IWF Metadata Harvester Manual

Metadata Harvesting.
Manual edition 1.0 for IWF Metadata Harvester Version 1.0
October 2006

Laurence D. Finston

This is the IWF Metadata Harvester User and Reference Manual, edition 1.0 for the IWF Metadata Harvester 1.0. This manual was last updated on 18 October 2006. IWF Metadata Harvester is a package for metadata harvesting. The author is Laurence D. Finston.

Copyright © 2006, 2007 IWF Wissen und Medien gGmbH

> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Short Contents

Table of Contents

1 Introduction

The IWF Metadata Harvester is a package for extracting data from servers, writing it to databases, and displaying it.

Numerous libraries, archives, and other institutions provide data publically or on a restricted basis on *servers* dedicated to this purpose. For example, many libraries make bibliographic data on their collections available on a server, for use by other libraries. Another institution or private person who wishes to access this data does so by means of a *client application* or *client*. Such a client initiates a *connection* to a server over a computer *network* and requests that the server send it particular data by means of one or more *queries*. If the data is available, the server then responds by sending the data over the network to the client. After the client has received the data, and has no further requests to make, it *terminates* the connection with the server.

1.1 Standards and Protocols

In order for the communication between client and server to function, both must adhere to a common set of *standards* and *protocols*. There are many standards governing the communication of clients and servers over computer networks, old and new, for various purposes, and development in this area continues apace.

These standards and protocols and their implementation is a complex subject, and different standards and protocols are not always directly comparable. It often useful to speak of the "kind" or "type" of a server, but it can be difficult to define precisely what this means.

Currently, the IWF Metadata Harvester can access two "types" of server: "OAI servers", i.e., ones using the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), and "Z39.50 Servers", i.e., ones using the Z39.50 protocol.

The IWF Metadata Harvester contains two programs, ATest and ZTest, whereby ATest accesses *OAI servers* and ZTest accesses *Z39.50 servers*. Of the two protocols, the OAI-PMH is much simpler, more straightforward, and more modern than Z39.50, which seems unnecessarily complex and old-fashioned in comparison. Nonetheless, Z39.50 does have some advantages over OAI-PMH, and is still widely used, especially by libraries.

1.2 Retrieving Data

1.2.1 OAI

Retrieving data from an OAI server is no problem, because one simply uses the Hypertext Transfer Protocol used in the World Wide Web (WWW) for transferring data in HTML (Hypertext Markup Language) format, such as web pages. Microsoft Visual C++ provides library functions for accessing servers using HTTP, as do other implementations of C++ libraries.

1.2.2 Z39.50

Ultimately, data is retrieved from a Z39.50 server using TCP/IP, but ZTest does not use these protocols directly. Instead, it uses the YAZ library supplied by the Danish company Index Data.

1.3 Data Formats

1.3.1 OAI

The data supplied by an OAI server is in *XML format*. XML has become a popular format for exchanging data in several different areas within the field of computer and database programming. It is a very clearly structured format and allows for a hierarchical organization of data. Because of its popularity, documentation is easily accessible, code samples and software for processing XML data is widely available, and the relevant standards committees are active. The use of such a popular and well-supported format for data exchange makes it easier to use OAI interfaces, because established methods can be used for accessing and manipulating the data. In fact, it was unnecessary to program an XML parser in ATest, because Microsoft SQL Server 2000 provides a *system stored procedure* for representing XML data in tabular form.

1.3.2 Z39.50

Z39.50 servers, on the other hand, can provide data in several different formats. The most popular is *USMARC*, formerly known as *MARC*. However, the IWF Metadata Harvester currently cannot process records in USMARC format. It does process records in PICA format, which is used by the GBV in Germany. The PICA format is much more concise than the rather verbose XML format, and Pica records are correspondingly more compact. However, it is very complex, with nearly 400 *categories*, each of which has at least one, and possibly several, *fields*. Since PICA format is little-used, outside of Germany and the Netherlands, ZTest includes a specially-written parser for parsing PICA records. However, it does not yet process all of the categories and fields, and the PICA database does not yet have database tables and columns to correspond to all of the PICA categories and fields.

2 About This Manual

2.1 Copying Conditions

See Appendix A [GNU Free Documentation License], page 156, (a.k.a. GNU FDL) for copying conditions. The code is contained in the file '`fdl.texi`', which should have been included in the in the distribution. If you haven't received a copy of the GNU Free Documentation License, you can obtain one from this address:

The Free Software Foundation
51 Franklin St, Fifth Floor
Boston, MA 02110-1301, USA
`http://www.gnu.org/`
`http://www.gnu.org/licenses/licenses.html`

3 Databases

3.1 Association Tables

The databases which are included in the IWF Metadata Harvester make heavy use of "association tables". These are database tables that associate entries of two or more other database tables.

Association tables are needed when a data record can contain multiple items of information of the same type. For example, a book may have more than one author. It would be possible to define a "`Book`" database table with a column "`author_name`", but then it would only be possible to store the name of a single author in a line of of the `Books` table.

A better approach is to define a second table "`Authors`". Both the `Books` table and the `Authors` table should have a column containing a unique, numeric identifier, `book_id`, and `author_id`, respectively. These columns should each be associated with *primary key constraints* for their table. Let us further assume that the `Authors` table has a column "`author_name`".

Now, a third table, "`Records_Authors`" can be defined with the two columns "`book_id`" and "`author_id`". These columns should be associated with *foreign key constraints* referencing the `Books` table and the `Authors` table, respectively.

Consider the case that a book has three authors. First, an entry is created in the `Books` table, and it is assigned a `book_id`, say `1`. Then, three entries are created in the `Authors` table, and they are assigned `author_ids`, for example, with the values '`1`', '`2`', and '`3`'. Then, three entries are created in the `Books_Authors` table as follows:

book_id	**author_id**
1	1
1	2
1	3

Now, the names of all of the authors associated with this book can be found by using the following Transact-SQL query:

```
select A.author_name
from Authors as A, Books as B, Books_Authors as BA
where
A.author_id = BA.author_id
and
B.book_id = BA.book_id
and
B.book_id = 1
```

3.2 "Continuation" Columns

When strings are too long, they are not written to the database correctly, but no error is signalled. However, `CDatabase::ExecuteSQL` or `CRecordset::Open` throws and exception, when trying to read the string out of the database: `Error Code 1021 ``Daten abgeschnitten'' (``Data truncated'')`.

This shouldn't happen, because enough space should have been allocated in the database table definitions. However, one never knows what a record might contain, so it's entirely possible that one might contain more text than expected. In order to account for this possibility, some of the tables in the PICA database have *continuation* columns. !! TODO: Explain.

4 Multi-Threading

Currently, neither ATest nor ZTest uses multi-threading. However, I have made an effort to ensure that it will be possible to do so, if need arises. I avoid the use of static non-constant variables in functions, and static variables local to files, in order to ensure that all functions are thread-safe. There are still some static variables left, but I am in the process of eliminating them.

5 Present Tasks

6 Future Tasks

7 Special Character Encodings.

See also “UTF-8 encoding table and Unicode characters” (`http://www.utf8-chartable.de/`) and “Universal Character Set — Wikipedia, the free encyclopedia” (`http://en.wikipedia.org/wiki/Universal_Character_Set`).

Hex.	Oct.	Dec.	UTF-8	Enc.	Sym.	Unicode Name
#80	°*200*	128	#C2 80	Â\200	\200	⟨control⟩
#81	°*201*	129	#C2 81	Â\201	\201	⟨control⟩
#82	°*202*	130	#C2 82	Â\202	\202	⟨control⟩
#83	°*203*	131	#C2 83	Â\203	\203	⟨control⟩
#84	°*204*	132	#C2 84	Â\204	\204	⟨control⟩
#85	°*205*	133	#C2 85	Â\205	\205	⟨control⟩
#86	°*206*	134	#C2 86	Â\206	\206	⟨control⟩
#87	°*207*	135	#C2 87	Â\207	\207	⟨control⟩
#88	°*210*	136	#C2 88	Â\210	\210	⟨control⟩
#89	°*211*	137	#C2 89	Â\211	\211	⟨control⟩
#8A	°*212*	138	#C2 8A	Â\212	\212	⟨control⟩
#8B	°*213*	139	#C2 8B	Â\213	\213	⟨control⟩
#8C	°*214*	140	#C2 8C	Â\214	\214	⟨control⟩
#8D	°*215*	141	#C2 8D	Â\215	\215	⟨control⟩
#8E	°*216*	142	#C2 8E	Â\216	\216	⟨control⟩
#8F	°*217*	143	#C2 8F	Â\217	\217	⟨control⟩
#93	°*223*	147	#C2 93	Â\223	\223	⟨control⟩
#96	°*226*	150	#C2 96	Â\226	\226	⟨control⟩
#97	°*227*	151	#C2 97	Â\227	\227	⟨control⟩
#9B	°*233*	155	#C2 9B	Â\233	\233	⟨control⟩
#9C	°*234*	156	#C2 9C	Â\234	\234	⟨control⟩
#9F	°*237*	159	#C2 9F	Â\237	\237	⟨control⟩
#A0	°*240*	160	#C2 A0	Â\240	\240	No-Break Space
#A1	°*241*	161	#C2 A1	Â¡	¡	Inverted Exclamation Mark
#A4	°*244*	164	#C2 A4	Â¤	¤	Currency Sign
#A8	°*250*	168	#C2 A8	Â¨	¨	Diaeresis
#A7	°*247*	167	#C2 A7	Â§	§	Section Sign
#A9	°*251*	169	#C2 A9	Â©	©	Copyright Sign
#AD	°*255*	173	#C2 AD	Â-	-	Soft Hyphen
#AE	°*256*	174	#C2 AE	Â®	®	Registered Sign
#AF	°*257*	175	#C2 AF	Â¯	¯	Macron

(cont.)

Hex.	Oct.	Dec.	UTF-8	Enc.	Sym.	Unicode Name
#B0	°260	176	#C2 B0	Â°	°	Degree Sign
#B1	°261	177	#C2 B1	Â±	±	Plus-Minus Sign
#B3	°263	179	#C2 B3	Â^3	^{3}	Superscript Three
#B4	°264	180	#C2 B4	Â´	´	Acute Accent
#B5	°265	181	#C2 B5	Âµ	µ	Micro Sign
#B6	°266	182	#C2 B6	Â¶	¶	Pilcrow Sign
#B7	°267	183	#C2 B7	Â·	·	Middle Dot
#B8	°270	184	#C2 B8	Â¸	¸	Cedilla
#BC	°274	188	#C2 BC	Â¼	¼	Vulgar Fraction One Quarter
#BD	°275	189	#C2 BD	Â½	½	Vulgar Fraction One Half
#BF	°277	191	#C2 BF	Â¿	¿	Inverted Question Mark
#C0	°300	192	#C3 80	Ã\200	À	Latin Capital Letter A With Grave
#C1	°301	193	#C3 81	Ã\201	Á	Latin Capital Letter A With Acute
#C2	°302	194	#C3 82	Ã\202	Â	Latin Capital Letter A With Circumflex
#C3	°303	195	#C3 83	Ã\203	Ã	Latin Capital Letter A With Tilde
#C4	°304	196	#C3 84	Ã\204	Ä	Latin Capital Letter A With Diaeresis
#C5	°305	197	#C3 85	Ã\205	Å	Latin Capital Letter A With Ring Above
#C6	°306	198	#C3 86	Ã\206	Æ	Latin Capital Letter AE
#C7	°307	199	#C3 87	Ã\207	Ç	Latin Capital Letter C With Cedilla
#C8	°310	200	#C3 88	Ã\210	È	Latin Capital Letter E With Grave
#C9	°311	201	#C3 89	Ã\211	É	Latin Capital Letter E With Acute
#CA	°312	202	#C3 8A	Ã\212	Ê	Latin Capital Letter E With Circumflex
#CB	°313	203	#C3 8B	Ã\213	Ë	Latin Capital Letter E With Diaeresis
#CE	°316	206	#C3 8E	Ã\216	Î	Latin Capital Letter I With Circumflex
#D6	°326	214	#C3 96	Ã\226	Ö	Latin Capital Letter O With Diaeresis
#D7	°327	215	#C3 97	Ã\227	×	Multiplication Sign
#DB	°333	219	#C3 9B	Ã\233	Û	Latin Capital Letter U With Circumflex
#DC	°334	220	#C3 9C	Ã\234	Ü	Latin Capital Letter U With Diaeresis
#DF	°337	223	#C3 9F	Ã\237	ß	Latin Small Letter Sharp S

(cont.)

