struct Z_PathUnit
{
    int *tagType;
    Z_StringOrNumeric *tagValue;
} Z_PathUnit;

typedef struct Z_Path
{
    int num;
    Z_PathUnit **list;
} Z_Path;

struct Z_ElementDataType;
typedef struct Z_ElementDataType Z_ElementDataType;

typedef struct Z_ElementInfo
{
    char *elementName;
    Z_Path *elementTagPath;
    Z_ElementDataType *dataType;            /* OPTIONAL */
    bool_t *required;
    bool_t *repeatable;
    Z_HumanString *description;             /* OPTIONAL */
} Z_ElementInfo;

typedef struct Z_ElementInfoList
{
    int num;
    Z_ElementInfo **list;
} Z_ElementInfoList;

struct Z_ElementDataType
{
    int which;
#define Z_ElementDataType_primitive 0
#define Z_ElementDataType_structured 1
    union
    {
       int *primitive;
#define Z_PrimitiveElement_octetString         0
#define Z_PrimitiveElement_numeric             1
#define Z_PrimitiveElement_date                2
#define Z_PrimitiveElement_external            3
#define Z_PrimitiveElement_string              4
#define Z_PrimitiveElement_trueOrFalse         5
#define Z_PrimitiveElement_oid                 6
#define Z_PrimitiveElement_intUnit             7
```

```
#define Z_PrimitiveElement_empty                8
#define Z_PrimitiveElement_noneOfTheAbove       100
   Z_ElementInfoList *structured;
    } u;
};

typedef struct Z_TagSetInfoElements
{
    char *elementName;
    int num_nicknames;
    char **nicknames;                           /* OPTIONAL */
    Z_StringOrNumeric *elementTag;
    Z_HumanString *description;                 /* OPTIONAL */
    int *dataType;                              /* OPTIONAL */
    /* (value as in Z_PrimitiveElement) */
    Z_OtherInformation *otherTagInfo;           /* OPTIONAL */
} Z_TagSetInfoElements;

typedef struct Z_SchemaInfo
{
    Z_CommonInfo *commonInfo;                   /* OPTIONAL */
    /*
     * Key elements
     */
    Odr_oid *schema;
    /*
     * Non-key elements
     */
    char *name;
    /*
     * Non-brief elements
     */
    Z_HumanString *description;                 /* OPTIONAL */
    int num_tagTypeMapping;
    Z_TagTypeMapping **tagTypeMapping;          /* OPTIONAL */
    int num_recordStructure;
    Z_ElementInfo **recordStructure;            /* OPTIONAL */
} Z_SchemaInfo;

typedef struct Z_RecordSyntaxInfo
{
    Z_CommonInfo *commonInfo;                   /* OPTIONAL */
    /*
     * Key elements
     */
    Odr_oid *recordSyntax;
    /*
     * Non-key elements
     */
    char *name;
    /*
     * Non-brief elements
     */
    int num_transferSyntaxes;
    Odr_oid **transferSyntaxes;                 /* OPTIONAL */
    Z_HumanString *description;                 /* OPTIONAL */
    char *asn1Module;                           /* OPTIONAL */
    int num_abstractStructure;
    Z_ElementInfo **abstractStructure;          /* OPTIONAL */
} Z_RecordSyntaxInfo;

typedef struct Z_AttributeDescription
{
    char *name;                                 /* OPTIONAL */
```

```
    Z_HumanString *description;             /* OPTIONAL */
    Z_StringOrNumeric *attributeValue;
    int num_equivalentAttributes;
    Z_StringOrNumeric **equivalentAttributes; /* OPTIONAL */
} Z_AttributeDescription;

typedef struct Z_AttributeType
{
    char *name;                             /* OPTIONAL */
    Z_HumanString *description;             /* OPTIONAL */
    int *attributeType;
    int num_attributeValues;
    Z_AttributeDescription **attributeValues;
} Z_AttributeType;

typedef struct Z_AttributeSetInfo
{
    Z_CommonInfo *commonInfo;               /* OPTIONAL */
    /*
     * Key elements
     */
    Odr_oid *attributeSet;
    /*
     * Non-key elements
     */
    char *name;
    /*
     * Non-brief elements
     */
    int num_attributes;
    Z_AttributeType **attributes;           /* OPTIONAL */
    Z_HumanString *description;             /* OPTIONAL */
} Z_AttributeSetInfo;

typedef struct Z_TermListElement
{
    char *name;
    Z_HumanString *title;                    /* OPTIONAL */
    int *searchCost;                         /* OPTIONAL */
#define Z_TermListInfo_optimized           0
#define Z_TermListInfo_normal              1
#define Z_TermListInfo_expensive           2
#define Z_TermListInfo_filter              3
    bool_t *scanable;
    int num_broader;
    char **broader;                          /* OPTIONAL */
    int num_narrower;
    char **narrower;                         /* OPTIONAL */
} Z_TermListElement;

typedef struct Z_TermListInfo
{
    Z_CommonInfo *commonInfo;               /* OPTIONAL */
    /*
     * Key elements
     */
    Z_DatabaseName *databaseName;
    /*
     * Non-key elements
     */
    int num_termLists;
    Z_TermListElement **termLists;
} Z_TermListInfo;
```

```
typedef struct Z_RecordTag
{
    Z_StringOrNumeric *qualifier;           /* OPTIONAL */
    Z_StringOrNumeric *tagValue;
} Z_RecordTag;

typedef struct Z_PerElementDetails
{
    char *name;                             /* OPTIONAL */
    Z_RecordTag *recordTag;                 /* OPTIONAL */
    int num_schemaTags;
    Z_Path **schemaTags;                    /* OPTIONAL */
    int *maxSize;                           /* OPTIONAL */
    int *minSize;                           /* OPTIONAL */
    int *avgSize;                           /* OPTIONAL */
    int *fixedSize;                         /* OPTIONAL */
    bool_t *repeatable;
    bool_t *required;
    Z_HumanString *description;             /* OPTIONAL */
    Z_HumanString *contents;                /* OPTIONAL */
    Z_HumanString *billingInfo;             /* OPTIONAL */
    Z_HumanString *restrictions;            /* OPTIONAL */
    int num_alternateNames;
    char **alternateNames;                  /* OPTIONAL */
    int num_genericNames;
    char **genericNames;                    /* OPTIONAL */
    Z_AttributeCombinations *searchAccess;  /* OPTIONAL */
} Z_PerElementDetails;

typedef struct Z_ElementSetDetails
{
    Z_CommonInfo *commonInfo;               /* OPTIONAL */
    /*
     * Key elements
     */
    Z_DatabaseName *databaseName;
    char *elementSetName;
    Odr_oid *recordSyntax;
    /*
     * Brief elements
     */
    Odr_oid *schema;
    /*
     * Non-brief elements
     */
    Z_HumanString *description;             /* OPTIONAL */
    int num_detailsPerElement;
    Z_PerElementDetails **detailsPerElement;  /* OPTIONAL */
} Z_ElementSetDetails;
```