Hex.	**Oct.**	**Dec.**	**UTF-8**	**Enc.**	**Sym.**	**Unicode Name**
#E0	°*340*	224	#C3 A0	Ã\240	à	Latin Small Letter A With Grave
#E1	°*341*	225	#C3 A1	Ã¡	á	Latin Small Letter A With Acute
#E4	°*344*	228	#C3 A4	Ã¤	ä	Latin Small Letter A With Diaeresis
#E7	°*347*	231	#C3 A7	Ã§	ç	Latin Small Letter C With Cedilla
#E8	°*350*	232	#C3 A8	Ã¨	è	Latin Small Letter E With Grave
#E9	°*351*	233	#C3 A9	Ã©	é	Latin Small Letter E With Acute
#ED	°*355*	237	#C3 AD	Ã-	í	Latin Small Letter I With Acute
#EE	°*356*	238	#C3 AE	Ã®	î	Latin Small Letter I With Circumflex
#EF	°*357*	239	#C3 AF	Ã¯	ï	Latin Small Letter I With Diaeresis
#F1	°*361*	241	#C3 B1	Ã±	ñ	Latin Small Letter N With Tilde
#F3	°*363*	243	#C3 B3	Ã³	ó	Latin Small Letter O With Acute
#F6	°*366*	246	#C3 B6	Ã¶	ö	Latin Small Letter O With Diaeresis
#FC	°*374*	252	#C3 BC	Ã¼	ü	Latin Small Letter U With Diaeresis
#FD	°*375*	253	#C3 BD	Ã½	ý	Latin Small Letter Y With Acute
#FF	°*377*	255	#C3 BF	Ã¿	ÿ	Latin Small Letter Y With Diaeresis

8 ATest

ATest is a package that retrieves records from servers using the *OAI* (*Open Archives Initiative*) interface. The Open Archives Initiative is a body that develops and publishes standards, protocols, etc., for the interchange of data. One such protocol is the *OAI–PMH* (*Open Archives Initiative–Protocol for Metadata Harvesting*).

Institutions that wish to make their archives publically available may set up an *OAI server*. Users can connect to the servers using The records are transmitted over the internet using the *Hypertext Transfer Protocol* (*HTTP*) in XML format. The OAI–PMH specifies the XML tags that can be used in the metadata made available via the OAI interface. They are based on the Dublin Core fields.

These are the steps that ATest performs:

1. Connect with an OAI server and send it a query. If the query is successful, the OAI server responds by returning an HTTP packet containing a set of metadata records in an XML structure.
2. Write the information returned by the query to a file. Special characters are converted to a coding that Microsoft SQL Server 2000 can handle.
3. Process the file of XML code. Extraneous information is discarded, and individual records are extracted and processed sequentially. Information is extracted and Transact-SQL commands are executed, in order to write this information to the appropriate tables in the `dc_test` database.
4. Once the data have been written to the database, other functions can be called to search the database and display its contents using various options.

9 Dialog_1

Class `Dialog_1` is declared in ‘`dialog1a.web`’.

9.1 Downloading

Group “Download Records”

Radio Button “Yesterday”
Radio Button “This week”
Radio Button “Last week”
Radio Button “This month”
Radio Button “Last month”
Radio Button “Last six months”
Radio Button “This year”
Radio Button “Last year”
Radio Button “Last two years”
Radio Button “Last five years”
Radio Button “Last 10 years”
Radio Button “Last 20 years”
Radio Button “All records”

Group The following radio buttons are used to determine from which OAI server the records should be downloaded. At present, ATest can only download records from these two servers.

Radio Button “TIMMS (Tübingen)”
Radio Button “DBT (Digitale Bibliothek Thüringen)”

Edit Box Contains messages for the user. The user doesn’t enter text into it.

Check Box “Delete Old Records”.
If checked, then the old records in the `dc_test` database are deleted before the data from the newly-downloaded records is written to it. This is useful for testing purposes, but will not be done in production.

Button “Download”
ATest begins downloading records according to the options specified by the user, or the default settings. No further actions are possible until downloading is completed, and the data from the records are stored in the database.

Button “Continue”
The current dialog box is closed. Clicking the right mouse button in the empty view window that remains causes the second dialog box to appear.

Button “Cancel”
ATest exits.

9.2 Data Members

`unsigned short timespan` [Protected Variable]

`CString edit_1_str` [Protected Variable]

`unsigned short metadata_source` [Protected Variable]

`int day_of_month` [Protected Variable]

`int day_of_week` [Protected Variable]

`int month_of_year` [Protected Variable]

`CTime t0` [Protected Variable]

9.3 Functions

9.3.1 Constructor

`void Dialog_1` ([`CWnd*` *pParent* = `NULL`]) [Constructor]

9.3.2 Destructor

`void ~Dialog_1 (void)` [Destructor]

9.3.3 Data Exchange

`void DoDataExchange` (`CDataExchange*` *pDX*) [Virtual Protected Function]

9.3.4 Downloading Records

`int download_records (` [Function]
`const unsigned short` *metadata_source*,
`const unsigned short` *ttimespan*,
`const CTime*` *ctime*,
`char*` *resumption_token*,
`char*` *records_file_name*)

9.3.5 Event Handlers

`BOOL OnInitDialog (void)` [Virtual Function]

`afx_msg void OnBnClickedOk (void)` [Function]

`afx_msg void OnBnClickedCancel (void)` [Function]

`afx_msg void OnBnClickedDownload (void)` [Function]

`afx_msg void OnBnClickedAllRecords (void)` [Function]

`afx_msg void OnBnClickedTimms (void)` [Functions]
`afx_msg void OnBnClickedDbt (void)`

`afx_msg void OnBnClickedToday (void)` [Functions]
`afx_msg void OnBnClickedYesterday (void)`

```
afx_msg void OnBnClickedThisWeek (void)
afx_msg void OnBnClickedLastWeek (void)
afx_msg void OnBnClickedThisMonth (void)
afx_msg void OnBnClickedLastMonth (void)
afx_msg void OnBnClickedLast6Months (void)
afx_msg void OnBnClickedThisYear (void)
afx_msg void OnBnClickedLastYear (void)
afx_msg void OnBnClickedLast2Years (void)
afx_msg void OnBnClickedLast5Years (void)
afx_msg void OnBnClickedLast10Years (void)
afx_msg void OnBnClickedLast20Years (void)
```

10 Dialog_2

Class `Dialog_2` is declared in '`dialog2a.web`'.

10.1 Searching

The top half of the second dialog box is used for searching the `dc_test` database. It contains, from left-to-right, the following items:

Edit Box "Search String"
: Caption: "Search for string:". The user enters the string to be searched for in the database.

List Box "Tables"
: Caption: "in tables:". The user can choose from a list of database tables, and/or combinations of database tables, which will be searched for the specified string. It is possible to choose more than one entry from the list.

Group "Search Options"
: This group contains a set of radio buttons and a check box for controlling the way in which the search is performed.

  Radio Button "Beginning of word or whole word"
  Radio Button "Whole word only"
  Radio Button "Whole or partial word"
  Radio Button "Exact match"
  Check Box "Ignore case"

10.2 Listing

The bottom half of the second dialog box contains items used for listing the contents of the `dc_test` database in various ways. The listings are stored in files of HTML code. The following table describes these items, as they appear from left-to-right, and from top-to-bottom:

Static Text "List all records"
List Box "Sorted by"
Check Box "Descending"
Check Box "No duplicates"
Group "Timespan"

  Group (Specifier)
  : No caption.

    Radio Button "No limit"
    Radio Button "Last Year and This Year"
    Radio Button "This Year Only"
    Radio Button "Last Six Months"
    Radio Button "Last Month and This Month"
    Radio Button "This Month Only"
    Radio Button "This Week"

  Group (Date type)
  : No caption.

Radio Button "Use dc:date"
Radio Button "Use header_datestamp"

Button "List Records"
Group (List Fields)
No caption

Button "Titles"
The output is written to 'titles.html'.

Button "Creators"
The output is written to 'creators.html'.

Button "Contributors"
The output is written to 'contributors.html'.

Button "Subjects"
The output is written to 'subjects.html'.

Edit Box "Messages"
Button "OK"
Button "Cancel"

10.3 Data Members

`unsigned int select_value` [Protected Variable]

`CString results_str` [Protected Variable]

`CString search_str` [Protected Variable]

`BOOL ignore_case` [Protected Variable]

`unsigned int search_options` [Protected Variable]

`unsigned short timespan` [Protected Variable]

`unsigned short sort_order` [Protected Variable]

`unsigned short sort_field` [Protected Variable]

`unsigned short use_date_type` [Protected Variable]

`BOOL suppress_duplicate_records` [Protected Variable]

10.4 Functions

10.4.1 Constructor

void Dialog_2 ([CWnd* *pParent* = NULL]) [Constructor]

10.4.2 Destructor

void ~Dialog_2 (void) [Destructor]

10.4.3 Exchanging Data

void DoDataExchange (CDataExchange* *pDX*) [Protected virtual function]

10.4.4 Event Handlers

`BOOL OnInitDialog (void)` [Virtual function]

`afx_msg void OnBnClickedOk (void)` [Function]

`afx_msg void OnBnClickedCancel (void)` [Function]

`afx_msg void OnBnClickedSearch (void)` [Function]

`afx_msg void OnBnClickedBegOrWholeWord (void)` [Functions]
`afx_msg void OnBnClickedWholeWordOnly (void)`
`afx_msg void OnBnClickedWholeOrPartialWord (void)` [Function]

`afx_msg void OnBnClickedCaseIgnore (void)` [Function]

`afx_msg void OnBnClickedExactMatch (void)` [Function]

`afx_msg void OnBnClickedAllDates (void)` [Function]

`afx_msg void OnBnClickedSinceLastYear (void)` [Functions]
`afx_msg void OnBnClickedThisYear (void)`
`afx_msg void OnBnClickedLast6Months (void)` [Function]
`afx_msg void OnBnClickedLastMonth (void)` [Function]
`afx_msg void OnBnClickedThisMonth (void)` [Function]
`afx_msg void OnBnClickedThisWeek (void)` [Function]

`afx_msg void OnBnClickedListRecords (void)` [Function]

`afx_msg void OnBnClickedDescending (void)` [Function]

`afx_msg void OnBnClickedUseDcDate (void)` [Function]
`afx_msg void OnBnClickedUseHeaderDatestamp (void)` [Function]

`afx_msg void OnBnClickedListTitles (void)` [Function]

`afx_msg void OnBnClickedCreators (void)` [Function]
`afx_msg void OnBnClickedContributors (void)` [Function]
`afx_msg void OnBnClickedSubjects (void)` [Function]

11 Global Functions

`int get_http_file` (LPCTSTR *pszURL*) [Function]

12 MetadataSource

Class `MetadataSource` is declared in '`mtdtsrc.web`'.

12.1 Data Members

`unsigned short NULL_METADATA_SOURCE` [Public static constants]
`unsigned short TIMMS`
`unsigned short DBT`

`unsigned short MAX_TAG_LENGTH` [Public static constant]

`unsigned short MAX_RESUMPTION_TOKEN` [Public static constant]

12.2 Functions

`unsigned short id` [Protected variable]

12.2.1 Constructor

`void MetadataSource (void)` [Default Constructor]

`void MetadataSource (const unsigned short` *iid*`)` [Constructor]

12.2.2 Destructor

`void ~MetadataSource (void)` [Destructor]

12.2.3 Parsing Records

`int parse_record (ifstream*` *in_strm*, `string*` *out_str*, `char*` *resumption_token*`)` [Function]

12.2.4 Updating the Database

`int update_database (` [Function]
`BOOL` *delete_tables*,
`char*` *resumption_token*,
`char*` *records_file_name*`)`

`int sub_update_database_dbt (` [Function]
`CDatabase*` *cdb*,
`CRecords&` *curr_record*,
`char*` *resumption_token*,
`char*` *records_file_name*`)`

`int sub_update_database_timms (` [Function]
`CDatabase*` *cdb*,
`CRecords&` *curr_record*,
`char*` *resumption_token*,
`char*` *records_file_name*`)`

13 Selector

Class `Selector` is declared in '`selector.web`'.

13.1 Data Members

13.1.1 Constants

unsigned short QUERY_NULL_TYPE [Public Static Constants]
unsigned short QUERY_SEARCH
unsigned short QUERY_LISTING

unsigned short NULL_TIMESPAN [Public Static Constants]
unsigned short TODAY
unsigned short YESTERDAY
unsigned short THIS_WEEK
unsigned short LAST_WEEK
unsigned short THIS_MONTH
unsigned short LAST_MONTH
unsigned short LAST_6_MONTHS
unsigned short THIS_YEAR
unsigned short LAST_YEAR
unsigned short LAST_2_YEARS
unsigned short LAST_5_YEARS
unsigned short LAST_10_YEARS
unsigned short LAST_20_YEARS
unsigned short ALL_RECORDS

unsigned short USE_DC_DATE [Public Static Constants]
unsigned short USE_HEADER_DATESTAMP

unsigned short SORT_ASCENDING [Public Static Constants]
unsigned short SORT_DESCENDING

unsigned short SORT_FIELD_RECORD_ID [Public Static Constants]
unsigned short SORT_FIELD_CREATOR
unsigned short SORT_FIELD_TITLE
unsigned short SORT_FIELD_DC_DATE
unsigned short SORT_FIELD_HEADER_DATESTAMP

unsigned int CONTRIBUTORS [Public Static Constants]
unsigned int CREATORS
unsigned int DC_DATES
unsigned int DESCRIPTIONS
unsigned int DESCRIPTIONS
unsigned int HEADER_DATESTAMPS
unsigned int IDENTIFIERS
unsigned int LANGUAGES
unsigned int PUBLISHERS
unsigned int RIGHTS

unsigned int SUBJECTS
unsigned int TITLES
unsigned int TYPES

unsigned int BEG_OR_WHOLE_WORD [Public Static Constants]
unsigned short WHOLE_WORD_ONLY
unsigned short WHOLE_OR_PARTIAL_WORD
unsigned short EXACT_MATCH

unsigned int IGNORE_CASE [Public Static Constants]

13.1.2 Variables

CRecords curr_record [Protected Variable]

CDatabase* cdb [Variable]

CRecords records [Variables]
CContributors contributors
CCreators creators
CSubjects subjects
CTitles titles
CTemp_IDs temp_ids
CTemp_IDs_1 temp_ids_1
CRecords_Temp records_temp
CContributors_Temp contributors_temp
CCreators_Temp creators_temp
CDescriptions_Temp descriptions_temp
CIdentifiers_Temp identifiers_temp
CLanguages_Temp languages_temp
CPublishers_Temp publishers_temp
CRights_Temp rights_temp
CSubjects_Temp subjects_temp
CTitles_Temp titles_temp
CTypes_Temp types_temp

These variables reference database tables.

stringstream contributor_strm [Variables]
stringstream creator_strm
stringstream dc_date_strm
stringstream description_strm
stringstream header_datestamp_strm
stringstream identifier_strm
stringstream language_strm
stringstream publisher_strm
stringstream rights_strm
stringstream subject_strm
stringstream title_strm
stringstream type_strm

`stringstream temp_strm`

`unsigned short use_date_type` [Variable]

`unsigned short query_type` [Variable]

`ofstream html_strm` [Variable]

`map<unsigned short, string> sort_field_map` [Variable]

`map<unsigned short, string> timespan_map` [Variable]

13.2 Functions

13.2.1 Constructor

`void Selector (void)` [Constructor]

13.2.2 Destructor

`void ~Selector (void)` [Destructor]

13.2.3 Selecting Records

`int select_from_database` (const unsigned int *select_value*, CString& *search_str*, const unsigned int *search_options*) [Function]

`int fill_table_streams` (CDatabase* *cdb*)

13.2.4 Listing Records

`int list_records` (unsigned short *sort_field*, unsigned short *sort_order*, unsigned short *timespan*, BOOL *suppress_duplicate_records*) [Function]

`int write_html_from_streams` (unsigned short *query_type*, [unsigned int *select_value* = *0*, [CString* *search_str* = *0*]]) [Function]

`int list_table` (unsigned short *table*, unsigned short *timespan*, unsigned short *sort_order*) [Function]

14 ODBC Classes for ATest

14.1 Records

Class `Records` is declared in '`records.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Records` table in the `dc_test` database. See Section 15.1 [Records (ATest Database Tables)], page 30.

Data Members

`long m_record_id` [Variables]
`CStringA m_header_identifier`
`CTime m_header_datestamp`
`CStringA m_header_status`
`CTime m_dc_date`

These variables reference the corresponding columns in the `Records` table in the `dc_test` database. See Section 15.1 [Records (ATest Database Tables)], page 30.

14.2 Records_Temp

Class `Records_Temp` is declared in '`rcrdstmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Records_Temp` table in the `dc_test` database. See Section 15.2 [Records_Temp (ATest Database Tables)], page 30.

Data Members

`long m_record_id` [Variables]
`CStringA m_header_identifier`
`CTime m_header_datestamp`
`CStringA m_header_status`
`CTime m_dc_date`

These variables reference the corresponding columns in the `Records_Temp` table in the `dc_test` database. See Section 15.2 [Records_Temp (ATest Database Tables)], page 30.

14.3 Creators

Class `Creators` is declared in '`creators.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Creators` table in the `dc_test` database. See Section 15.3 [Creators (ATest Database Tables)], page 31.

Data Members

`long m_creator_id` [Variables]
`CStringA m_dc_creator`
`long m_institution_id`
`long m_person_id`

These variables reference the corresponding columns in the `Creators` table in the `dc_test` database. See Section 15.3 [Creators (ATest Database Tables)], page 31.

14.4 Creators_Temp

Class `Creators_Temp` is declared in '`crtrstmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Creators_Temp` table in the `dc_test` database. See Section 15.4 [Creators_Temp (ATest Database Tables)], page 31.

Data Members

`long m_creator_id` [Variables]
`CStringA m_dc_creator`
`long m_institution_id`
`long m_person_id`

> These variables reference the corresponding columns in the `Creators_Temp` table in the `dc_test` database. See Section 15.4 [Creators_Temp (ATest Database Tables)], page 31.

14.5 Contributors

Class `Contributors` is declared in '`cntrbtrs.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Contributors` table in the `dc_test` database. See Section 15.5 [Contributors (ATest Database Tables)], page 31.

Data Members

`long m_contributor_id` [Variable]
`CStringA m_dc_contributor`
`long m_institution_id`
`long m_person_id`

> These variables reference the corresponding columns in the `Contributors` table in the `dc_test` database. See Section 15.5 [Contributors (ATest Database Tables)], page 31.

14.6 Contributors_Temp

Class `Contributors_Temp` is declared in '`cntrbtmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Contributors_Temp` table in the `dc_test` database. See Section 15.6 [Contributors_Temp (ATest Database Tables)], page 32.

Data Members

`long m_contributor_id` [Variables]
`CStringA m_dc_contributor`
`long m_institution_id`
`long m_person_id`

> These variables reference the corresponding columns in the `Contributors_Temp` table in the `dc_test` database. See Section 15.6 [Contributors_Temp (ATest Database Tables)], page 32.

14.7 Titles

Class `Titles` is declared in '`titles.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Titles` table in the `dc_test` database. See Section 15.7 [Titles (ATest Database Tables)], page 32.

Data Members

`long m_title_id` [Variables]
`long m_record_id`
`CStringA m_dc_title`

These variables reference the corresponding columns in the `Titles` table in the `dc_test` database. See Section 15.7 [Titles (ATest Database Tables)], page 32.

14.8 Titles_Temp

Class `Titles_Temp` is declared in '`ttlstmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Titles_Temp` table in the `dc_test` database. See Section 15.8 [Titles_Temp (ATest Database Tables)], page 32.

Data Members

`long m_title_id` [Variables]
`long m_record_id`
`CStringA m_dc_title`

These variables reference the corresponding columns in the `Titles_Temp` table in the `dc_test` database. See Section 15.8 [Titles_Temp (ATest Database Tables)], page 32.

14.9 Descriptions_Temp

Class `Descriptions_Temp` is declared in '`dscrptmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. See Section 15.10 [Descriptions_Temp (ATest Database Tables)], page 33.

Data Members

`long m_description_id` [Variables]
`long m_record_id`
`CStringA m_dc_description`

These variables reference the corresponding columns in the `Descriptions_Temp` table in the `dc_test` database. See Section 15.10 [Descriptions_Temp (ATest Database Tables)], page 33.

14.10 Subjects

Class `Subjects` is declared in '`subjects.web`'. It is derived from the MFC class `CRecordset` using public derivation. See Section 15.11 [Subjects (ATest Database Tables)], page 33.

Data Members

`long m_subject_id` [Variables]

`CStringA m_dc_subject`
These variables reference the corresponding columns in the `Subjects` table in the `dc_test` database. See Section 15.11 [Subjects (ATest Database Tables)], page 33.

14.11 Subjects_Temp

Class `Subjects_Temp` is declared in '`sbjcttmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Subjects_Temp` table in the `dc_test` database. See Section 15.12 [Subjects_Temp (ATest Database Tables)], page 33.

Data Members

`long m_subject_id` [Variables]
`CStringA m_dc_subject`
These variables reference the corresponding columns in the `Subjects_Temp` table in the `dc_test` database. See Section 15.12 [Subjects_Temp (ATest Database Tables)], page 33.

14.12 Identifiers_Temp

Class `Identifiers_Temp` is declared in '`identtmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. See Section 15.14 [Identifiers_Temp (ATest Database Tables)], page 33.

Data Members

`long m_identifier_id` [Variables]
`CStringA m_dc_identifier`
These variables reference the corresponding columns in the `Identifiers_Temp` table in the `dc_test` database. See Section 15.14 [Identifiers_Temp (ATest Database Tables)], page 33.

14.13 Languages_Temp

Class `Languages_Temp` is declared in '`langtmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. See Section 15.16 [Languages_Temp (ATest Database Tables)], page 34.

Data Members

`long m_language_id` [Variables]
`CStringA m_dc_language`
These variables reference the corresponding columns in the `Languages_Temp` table in the `dc_test` database. See Section 15.16 [Languages_Temp (ATest Database Tables)], page 34.

14.14 Publishers_Temp

Class `Publishers_Temp` is declared in '`pblshtmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. See Section 15.18 [Publishers_Temp (ATest Database Tables)], page 34.

Data Members

`long m_publisher_id` [Variables]
`CStringA m_dc_publisher`
`long m_person_id`
`long m_institution_id`
`long m_company_id`

> These variables reference the corresponding columns in the `Publishers_Temp` table in the `dc_test` database. See Section 15.18 [Publishers_Temp (ATest Database Tables)], page 34.

14.15 Rights_Temp

Class `Rights_Temp` is declared in '`rghtstmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. See Section 15.20 [Rights_Temp (ATest Database Tables)], page 35.

Data Members

`long m_rights_id` [Variables]
`CStringA m_dc_rights`

> These variables reference the corresponding columns in the `Rights_Temp` table in the `dc_test` database. See Section 15.20 [Rights_Temp (ATest Database Tables)], page 35.

14.16 Types_Temp

Class `Types_Temp` is declared in '`typestmp.web`'. It is derived from the MFC class `CRecordset` using public derivation. See Section 15.22 [Types_Temp (ATest Database Tables)], page 35.

Data Members

`long m_type_id` [Variables]
`CStringA m_dc_type`

> These variables reference the corresponding columns in the `Types_Temp` table in the `dc_test` database. See Section 15.22 [Types_Temp (ATest Database Tables)], page 35.

14.17 Temp_IDs

Class `Temp_IDs` is declared in '`tempids.web`'. It is derived from the MFC class `CRecordset` using public derivation. See Section 15.23 [Temp_IDs (ATest Database Tables)], page 35.

Data Members

`long m_temp_id` [Variable]

> This variable references the corresponding column in the `Temp_IDs` table in the `dc_test` database. See Section 15.23 [Temp_IDs (ATest Database Tables)], page 35.

14.18 Temp_IDs_1

Class `Temp_IDs_1` is declared in '`tempids1.web`'. It is derived from the MFC class `CRecordset` using public derivation. See Section 15.24 [Temp_IDs_1 (ATest Database Tables)], page 35.

Data Members

`long m_temp_id` [Variable]

This variable references the corresponding column in the `Temp_IDs_1` table in the `dc_test` database. See Section 15.24 [Temp_IDs_1 (ATest Database Tables)], page 35.

15 Database Tables for ATest

The database tables for ATest are defined in the file 'dc_test/create_tables.sql'.

!! TODO: Write about temporary tables.

15.1 Records

This database table is referenced by the ODBC class Records. See Section 14.1 [Records (ATest; ODBC Classes)], page 24.

int identity not null record_id [Column]
primary key PK_Records [Constraint]

The values stored in the record_id column are the unique identifiers for the records stored in the Records table. These values are used in other tables, including association tables, for associating data from other tables with particular records.

varchar(512) not null header_identifier [Column]

The default value is 'N/A'.

datetime not null header_datestamp [Column]

varchar(128) not null header_status [Column]

The default value is 'normal'.

datetime null dc_date [Column]

int not null source_id [Column]

References Sources(source_id). The default value is 0.

15.2 Records_Temp

This database table is referenced by the ODBC class Records_Temp. See Section 14.2 [Records_Temp (ATest; ODBC Classes)], page 24.

int not null record_id [Column]

varchar(512) not null header_identifier [Column]

The default value is 'N/A'.

datetime not null header_datestamp [Column]

The default value is 'N/A'.

varchar(128) not null header_status [Column]

The default value is 'normal'.

datetime null dc_date [Column]

int not null source_id [Column]

15.3 Creators

This database table is referenced by the ODBC class `Creators`. See Section 14.3 [Creators (ATest; ODBC Classes)], page 24.

`int identity (0, 1) not null creator_id` [Column]
`primary key PK_Creators` [Constraint]

`varchar(64) not null dc_creator` [Column]
  The default value is `'N/A'`.

`int not null institution_id` [Column]
  References `Institutions(institution_id)`. The default value is 0.

`int not null person_id` [Column]
  References `Persons(person_id)`. The default value is 0.

15.4 Creators_Temp

This database table is referenced by the ODBC class `Creators_Temp`. See Section 14.4 [Creators_Temp (ATest; ODBC Classes)], page 25.

`int not null creator_id` [Column]

`varchar(64) not null dc_creator` [Column]
  The default value is `'N/A'`.

`int not null institution_id` [Column]

`int not null person_id` [Column]

15.5 Contributors

This database table is referenced by the ODBC class `Contributors`. See Section 14.5 [Contributors (ATest; ODBC Classes)], page 25.

`int identity not null contributor_id` [Column]
`primary key PK_Contributors` [Constraint]

`varchar(512) not null dc_contributor` [Column]
  The default value is `'N/A'`.

`int not null institution_id` [Column]
  References `Institutions(institution_id)`. The default value is `0`.

`int not null person_id` [Column]
  References `Persons(institution_id)`. The default value is `0`.

15.6 Contributors_Temp

This database table is referenced by the ODBC class `Contributors_Temp`. See Section 14.6 [Contributors_Temp (ATest; ODBC Classes)], page 25.

`int not null person_id` [Column]
  The default value is 0.

`int not null contributor_id` [Column]

`varchar(512) not null dc_contributor` [Column]
  The default value is `'N/A'`.

`int not null institution_id` [Column]
  The default value is 0.

15.7 Titles

This database table is referenced by the ODBC class `Titles`. See Section 14.7 [Titles (ATest; ODBC Classes)], page 25.

`int identity not null title_id` [Column]
`primary key PK_Titles` [Constraint]

`int not null record_id` [Column]
  References `Records(record_id)`.

`varchar(512) dc_title` [Column]

15.8 Titles_Temp

This database table is referenced by the ODBC class `Titles_Temp`. See Section 14.8 [Titles_Temp (ATest; ODBC Classes)], page 26.

`int not null title_id` [Column]

`int not null record_id` [Column]

`varchar(512) dc_title` [Column]

15.9 Descriptions

`int identity not null description_id` [Column]
`primary key PK_Descriptions` [Constraint]

`int not null record_id` [Column]
  References `Records(record_id)`.

`varchar(2048) not null dc_description` [Column]
  The default value is `'N/A'`.

15.10 Descriptions_Temp

This database table is referenced by the ODBC class `Descriptions_Temp`. See Section 14.9 [Descriptions_Temp (ATest; ODBC Classes)], page 26.

`int not null description_id` [Column]

`int not null record_id` [Column]

`varchar(2048) not null dc_description` [Column]
The default value is `'N/A'`.

15.11 Subjects

This database table is referenced by the ODBC class `Subjects`. See Section 14.10 [Subjects (ATest; ODBC Classes)], page 26.

`int identity not null subject_id` [Column]
`primary key PK_Subjects` [Constraint]

`varchar(512) unique not null dc_subject` [Column]

15.12 Subjects_Temp

This database table is referenced by the ODBC class `Subjects_Temp`. See Section 14.11 [Subjects_Temp (ATest; ODBC Classes)], page 27.

`int not null subject_id` [Column]

`varchar(512) not null dc_subject` [Column]

15.13 Identifiers

`int identity not null identifier_id` [Column]
`primary key PK_Identifiers` [Constraint]

`varchar(1024) not null dc_identifier` [Column]
The default value is `'N/A'`.

15.14 Identifiers_Temp

This database table is referenced by the ODBC class `Identifiers_Temp`. See Section 14.12 [Identifiers_Temp (ATest; ODBC Classes)], page 27.

`int not null identifier_id` [Column]

`varchar(1024) not null dc_identifier` [Column]
The default value is `'N/A'`.

15.15 Languages

`int identity not null language_id` [Column]
`primary key PK_Languages` [Constraint]

`varchar(64) not null dc_language` [Column]
The default value is `'N/A'`.

15.16 Languages_Temp

This database table is referenced by the ODBC class `Languages_Temp`. See Section 14.13 [Languages_Temp (ATest; ODBC Classes)], page 27.

`int not null language_id` [Column]

`varchar(64) not null  dc_language` [Column]
The default value is `'N/A'`.

15.17 Publishers

`int identity not null publisher_id` [Column]
`primary key PK_Publishers` [Constraint]

`varchar(64) not null dc_publisher` [Column]
The default value is `'N/A'`.

`int not null person_id` [Column]
References `Persons(person_id)`. The default value is 0.

`int not null institution_id` [Column]
References `Institutions(institution_id)`. The default value is 0.

`int not null company_id` [Column]
References `Companies(company_id)`. The default value is 0.

15.18 Publishers_Temp

This database table is referenced by the ODBC class `Publishers_Temp`. See Section 14.14 [Publishers_Temp (ATest; ODBC Classes)], page 27.

`int not null publisher_id` [Column]

`varchar(64) not null dc_publisher` [Column]
The default value is `'N/A'`.

`int not null person_id` [Column]
The default value is 0.

`int not null institution_id` [Column]
The default value is 0.

`int not null company_id` [Column]
The default value is 0.

15.19 Rights

`int identity not null rights_id` [Column]
`primary key PK_Rights` [Constraint]

`varchar(1024) not null dc_rights` [Column]
The default value is `'N/A'`.

15.20 Rights_Temp

This database table is referenced by the ODBC class `Rights_Temp`. See Section 14.15 [Rights_Temp (ATest; ODBC Classes)], page 28.

`int not null rights_id` [Column]

`varchar(1024) not null  dc_rights` [Column]

The default value is `'N/A'`.

15.21 Types

`int identity (0,1) not null type_id` [Column]
`primary key PK_Types` [Constraint]

`varchar(64) not null dc_type` [Column]

The default value is `'N/A'`.

15.22 Types_Temp

This database table is referenced by the ODBC class `Types_Temp`. See Section 14.16 [Types_Temp (ATest; ODBC Classes)], page 28.

`int not null type_id` [Column]

`varchar(64) not null dc_type` [Column]

The default value is `'N/A'`.

15.23 Temp_IDs

This database table is referenced by the ODBC class `Temp_IDs`. See Section 14.17 [Temp_IDs (ATest; ODBC Classes)], page 28.

`int temp_id` [Column]

15.24 Temp_IDs_1

This database table is referenced by the ODBC class `Temp_IDs_1`. See Section 14.18 [Temp_IDs_1 (ATest; ODBC Classes)], page 29.

`int temp_id` [Column]

16 Database Stored Procedures

`create_catalogues` [Stored Procedure]

`create_tables` [Stored Procedure]

`delete_tables` [Stored Procedure]

`drop_tables` [Stored Procedure]

`fill_catalogues` [Stored Procedure]

`list_records (` [Stored Procedure]
```
        int @sort_field ,
        int @sort_order,
        bit @timespan)
```

`search_for_records (` [Stored Procedure]
```
        nvarchar(128) @search_str,
        int @search_options,
        bit @contributor_value,
        bit @creator_value,
        bit @description_value,
        bit @identifier_value,
        bit @language_value,
        bit @publisher_value,
        bit @rights_value,
        bit @subject_value,
        bit @title_value,
        bit @type_value)
```

`store_in_temp_tables (int @temp_record_id)` [Stored Procedure]

`write_to_tables (` [Stored Procedure]
```
        int @source_number,
        bigint @record_ctr,
        ntext @XML_data,
        nvarchar(4000) @namespace_data)
```

17 ZTest: Z39.50

ZTest is a package that retrieves records from servers using the Z39.50 interface, extracts information from these records, writes it to a database. It also provides various ways of searching and displaying this information.

Z39.50 is a set of standards, protocols, etc. It is used by many institutions throughout the world, especially libraries. It is a rather old set of standards, and quite complex.

!! TODO: Check the information in this paragraph. A Z39.50 interface can provide data in a number of different formats. Currently, ZTest can only process data in the *PICA* format. This format is widely used in Germany and the Netherlands, but not much elsewhere. The most commonly-used format is *USMARC* (formerly known as "*MARC*"). It would be desirable for ZTest to be able to process USMARC data as well.

The term PICA actually refers to a a pair of formats, *Pica+* and *Pica3*. Librarians who catalogue data and prepare it to be made available via a Z39.50 interface use Pica3. This data is then converted by a computer program to Pica+ format. Pica+ is thus the format of the data retrieved from the Z39.50 interface and is what ZTest processes.

17.1 Global Variables

`CMutex time_mutex` [Global Variable]

Declared in '`ztest.web`' and declared `extern` in '`stdafx.web`'.

17.2 Global Functions

`int zoomtst2 (` [Global Function]
`unsigned short` *server_selector*,
`CString` *search_command*,
[`bool` *perform_search* = `true`,
[`bool` *write_records_file* = `true`,
[`bool` *clear_database* = `true`,
[`bool` *parse_records* = `true`,
[`bool` *display_records* = `true`]]]]])

Defined in '`ztstzoom.web`'.

17.3 Adding Category and Field Functions ZTest

1. Add code to `ZClient::init_category_map`.
2. Add declarations and definitions of `f_` functions to '`sbctgcnt.web`'.
3. Add declarations and definitions of `F_` functions, if any, to '`ctgcntnr.web`'.

18 How ZTest Works

When ZTest is started, a *dialog box* is displayed. This dialog box is associated with the class `Dialog_Z_1`, which is described in Chapter 19 [Dialog_Z_1 ZTest], page 40. Clicking the "OK" button will cause the *event handler* `Dialog_Z_1::OnOK` to be called. See Section 19.3.3 [Event Handlers], page 41. `Dialog_Z_1::OnOK`, in turn, calls the global function `zoomtst2`. See Section 17.2 [Global Functions], page 37. Assuming all of the options are checked, `zoomtst2` performs the following actions:

1. **Create a `ZClient` object.**

   An identification number, the name of the server, and, optionally, a port number, are passed to the `ZClient` *constructor*, e.g.,

   ```
   ZClient z(ZClient::GBV_GVK_ID, "z3950.gbv.de:20010/GVK");
   ```

   See Section 20.2.1 [ZClient; Constructors], page 42. `ZClient::GBV_GVK_ID` is the identifier for the the German *Gemeinsamer Verbundkatalog* (GVK, engl. Union Catalogue) of the *Gemeinsamer Bibliotheksverbund* (GBV, engl. Common Library Network). See Section 20.1.1 [ZClient; Static Constants], page 42. The corresponding server, i.e., `z3950.gbv.de:20010/GVK`, is so far the only one addressed by ZTest.

2. **Initialize `ZClient::category_map`**

   `ZClient::category_map` is a `protected` data member of `ZClient`, and is of type `Category_Map_Type`. The latter is a *type definition* (or "*typedef*") for `map<string, Category_Container*>`. See Chapter 26 [Category_Container], page 70.

   Initialization is performed by the function `ZClient::init_category_map`. See Section 20.2.3 [ZClient Initialization], page 42.

3. **Search**

   The search query in the edit box will be sent to the Z39.50 interface.

4. **Write records file**

   If the search is successful, the Z39.50 interface returns a stream of data containing bibliographic records in *Pica+ format*. This data is written to the file '`records.txt`'.

5. **Clear database (only if parsing)**

   The function `ZClient::clear_database` causes the *Transact-SQL stored procedures* '`delete_tables`' and '`regenerate_tables`' to be executed in the `PICA` database. See Chapter 32 [ZTest; Database Stored Procedures], page 123.

6. **Parse records**

   The function `ZClient::parse_records` is called with the string `"records.txt"` as its argument. Parsing is somewhat complex. `ZClient::parse_records` is described in Section 20.2.7 [ZClient; Parsing], page 43.

7. **Display records**

   An object of type `DB_Display` called `db_display` is created. See Chapter 29 [Displaying Database Contents (DB_Display)], page 94. Then, the functions `DB_Display::open_html_file` and `DB_Display::display_records` are called on it. See Section 29.2.4 [DB_Display; Output], page 94. They cause HTML code to be written to the file whose name is passed as an argument to `DB_Display::open_html_file`, currently, `"records.html"`.

8. **Clean up and exit**

   Call `ZOOM_resultset_destroy` and `ZOOM_connection_destroy` to clean up. !! TODO: Document these functions! LDF 2006.09.11.

19 Dialog_Z_1

Class `Dialog_Z_1` is declared in '`dialogz1.web`'. It is derived from the MFC class `CDialog` using public derivation.

19.1 Dialog Box 1

Edit Box "Enter search string:"
: Contains a search query. The default is currently "`@and @and @attr 1=59 Clausthal @attr 1=1031 @attr 4=1 "Elektronische Ressource" @attr 1=31 @attr 2=4 @attr 4=4 2006`".

Group "Options"

- **Check Box** "Perform search"
- **Check Box** "Write records file"
- **Check Box** "Clear database (only if parsing)"
- **Check Box** "Parse Records"
- **Check Box** "Display Records"
  If checked, output is written to '`records.html`'.
- **Group** (Display Options)
  - **Radio Button** "All records"
  - **Radio Button** (Range)
    - **Edit Box** "From:"
      Record number at low end of range.
    - **Edit Box** "To:"
      Record number at high end of range.

Group "Search Options"

- **List Box** "Servers"
  List of servers to which connections should be made.

Button "OK"
: Press to perform actions.

Button "Cancel"
: ZTest exits.

19.2 Data Members

`CString search_command` [Public Variable]

`BOOL perform_search` [Public Variable]

`BOOL parse_records` [Public Variable]

`BOOL display_records` [Public Variable]

`BOOL write_records_file` [Public Variable]

19.3 Functions

19.3.1 Constructor

`void Dialog_Z_1 ([CWnd* pParent = NULL])` [Constructor]

19.3.2 Destructor

`void ~Dialog_Z_1 (void)` [Virtual Destructor]

19.3.3 Event Handlers

`BOOL OnInitDialog (void)` [Public Virtual Function]

`BOOL OnBnClickedPerformSearch (void)` [Public Virtual Function]

`BOOL OnBnClickedParseRecords (void)` [Public Virtual Function]

`BOOL OnBnClickedDisplayRecords (void)` [Public Virtual Function]

`BOOL OnBnClickedWriteRecordsFile (void)` [Public Virtual Function]

`BOOL OnBnClickedClearDatabase (void)` [Public Virtual Function]

`BOOL OnBnClickedDisplayAll (void)` [Public Virtual Function]

`void OnOK (void)` [Protected Virtual Function]

`void OnCancel (void)` [Protected Virtual Function]

19.3.4 Exchanging Data

`void DoDataExchange (CDataExchange* pDX)` [Protected Virtual Function]

20 ZClient

Class `ZClient` is declared in ‘`zclient.web`’. `Pica_Record` is `friend` of `ZClient`.

20.1 Data Members

20.1.1 Static Constants

`unsigned short MAX_ZOOM_CONNECTION` [Static constant]
Initialized to 16 in ‘`zclient.web`’.

`unsigned short MAX_ZOOM_RESULTSET` [Static constant]
Initialized to 16 in ‘`zclient.web`’.

`unsigned short GBV_GVK_ID` [Static constant]
Initialized to 1 in ‘`zclient.web`’.

20.1.2 Protected Variables

`vector<ZOOM_connection*> connections` [Protected variable]
`connections`.

`vector<ZOOM_resultset*> resultsets` [Protected variable]

`Category_Map_Type category_map` [Protected variable]

`Output_Stream_Type log_strm` [Protected variable]

`CSources source_recordset` [Protected variable]

`CDatabase* database` [Protected variable]

`unsigned short source_id` [Protected variable]

20.2 Functions

20.2.1 Constructors

`void ZClient (void)` [Default constructor]

`void ZClient (unsigned short` *`ssource_id`*, `char*` *`server_name`*, [`int` *`port_num`* = *0*]) [Constructor]

20.2.2 Destructor

`void ~ZClient (void)` [Destructor]

20.2.3 Initialization

`int init_category_map (void)` [Function]
Initializes `category_map`.

20.2.4 Connections

`ZOOM_connection* get_connection (void)` [Function]

`ZOOM_connection* get_last_connection (vector<ZOOM_connection*>::size_type` *`connection_ctr`*`)` [Function]

20.2.5 Resultsets

`ZOOM_resultset* get_resultset (void)` [Function]

`ZOOM_resultset* get_last_resultset (vector<ZOOM_resultset*>::size_type` *`resultset_ctr`*`)` [Function]

`size_t get_resultset_size (vector<ZOOM_resultset*>::size_type` *`resultset_ctr`*`)` [Function]

`size_t get_last_resultset_size (void)` [Function]

20.2.6 Searching

`int search_pqf (char*` *`curr_query_str`*`, [ZOOM_connection*` *`curr_connection`* `= 0])` [Function]

20.2.7 Parsing

`int parse_records (const char*` *`in_filename`*`)` [Public Functions]
`int write_field_data (string&` *`category_id`*`, string&` *`repeat_code`*`, char` *`field_id`*`, string&` *`field_value`*`, Pica_Record*` *`pica_record`*`)`

Together with `Pica_Record::write_to_database` and other functions mentioned below, they implement a *lexical scanner* a.k.a. a *low-level parser*. See Section 25.2.3 [Pica_Record; Writing to Database], page 69.

`parse_records` first opens the file whose name is passed as the *in_filename* argument for reading. Assuming this file exists, it then reads its contents character-by-character. It thus uses the "category code" approach to scanning, as in Donald Knuth's TEX and METAFONT packages, and in the author's GNU 3DLDF package. A more common approach to lexical scanning uses *regular expressions*. Lexical scanners using regular expressions are often programmed using the lex or Flex packages.

Assuming a successful search, the input file will consist of one or more *records* in Pica+ format. (If the search was not successful, it will contain an error message). A record consists of one or more non-blank lines. A blank line marks the end of a record. Each line begins with a *category id*, which consists of four characters. These may be letter, digits, or the commercial "at"-character: '`@`'. Typical category ids are "`001@`", "`021A`", or "`045G`".

A category id may be followed directly, i.e., with no intervening whitespace, by a *repeat code* consisting of a slash, followed by two digits. Typical category ids with repeat codes are "`029F/01`", "`045Q/02`", or "`028C/03`".

A single space character follows the repeat code, if if present, or the category id otherwise. The rest of the line is occupied by one or more *field ids* followed by their associated *field values* or *arguments*. A field id consists of the special character octal 237 (decimal 159) followed directly by a single letter or digit. Octal 237 isn't a seven-bit ASCII character, so there's no universal standard for representing it. It is displayed as "`$`" in the GBV online-catalogue. However, the dollar sign *is* represented by a seven-bit ASCII character, namely octal 44 (decimal 36), and this character may also occur in field values, so this convention is potentially confusing.

The field value is at this point simply treated as text. It is terminated by the next field id, if any, or the end of the line. The end of the line also indicates the end of the list of fields associated with the category id at the beginning of the line.

A given category id may be occur multiple times within a single record. Sometimes, subsequent occurrences are marked by using a repeat code, but not always. On the other hand, a repeat code doesn't always indicate multiple occurence of a category id within a record: Some category ids are always used with a particular repeat code, or one of a number of repeat codes. Within a line, a given field id may occur multiple times. There is no repeat code for fields.

`parse_records` can be in one of four *states*. It keeps track of this in an `unsigned short` variable called '`state`'. The values are constant `unsigned shorts` called '`OUTSIDE_RECORD`', '`COLLECTING_CATEGORY_ID`', and '`COLLECTING_FIELD`'.

Before it starts reading characters, '`state`' is '`OUTSIDE_RECORD`'. That is, the '`state`' variable has the value '`OUTSIDE_RECORD`'. It so happens that '`OUTSIDE_RECORD`' = 1, '`COLLECTING_CATEGORY_ID`' = 2, and '`COLLECTING_FIELD`' = 3, but the actual values don't matter, as long as they differ.

Characters are read within a *"while" loop*. Before the loop starts, the variable '`Pica_Record* curr_pica_record`' is declared and initialized to 0. That is, '`curr_pica_record`' is a pointer to a `Pica_Record` object, but starts out pointing to 0.

At the beginning of each iteration of the loop, a character is read from the input file. The actions performed in the loop are determined by what character it is, and the value of the '`state`' variable. This is controlled by a fairly complex *conditional construction* using '`if`', '`else if`', and '`else`'. Much of the complexity is to ensure that all characters are processed correctly under all circumstances, even if the structure of the input file differs from what we expect. However, parsing Pica+ records is usually straightforward.

1. **Skip Whitespace**
   There is normally no whitespace at the beginning of an input file, but if there is, it's skipped.
2. **Collect Category ID**
   A line always begins with a *category id*, so the first non-blank character at the beginning of the input file, or at the beginning of a line, is interpreted as the first character in a category id. '`state`' has the value '`OUTSIDE_RECORD`'; it is now set to '`COLLECTING_CATEGORY_ID`'. Memory is now allocated for a new '`Pica_Record`' object, and '`curr_pica_record`' is pointed at it. In addition, a counter called '`record_ctr`', that keeps track of the number of records we process, is incremented. It will now have the value 1.

   The current character ('`curr_char`') is placed onto the empty string '`curr_category_id`'. `parse_records` now continues to collect characters and add them to 'curr_category_id' until it reads a whitespace character. The category id is now complete. It should consist of four characters, optionally followed by a *repeat code*. A repeat code consists of a slash followed by two digits, e.g., /01, /02, etc.

If a repeat code is present, it is now removed from the category id and stored in the string 'curr_repeat_code'. 'state' is now set to 'COLLECTING_FIELD'.

3. **Collect Fields**
   The first character following the whitespace that follows a category id or a repeat code should be octal 237, which should be followed by a letter or digit. Each category has a set of valid fields associated with it. These are documented in the *cataloguing guidelines* (germ. *Katagorisierungsrichtlinien*) of the GBV: http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/inhalt.shtml Field ids are only unique with respect to a given category id! That is, the same field id may have a completely different interpretation with respect to two different category ids.

   An arbitrary number of characters, including whitespace, may following the field id. These characters comprise the *field value*. The field value is terminated by a new field id, or the end of the line.

4. **Write Field Data**
   When a field value has been collected, parse_records calls the function ZClient::write_field_data, passing the current category id ('curr_category_id'), the (possibly non-existent) repeat code ('curr_repeat_code'), the (possibly non-existent) previous field id ('prev_field_id'), the current field value ('curr_field_value'), and the pointer to the current Pica_Record object ('curr_pica_record') as arguments.

   write_field_data checks whether the current category is one which is handled for this particular ZClient object. Whether a ZClient object handles any given category is determined by the function ZClient::init_category_map. See Section 20.2.3 [ZClient; Initialization], page 42. Currently, there is only one version of this function, and it is designed for use with the Z39.50 server for the GVK database of the GBV. When ZTest is adapted for use with other servers, it may be possible to use init_category_map unchanged or it may have to be modified. Alternatively, other, similar functions may be defined for use with other servers.

   If a category is "unknown" to this instance of ZClient, the field id and field value we've collected is simply discarded. Otherwise, write_field_data checks to see whether a Category_Container object for the present combination of category id and repeat code has already been created and placed onto the Category_Multimap_Type categories data member of the current Pica_Record object. If so, it will be used for any further actions. Otherwise, it will create a new Category_Container object, if the current field id is also known.

   The next step is to check whether this is the case. If the field id is unknown to this instance of ZClient, no further action is taken, and write_field_data exits. Otherwise, a new Subcategory_Container object is created, and it together with the current field id are placed onto the Subcategory_Vector_Type subcategory_vector data member of the current Category_Container object.

(Subcategory_Vector_Type is a *type definition* for vector<pair<char, Subcategory_Container*> > Subcategory_Vector_Type). Please note that a new Subcategory_Vector_Type is *always* added to Category_Container::subcategory_vector; A given field often occurs more than once within a record line.

5. **Write Record Data**
   A record ends with a blank line, or an EOF (end-of-file), which also indicates that the current record is the last one. parse_records now calls the function Pica_Record::write_to_database on *curr_pica_record, i.e., the Pica_Record object referenced by the pointer curr_pica_record. At this point, the Category_Multimap_Type categories data member of *curr_pica_record contains all of the information for the categories and fields that parse_records has collected in the previous steps. Category_Multimap_Type is a *type definition* or "typedef" for multimap<string, Category_Container*>.

   A pointer to the current ZClient object is passed to Pica_Record::write_to_database, along with a pointer to a CDatabase object that references the PICA database. See Chapter 31 [Database Tables for ZTest], page 110, and a reference to Output_Stream_Type used for accessing a log file. write_to_database now iterates through the Category_Multimap_Type Pica_Records::categories data member, referencing each Category_Multimap_Type (i.e., pair<string, Category_Container*>) object on categories in turn. categories is a multimap rather than a map, because there can be multiple lines for the same category within a single record. It's convenient to use a multimap in ZClient::write_field_data, but in Pica_Record::write_to_database, the order of the categories doesn't matter, so it just iterates through them from beginning to end, as though it were a vector.

   write_to_database now extracts the category id from the current Category_Multimap_Type object, and looks it up in the Category_Map_Type category_map data member of our ZClient object. This is the map that's initialized by ZClient::init_category_map. If a Category_Map_Type (i.e., pair<string, Category_Container*>, just like Category_Multimap_Type) object correspoding to the current category id is found on ZClient::category_map, then write_to_database exits. Otherwise, it iterates through the pair<char, Subcategory_Container*> objects the Subcategory_Vector_Type subcategory_vector data member of the Category_Container objects referenced by the pointers to Category_Container in the current Category_Multimap_Type object. (Subcategory_Vector_Type is a type definition for vector<pair<char, Subcategory_Container*> >.)

   Now, write_to_database extracts the field id from the current <pair<char, Subcategory_Container*> > and looks it up in the Subcategory_Map_Type subcategory_map data member of the current reference Category_Container from the ZClient object. If no corresponding <pair<char, Subcategory_Container*> > is found, then write_to_database ignores this field and continues. Otherwise, it cycles through the vector<Database_Command*> database_commands data member of the Subcategory_Container pointed to by the Subcategory_Container pointer in the pair returned. If

Subcategory_Container::database_commands isn't empty, it will contain pointers to objects of type Database_Command. A Database_Command object contains a data member called 'function', which is a pointer to a function taking the following arguments:

1. CDatabase* *database*
2. long *record_id*
3. Category_Container* *category*
4. Subcategory_Container* *subcategory*
5. Output_Stream_Type& *log_strm*

When write_to_database cycles through the database_commands vector, it calls the function referenced by the 'function' data member belonging to each Database_Command object. These Database_Command objects are on the reference Subcategory_Container object, which is on the reference Category_Container object, which, in turn, is on the ZClient object. In other words, they have no connection with the current Pica_Record object that contains the Category_Container objects that contain the Subcategory_Container objects that contain the field ids and values that we've collected from the records in the input file. Therefore, pointers to the current Category_Container and Subcategory_Container from the current Pica_Record are passed to 'function' as arguments.

These functions are all static member functions of Subcategory_Container and have names corresponding to the Pica+ category and field ids, e.g., 'f_001A_0', 'f_011_AT_b', 'f_047I_a', etc. They all have the same return type and arguments (otherwise it wouldn't be possible to refer to them all using Database_Command::function). However, they fall into two different types, according to the way they function. Where the information in a field value conceptually "stands alone" and can be processed without reference to the values of other fields, the function can simply cause the appropriate Transact-SQL code to be executed in the database. Sometimes, however, fields are related, and the way they're processed depends on their order, what other fields are present, or other conditions. In this case, the current field id and value are pushed onto the vector<pair<char, string> > database_command_arguments data member of the current Category_Container to be processed in the next step.

Once all the Subcategory_Container objects for the current Category_Container object have been processed, write_to_database checks the database_commands vector on the current reference Category_Container object. Since many fields are processed by the corresponding Subcategory_Container functions alone, not every category requires database commands for the Category_Container. Some do, however, and these have non-empty database_command and database_command_arguments vectors. The latter contain the field ids and values from the Subcategory_Container objects from the current Category_Container object's subcategory_vector data member, as explained above. The Database_Command objects referenced by the pointers on Category_Container::database_commands are just like the ones on Subcategory_Container::database_commands, except that 'function' points to a static member function of Category_Container. These functions are similar to the field-handling functions of Subcategory_Container, except that their names begin with 'F_' and are otherwise derived from the category ids alone, e.g., F_001B, F_041A, F_209A, etc. They take the same arguments as the Subcategory_Container::f_* functions, but the Subcategory_Container* subcategory argument is always 0 when they're called. Declaring them to have the same arguments makes it possible to use the Database_Command type both for Category_Container and Subcategory_Container. The Category_Container::f_* functions process the field data stored in Category_Container::database_command_arguments and cause the appropriate Transact-SQL commands to be executed in the database, like the Subcategory_Container::f_* functions.

Category_Container::database_commands and Subcategory_Container::database_commands are vectors, so that any number of functions could be called for a given Category_Container or Subcategory_Container. However, currently, only one function is ever called for each Subcategory_Container, and either no functions, or only one function is ever called for each Category_Container.

20.2.8 Clearing the Database

`int clear_database (void)` [Public Function]

21 Utility Types

21.1 Output_Stream_Type

Struct `Zutput_Stream_Type` is declared in ‘`opstrmtp.web`’.

21.2 Data Members

`ofstream output_stream` [Public variable]

`CMutex mutex` [Public variable]

21.3 Functions

`int lock (void)` [Function]

`int unlock (void)` [Function]

22 Pica Formats and Records

The term "Pica" refers to a pair of formats used for library records, namely "Pica3" and "Pica+". Pica is commonly used in Germany, in particular, by the *Gemeinsamer Bibliotheksverbund* (Common Library Network) or GBV.

!! TODO: Write about Erfassungspraxis/Cataloguing practice Pica3 800 vs. Pica+ 5100–5199

22.1 Parsing

22.2 Categories and Fields

22.2.1 Repeat Codes for Categories

Where consecutive lines belong to the same category, *repeat codes* may be used, but aren't always. Used in this way, any category code can be followed by a repeat code.[1]

On the other hand, specific repeat codes change the meaning of the associated category. In this case, the Pica+ category code may correspond to different Pica3 categories, depending on the repeat code.

!! TODO: Add example.

Currently, in ZTest, an object of `class Category_Container` is created for each line in a *Pica record*, *unless* two or more lines within a group of lines of the same category either have no repeat code, or share the same one. The field codes and their associated values are stored in the same `class Category_Container` and they are processed as though they had all occurred on a single line. It would be possible to change this behavior, and force lines of fulfilling these criteria to be handled separately. Alternatively, it would be possible to continue to use a single `Category_Container`, but to handle the groups of fields belonging to each lines separately. One way of implementing doing this would be to push an ASCII null character (`'\0'`) onto the list of field codes. This character isn't used as a Pica field code.

!! TODO: Add example and cross references.

!! TODO: Explain about

22.2.2 Repeated Fields

!! TODO: Note the use of different field codes for additional items, as in Pica+ 041A/Pica3 5100–5199. Note, too, that in this case, I've found that '`a`' wasn't used, and that '`f`' was repeated. See Section 23.7 [041A Subject], page 55, and Section 24.1 [800; 5100–5199 Subject], page 60.

22.3 Displaying

[1] I believe this to be true, but there may be categories for which repeat codes don't make sense, or for which the cataloguing guidelines (*Katalogisierungsrichtlinien*) forbid their use.

23 Pica+ Codes

23.1 001A Identifier and Date of the Original Catalogue Entry

Pica+ Code:
001A

Pica3 Code:
0200
See Section 24.2 [0200 Pica3 Codes], page 62.

German: Kennung und Datum der Ersterfassung

English: Identifier and Date of the Original Catalogue Entry

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/0200.pdf`

Database Tables:
`Records`. See Section 31.5 [ZTest; Database Tables; Records], page 111.

No `Category_Container` Functions.

`Subcategory_Container` Function:
`f_001A_0`. See Section 27.3.4.1 [Identifier and Date of the Original Catalogue Entry], page 77.

23.1.1 001A Fields

Pica+	**Pica3**	**German**	**English**
0	(Not present)	Kennung und Datum der Ersterfassung Position 01-04: Erfassungskennung, Position 05-13: Datum (:TT-MM-JJ)	Identifier and Date of the Original Catalogue Entry Position 01-04: Cataloguing identifier, Position 05-13: Date (:DD-MM-YY)

23.2 009P Remote Access to Electronic Resources

!! TODO: Write about this. Add corresponding section to '`pica3cds.texi`'.

This is the first category for which I place `Database_Command` objects on the `curr_category_container` object and the `curr_subcategory_container` objects, where the '`function`' members point to functions belonging to another categories, namely Pica+ 209R, Pica3 7133 "Lokale Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff", "Local information regarding remote access to electronic resources". See Section 23.11 [209R Local information regarding remote access to electronic resources], page 59, and Section 24.9 [7133 Local information regarding remote access to electronic resources], page 66.

The fields are the same, and the data is written to the same tables, i.e., `Remote_Access` and `Records_Remote_Access`. See Section 31.24 [Remote_Access], page 117, and Section 31.25 [Records_Remote_Access], page 118.

23.3 010@ Code(s) for Languages

Pica+ Code:
: 010@

Pica3 Code:
: 1500
: See Section 24.4 [1500 Code(s) for Languages], page 63.

German: Code(s) für Sprachen

English: Code(s) for Languages

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/1500.pdf`

Database Tables:
: `Languages`, `Records_Languages`. See Section 31.20 [Languages], page 116, and See Section 31.21 [Records_Languages], page 116.

`Subcategory_Container` Functions:
: `Subcategory_Container::f_010@_a`. See Section 27.3.4.3 [Code(s) for Languages], page 79.

23.3.1 010@ Fields

Pica+	Pica3	German	English
a	/1	1. Sprachencode für den vorliegenden Text	First language code for the present text
a	/1	jeder weitere Sprachencode für den vorliegenden Text	each subsequent language code for the present text
b	/2	1. Sprachencode der Intermediärsprache	First language code for the intermediary language
b	/2	jeder weitere Sprachencode für Intermediärsprachen	Each subsequent language code for the intermediary language
c	/3	1. Sprachencode für die Originalsprache	First language code for the original language
c	/3	jeder weitere Sprachencode für Originalsprachen	Each subsequent language code for the original language

In data from external sources, the category Pica+ 010@/Pica3 1500 can contain additional fields.

Pica+	**Pica3**	**German**	**English**
d	/4	1. Sprachencode für die Zusammenfassung	First language code for the abstract
d	/4	jeder weitere Sprachencode für die Zusammenfassung	Each subsequent language code for the abstract

Currently, ZTest only processes field '`a`'.

23.4 011@ Year of Appearance

Pica+ Code:
: `011@`

Pica3 Code:
: 1100
: See ⟨undefined⟩ [1100 Year of Appearance], page ⟨undefined⟩.

German: Erscheinungsjahr

English: Year of Appearance

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/1100.pdf`

Database Tables:
: `Records`. See Section 31.5 [Records], page 111.

`Subcategory_Container` Functions:
: `f_011_AT_a`, `f_011_AT_b`, `f_011_AT_e`, `f_011_AT_n`, and `year_appearance_func`. See Section 27.3.4.4 [Year of Appearance], page 80.

23.4.1 011@ Fields

23.5 033A Place, Publisher

Pica+ Code:
: 033A

Pica3 Code:
: 4030
: See Section 24.5 [4030 Pica3 Codes], page 64.

German: Ort, Verlag

English: Place, Publisher

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/4030.pdf`

Database Tables:
: `Publishers`, `Records_Publishers` See Section 31.16 [Publishers], page 115, and See Section 31.17 [Records_Publishers], page 115.

`Category_Container` Functions:
: `Category_Container::F_033A`, `Category_Container::publishers_database_providers_func`. See Section 26.3.3.4 [Place of Publication; Publisher], page 71.

`Subcategory_Container` Functions:
: `Subcategory_Container::f_033A_n`, `Subcategory_Container::f_033A_p`. See Section 27.3.4.9 [Place of Publication; Publisher], page 84.

23.5.1 033A Fields

Pica+	**Pica3**	**German**	**English**
n	¬:¬	Verlag	Publisher
p	(None)	Ort	Place

23.6 034D Size or Range; Specification of Material; Technical System

Pica+ Code:
: 034D

Pica3 Code:
: 4060
: See Section 24.6 [4060 Pica3 Codes], page 65.

German: Umfangsangabe, spezifische Materialbenennung, technisches System

English: Size or Range, Specification of Material, Technical System

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/4060.pdf`

Database Tables:
: `Physical_Descriptions`, `Records_Physical_Descriptions`. See Section 31.22 [Physical_Descriptions], page 116, and Section 31.23 [Records_Physical_Descriptions], page 117.

`Subcategory_Container` Function:
: `Subcategory_Container::f_034D_a`. See Section 27.3.4.10 [Physical Description], page 85.

23.6.1 034D Fields

Pica+	**Pica3**	**German**	**English**
a	(Not present)	Text	Text

23.7 041A Subject

Pica+ Code:
041A

Pica3 Codes:
800; 5100–5199
See Section 24.1 [800; 5100–5199 Pica3 Codes], page 60.

German: Hauptschlagwort und Unterschlagwoerter (Schlagwortansetzung) (Pica3 800)
RSWK-Ketten (Pica3 5100–5199)

English: Main Subject and Subsidiary Subjects (Subject Assignment) (Pica3 800)
RSWK Chains (Pica3 5100–5199)

Cataloguing Guidelines:
Pica3 800: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/norm/800.pdf`
Pica3 5100–5199: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/5100.pdf`

Database Tables:
`Subjects`. See Section 31.31 [Subjects], page 121.
`Records_Subjects`. See Section 31.32 [Records_Subjects], page 121.
`Permutation_Patterns`. See Section 31.33 [Permutation_Patterns], page 121.

`Category_Container` Functions:
`F_041A` and `sub_F_041A`.
See Section 26.3.3.5 [Subject], page 72.

`Subcategory_Container` Functions:
`f_041A_9`
`f_041A_a`
`f_041A_f`
`f_041A_S`.
See Section 27.3.4.11 [Subject], page 86.

23.7.1 041A Fields

Pica3 800

Pica+	**Pica3**	**German**	**English**
a	(Not Present)	Haupt- bzw. Unterschlagwort	Main or Subsidiary Subject
S	`|...|¬`	Indikator	Indicator

The following table shows the values that the “Indikator” (“Indicator”) field can contain. Some are only used with Pica3 5100–5199.

Value	**German**	**English**	**Pica3 800**

c	Körperschaftsschlagwort (Ort)	Entity Subject (Location)	Yes
f	Formschlagwort	Form Subject	No
g	Geografisches/ethnografisches Schlagwort	Geographical/Ethnographical Subject	Yes
p	Personenschlagwort	Personal Subject	Yes
s	Sachschlagwort	Material Subject	Yes
t	Titelschlagwort (800)/Werktitel (51xx)	Title Subject (800)/Work Title (51xx)	Yes
k	Körperschaftsschlagwort (Name)	Entity Subject (Name)	Yes
z	Zeitschlagwort	Temporal Subject	No

Pica3 5100–5199

Please note that the field '`a`' is ambiguous, and can have three different meanings.

Field Set 1:

<table>
<tr><th>Pica+</th><th>Pica3</th><th>German</th><th>English</th></tr>
<tr><td>9</td><td>!...!</td><td>Identifikationsnummer (PPN)</td><td>Identification Number (PPN)</td></tr>
<tr><td>a</td><td>(Not Present)</td><td>Schlagwort</td><td>Subject</td></tr>
<tr><td>S</td><td>|...|</td><td>Schlagwortindikator</td><td>Subject Indicator</td></tr>
</table>

Field Set 2

Pica+	Pica3	German	English
f	⟨...⟩	Permutationsmuster	Permutation Pattern
a	(Not Present)	Zweites und weiteres Permutationsmuster	Second and additional permutation pattern

Field Set 3

Pica+	Pica3	German	English
a	(Not Present)	Angaben zur Schlagwortkette	Subject Chain Information

23.8 203@ Exemplar Production Number

Pica+ Code:
: 203@

Pica3 Code:
: 7800
: See Section 24.10 [7800 Exemplar Production Number], page 67.

German: Exemplar-Produktionsnummer

English: Exemplar Production Number

Cataloguing Guidelines:
: Not present.

Database Table:
: `Exemplar_Production_Numbers`. See Section 31.26 [Exemplar_Production_Numbers], page 118.

`Subcategory_Container` Functions:
: `Subcategory_Container::f_203@_S`. See Section 27.3.4.13 [Exemplar Production Number], page 87.

23.8.1 203@ Fields

Pica+	**Pica3**	**German**	**English**
0	(Not present)	Exemplar-Produktionsnummer	Exemplar Production Number

23.9 209A Call Number

Pica+ Code:
: 209A

Pica3 Codes:
: 7100–7109
: See Section 24.8 [7100–7109 Call Number], page 65.

German: Signatur

English: Call Number

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7100.pdf`

Database Table:
: `Call_Numbers`. See Section 31.27 [Call_Numbers], page 119.

`Category_Container` Functions:
: `Category_Container::F_209A` and `Category_Container::sub_F_209A`. See Section 26.3.3.6 [Call Number], page 73.

Subcategory_Container Functions:
Subcategory_Container::f_209A_a, Subcategory_Container::f_209A_b, Subcategory_Container::f_209A_f, and Subcategory_Container::f_209A_j. See Section 27.3.4.14 [Call Number], page 88.

23.9.1 209A Fields

Pica+	Pica3	German	English
a	(Not present)	Signatur	Call Number
b, j	.../...#	Bibliotheksnummer / Abteilung der Bibliothek	Library Number / Library Department
f	!...!	Sonderstandort	Special Location

The Pica+ category 209A and the corresponding Pica3 categories 7100–7109 have additional fields, which ZTest currently doesn't use. See the cataloguing guidelines for more information: http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7100.pdf

23.10 209C Access Number

Pica+ Code:
209C

Pica3 Codes:
8100
See Section 24.11 [8100 Access Number], page 68.

German: Zugangsnummer

English: Access Number

Cataloguing Guidelines:
http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/8100.pdf

Database Table:
Access_Numbers. See Section 31.29 [Access_Numbers], page 119.

Subcategory_Container Function:
Subcategory_Container::f_209C_a. See Section 27.3.4.15 [Access Number], page 90.

23.10.1 209C Fields

Pica+	Pica3	German	English
a	(Not present)	Zugangsnummer	Access Number

23.11 209R Local information regarding remote access to electronic resources

Pica+ Code:
209R

Pica3 Code:
7133
See Section 24.9 [7133 Local Information Regarding Remote Access to Electronic Resources], page 66.

German: Lokale Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff

English: Local information regarding remote access to electronic resources

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7133.pdf`

Database Table:
`Remote_Access`. See Section 31.24 [Remote_Access], page 117.

`Category_Container` Functions:
`Category_Container::F_209R` and `Category_Container::sub_F_209R`.
See ⟨undefined⟩ [Functions for Specific Categories], page ⟨undefined⟩.

`Subcategory_Container` Functions:
`Subcategory_Container::f_209R_S`. See Section 27.3.4.16 [Local information regarding remote access to electronic resources], page 90.

23.11.1 209R Fields

The fields for Pica+ 209R/Pica3 7133 are the same as for Pica3 408x: "Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff", "Information regarding remote to electronic resources". "408x" currently refers to the follow Pica3 categories: 4083, 4085, 4086, 4087, 4088, and 4089.

See the cataloguing guidelines for Pica3 408x/Pica+ 009P for more information: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/408x.pdf`.

Pica+	Pica3	German	English
0		Format	Format
a	¬ =A¬	URL (Universal Resource Locator)	URL (Universal Resource Locator)
g	¬ =G¬	URN (Universal Resource Name)	URN (Universal Resource Name)
S	⟨ ⟩	Lizenzindikator	License indicator
x	¬ =X¬	Interne Bemerkungen	Internal Remarks
y	¬[. . .]	Text für die Web-Anzeige	Text for Web Display

24 Pica3 Codes

24.1 800; 5100–5199 Subject

Pica3 Codes:
800
5100–5199

Pica+ Code:
041A
See Section 23.7 [041A Pica+ Codes], page 55.

German: Hauptschlagwort und Unterschlagwoerter (Schlagwortansetzung) (Pica3 800)
RSWK-Ketten (Pica3 5100–5199)

English: Main Subject and Subsidiary Subjects (Subject Assignment) (Pica3 800)
RSWK Chains (Pica3 5100–5199)

Cataloguing Guidelines:
Pica3 800: http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/norm/800.pdf
Pica3 5100–5199: http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/5100.pdf

Database Tables:
`Subjects`. See Section 31.31 [Subjects], page 121.
`Records_Subjects`. See Section 31.32 [Records_Subjects], page 121.
`Permutation_Patterns`. See Section 31.33 [Permutation_Patterns], page 121.

`Category_Container` Functions:
`F_041A` and `sub_F_041A`.
See Section 26.3.3.5 [Subject], page 72.

`Subcategory_Container` Functions:
`f_041A_9`
`f_041A_a`
`f_041A_f`
`f_041A_S`.
See Section 27.3.4.11 [Subject], page 86.

24.1.1 800; 5100–5199 Fields

Pica3 800

Pica+	**Pica3**	**German**	**English**
a	(Not Present)	Haupt- bzw. Unterschlagwort	Main or Subsidiary Subject
S	`|...|¬`	Indikator	Indicator

The following table shows the values that the “Indikator” (“Indicator”) field can contain. Some are only used with Pica3 5100–5199.

Value	German	English	Pica3 800
c	Körperschaftsschlagwort (Ort)	Entity Subject (Location)	Yes
f	Formschlagwort	Form Subject	No
g	Geografisches/ethnografisches Schlagwort	Geographical/Ethnographical Subject	Yes
p	Personenschlagwort	Personal Subject	Yes
s	Sachschlagwort	Material Subject	Yes
t	Titelschlagwort (800)/Werktitel (51xx)	Title Subject (800)/Work Title (51xx)	Yes
k	Körperschaftsschlagwort (Name)	Entity Subject (Name)	Yes
z	Zeitschlagwort	Temporal Subject	No

Pica3 5100–5199

Please note that the field ‘`a`’ is ambiguous, and can have three different meanings.

Field Set 1:

Pica+	Pica3	German	English
9	!...!	Identifikationsnummer (PPN)	Identification Number (PPN)
a	(Not Present)	Schlagwort	Subject
S	`|...|`	Schlagwortindikator	Subject Indicator

Field Set 2

Pica+	Pica3	German	English
f	⟨...⟩	Permutationsmuster	Permutation Pattern
a	(Not Present)	Zweites und weiteres Permutationsmuster	Second and additional permutation pattern

Field Set 3

Pica+	Pica3	German	English
a	(Not Present)	Angaben zur Schlagwortkette	Subject Chain Information

24.2 0200 Identifier and Date of the Original Catalogue Entry

Pica+ Code:
: 001A

Pica3 Code:
: 0200
: See Section 24.2 [0200 Pica3 Codes], page 62.

German: Kennung und Datum der Ersterfassung

English: Identifier and Date of the Original Catalogue Entry

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/0200.pdf`

Database Tables:
: `Records`. See Section 31.5 [ZTest; Database Tables; Records], page 111.

No `Category_Container` Functions.

`Subcategory_Container` Function:
: `f_001A_0`. See Section 27.3.4.1 [Identifier and Date of the Original Catalogue Entry], page 77.

24.2.1 0200 Fields

Pica+	Pica3	German	English
0	(Not present)	Kennung und Datum der Ersterfassung Position 01-04: Erfassungskennung, Position 05-13: Datum (:TT-MM-JJ)	Identifier and Date of the Original Catalogue Entry Position 01-04: Cataloguing identifier, Position 05-13: Date (:DD-MM-YY)

24.3 1100 Year of Appearance

Pica3 Code:
: 1100

Pica+ Code:
: `011@`
: See ⟨undefined⟩ [011@ Year of Appearance], page ⟨undefined⟩.

German: Erscheinungsjahr

English: Year of Appearance

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/1100.pdf`

Database Tables:
`Records`. See Section 31.5 [Records], page 111.

`Subcategory_Container` Functions:
`f_011_AT_a`, `f_011_AT_b`, `f_011_AT_e`, `f_011_AT_n`, and `year_appearance_func`. See Section 27.3.4.4 [Year of Appearance], page 80.

24.3.1 1100 Fields

24.4 1500 Code(s) for Languages

Pica3 Code:
1500

Pica+ Code:
010@
See Section 23.3 [010@ Code(s) for Languages], page 52.

German: Code(s) für Sprachen

English: Code(s) for Languages

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/1500.pdf`

Database Tables:
`Languages`, `Records_Languages`. See Section 31.20 [Languages], page 116, and See Section 31.21 [Records_Languages], page 116.

`Subcategory_Container` Functions:
`Subcategory_Container::f_010@_a`. See Section 27.3.4.3 [Code(s) for Languages], page 79.

24.4.1 1500 Fields

Pica+	**Pica3**	**German**	**English**
a	/1	1. Sprachencode für den vorliegenden Text	First language code for the present text
a	/1	jeder weitere Sprachencode für den vorliegenden Text	each subsequent language code for the present text
b	/2	1. Sprachencode der Intermediärsprache	First language code for the intermediary language
b	/2	jeder weitere Sprachencode für Intermediärsprachen	Each subsequent language code for the intermediary language

c	/3	1. Sprachencode für die Originalsprache	First language code for the original language
c	/3	jeder weitere Sprachencode für Originalsprachen	Each subsequent language code for the original language

In data from external sources, the category Pica+ 010@/Pica3 1500 can contain additional fields.

Pica+	Pica3	German	English
d	/4	1. Sprachencode für die Zusammenfassung	First language code for the abstract
d	/4	jeder weitere Sprachencode für die Zusammenfassung	Each subsequent language code for the abstract

Currently, ZTest only processes field 'a'.

24.5 4030 Place, Publisher

Pica3 Code:
: 4030

Pica+ Code:
: 033A
: See Section 23.5 [033A Pica+ Codes], page 53.

German: Ort, Verlag

English: Place, Publisher

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/4030.pdf`

Database Tables:
: `Publishers`, `Records_Publishers` See Section 31.16 [Publishers], page 115, and See Section 31.17 [Records_Publishers], page 115.

`Category_Container` Functions:
: `Category_Container::F_033A`, `Category_Container::publishers_database_providers_func`. See Section 26.3.3.4 [Place of Publication; Publisher], page 71.

`Subcategory_Container` Functions:
: `Subcategory_Container::f_033A_n`, `Subcategory_Container::f_033A_p`. See Section 27.3.4.9 [Place of Publication; Publisher], page 84.

24.5.1 4030 Fields

Pica+	Pica3	German	English

n	¬:¬	Verlag	Publisher
p	(None)	Ort	Place

24.6 4060 Size or Range; Specification of Material; Technical System

Pica3 Code:
4060

Pica+ Code:
034D
See Section 23.6 [034D Pica+ Codes], page 54.

German: Umfangsangabe, spezifische Materialbenennung, technisches System

English: Size or Range, Specification of Material, Technical System

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/4060.pdf`

Database Tables:
`Physical_Descriptions`, `Records_Physical_Descriptions`. See Section 31.22 [Physical_Descriptions], page 116, and Section 31.23 [Records_Physical_Descriptions], page 117.

`Subcategory_Container` Function:
`Subcategory_Container::f_034D_a`. See Section 27.3.4.10 [Physical Description], page 85.

24.6.1 4060 Fields

Pica+	**Pica3**	**German**	**English**
a	(Not present)	Text	Text

24.7 5100–5199 Subject

See Section 24.1 [800; 5100–5199 Subject], page 60.

24.8 7100–7109 Call Number

Pica3 Codes:
7100–7109

Pica+ Code:
209A
See Section 23.9 [209A Call Number], page 57.

German: Signatur

English: Call Number

Cataloguing Guidelines:
: http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7100.pdf

Database Table:
: Call_Numbers. See Section 31.27 [Call_Numbers], page 119.

Category_Container Functions:
: Category_Container::F_209A and Category_Container::sub_F_209A. See Section 26.3.3.6 [Call Number], page 73.

Subcategory_Container Functions:
: Subcategory_Container::f_209A_a, Subcategory_Container::f_209A_b, Subcategory_Container::f_209A_f, and Subcategory_Container::f_209A_j. See Section 27.3.4.14 [Call Number], page 88.

24.8.1 7100–7109 Fields

Pica+	**Pica3**	**German**	**English**
a	(Not present)	Signatur	Call Number
b, j	.../...#	Bibliotheksnummer / Abteilung der Bibliothek	Library Number / Library Department
f	!...!	Sonderstandort	Special Location

The Pica+ category 209A and the corresponding Pica3 categories 7100–7109 have additional fields, which ZTest currently doesn't use. See the cataloguing guidelines for more information: http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7100.pdf

24.9 7133 Local information regarding remote access to electronic resources

Pica3 Code:
: 7133

Pica+ Code:
: 209R
: See Section 23.11 [209R Local Information Regarding Remote Access to Electronic Resources], page 59.

German: Lokale Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff

English: Local information regarding remote access to electronic resources

Cataloguing Guidelines:
: http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7133.pdf

Database Table:
: Remote_Access. See Section 31.24 [Remote_Access], page 117.

`Category_Container` Functions:
: `Category_Container::F_209R` and `Category_Container::sub_F_209R`. See ⟨undefined⟩ [Functions for Specific Categories], page ⟨undefined⟩.

`Subcategory_Container` Functions:
: `Subcategory_Container::f_209R_S`. See Section 27.3.4.16 [Local information regarding remote access to electronic resources], page 90.

24.9.1 7133 Fields

The fields for Pica+ 209R/Pica3 7133 are the same as for Pica3 408x: "Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff", "Information regarding remote to electronic resources". "408x" currently refers to the follow Pica3 categories: 4083, 4085, 4086, 4087, 4088, and 4089.

See the cataloguing guidelines for Pica3 408x/Pica+ 009P for more information: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/408x.pdf`.

Pica+	**Pica3**	**German**	**English**
0		Format	Format
a	¬ =A¬	URL (Universal Resource Locator)	URL (Universal Resource Locator)
g	¬ =G¬	URN (Universal Resource Name)	URN (Universal Resource Name)
S	⟨ ⟩	Lizenzindikator	License indicator
x	¬ =X¬	Interne Bemerkungen	Internal Remarks
y	¬[. . .]	Text für die Web-Anzeige	Text for Web Display

24.10 7800 Exemplar Production Number

Pica3 Code:
: 7800

Pica+ Code:
: 203@
: See Section 23.8 [203@ Exemplar Production Number], page 57.

German:
: Exemplar-Produktionsnummer

English:
: Exemplar Production Number

Cataloguing Guidelines:
: Not present.

Database Table:
: `Exemplar_Production_Numbers`. See Section 31.26 [Exemplar_Production_Numbers], page 118.

`Subcategory_Container` Functions:
`Subcategory_Container::f_203@_S`. See Section 27.3.4.13 [Exemplar Production Number], page 87.

24.10.1 7800 Fields

Pica+	**Pica3**	**German**	**English**
0	(Not present)	Exemplar-Produktionsnummer	Exemplar Production Number

24.11 8100 Access Number

Pica3 Codes:
8100

Pica+ Code:
209C
See Section 23.10 [209C Access Number], page 58.

German: Zugangsnummer

English: Access Number

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/8100.pdf`

Database Table:
`Access_Numbers`. See Section 31.29 [Access_Numbers], page 119.

`Subcategory_Container` Function:
`Subcategory_Container::f_209C_a`. See Section 27.3.4.15 [Access Number], page 90.

24.11.1 8100 Fields

Pica+	**Pica3**	**German**	**English**
a	(Not present)	Zugangsnummer	Access Number

25 Pica_Record

Class `Pica_Record` is declared in '`picarcrd.web`'. `class ZClient` is a `friend` of `Pica_Record`.

25.1 Data Members

`Category_Multimap_Type categories` [Protected variable]

25.2 Functions

25.2.1 Constructor

`void Pica_Record` (*void*) [Default Constructor]

25.2.2 Destructor

`void ~Pica_Record` (*void*) [Destructor]

25.2.3 Writing to Database

`int write_to_database` (`ZClient*` *zclient*, `CDatabase*` *database*, `Output_Stream_Type&` *log_strm*) [Function]

25.2.4 Showing

`int show` (`stringstream&` *local_strm*, [`string` *s* = `""`]) [Function]

26 Category_Container

Class Category_Container is declared in 'ctgcntnr.web'. The classes Pica_Record, ZClient, and Subcategory_Container are friends of Category_Container.

26.1 Local Static Variables

These static variables are local to 'ctgcntnr.web'. They are not members of class Category_Container.

long* subject_id [Static variable]
    Initialized to 0. Used in Category_Container::F_041A and Category_Container::sub_F_041A. See Section 26.3.3.5 [Subject], page 72.

long* subject_id_start [Static variable]
    Initialized to 0. Used in Category_Container::F_041A and Category_Container::sub_F_041A. See Section 26.3.3.5 [Subject], page 72.

unsigned short previous_repeat_code_ctr [Static variable]
    Initialized to 0. Used in Category_Container::F_041A and Category_Container::sub_F_041A. See Section 26.3.3.5 [Subject], page 72.

26.2 Data Members

string pica_plus_category_id [Protected variable]

string pica_3_category_id [Protected variable]

string content_description_english [Protected variable]

string content_description_german [Protected variable]

string repeat_code [Protected variable]

Subcategory_Map_Type subcategory_map [Protected variable]

Subcategory_Vector_Type subcategory_vector [Protected variable]

vector<Database_Command*> database_commands [Protected variable]

vector<pair<char, string> > database_command_arguments [Protected variable]

26.3 Functions

26.3.1 Assignment

void operator= (const Category_Container& *c*) [Assignment Operator]

26.3.2 Showing

int show (stringstream& *local_strm*, [string *prefix* = ""]) [Function]

26.3.3 Category Functions

26.3.3.1 Functions for Specific Categories

int F_001B (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

int F_021A (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

int F_028A (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

26.3.3.2 Second and Additional Authors

int F_028B (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

26.3.3.3 Other Contributing Persons

int F_028C (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

26.3.3.4 Place of Publication; Publisher

int F_033A (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

Pica+ Code:
: 033A
See Section 23.5 [033A Pica+ Codes], page 53.

Pica3 Code:
: 4030
See Section 24.5 [4030 Pica3 Codes], page 64.

German: Ort, Verlag

English: Place, Publisher

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/4030.pdf`

Database Tables:
`Publishers`, `Records_Publishers` See Section 31.16 [Publishers], page 115, and See Section 31.17 [Records_Publishers], page 115.

`Subcategory_Container` Functions:
`Subcategory_Container::f_033A_n`, `Subcategory_Container::f_033A_p`. See Section 27.3.4.9 [Place of Publication; Publisher], page 84.

See also Section 23.5.1 [033A Fields], page 54, or Section 24.5.1 [4030 Fields], page 64.

`int publishers_database_providers_func (CDatabase* database, long record_id, string column_str, string place_str, bool primary_switch, bool table_switch, Output_Stream_Type& log_strm)` [Static Function]

Called in `Category_Container::F_033A` and `Category_Container::F_033B` (which doesn't yet exist).

26.3.3.5 Subject

`int F_041A (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ Code:
041A
See Section 23.7 [041A Pica+ Codes], page 55.

Pica3 Codes:
800; 5100–5199
See Section 24.1 [800; 5100–5199 Pica3 Codes], page 60.

German: Hauptschlagwort und Unterschlagwoerter (Schlagwortansetzung) (Pica3 800)
RSWK-Ketten (Pica3 5100–5199)

English: Main Subject and Subsidiary Subjects (Subject Assignment) (Pica3 800)
RSWK Chains (Pica3 5100–5199)

Cataloguing Guidelines:

Pica3 800: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/norm/800.pdf`
Pica3 5100–5199: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/5100.pdf`

Database Tables:

`Subjects`. See Section 31.31 [Subjects], page 121.
`Records_Subjects`. See Section 31.32 [Records_Subjects], page 121.
`Permutation_Patterns`. See Section 31.33 [Permutation_Patterns], page 121.

`Subcategory_Container` Functions:

`f_041A_9`
`f_041A_a`
`f_041A_f`
`f_041A_S`.
See Section 27.3.4.11 [Subject], page 86.

See also Section 23.7.1 [041A Fields], page 55, or Section 24.1.1 [800; 5100–5199 Fields], page 60.

`int sub_F_041A (CDatabase* database, long record_id, Category_Container* category, unsigned short repeat_code_ctr, char subject_type_char, string subject, long id_number_ppn, short chain_number, short chain_link_number, string chain_info, string permutation_pattern, bool permutation_switch, Output_Stream_Type& log_strm)` [Static Function]

This function is called in `F_041A`. See above.

26.3.3.6 Call Number

`int F_209A (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ Code:

209A
See Section 23.9 [209A Call Number], page 57.

Pica3 Codes:

7100–7109
See Section 24.8 [7100–7109 Call Number], page 65.

German: Signatur

English: Call Number

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7100.pdf`

Database Table:
`Call_Numbers`. See Section 31.27 [Call_Numbers], page 119.

`Subcategory_Container` Functions:
`Subcategory_Container::f_209A_a`, `Subcategory_Container::f_209A_b`, `Subcategory_Container::f_209A_f`, and `Subcategory_Container::f_209A_j`. See Section 27.3.4.14 [Call Number], page 88.

See also Section 23.9.1 [209A Fields], page 58, or Section 24.8.1 [7100–7109 Fields], page 66.

`int sub_F_209A (CDatabase* database, long record_id, Category_Container* category, string call_number_str, string library_number_str, string library_department_str, string special_location_str, long* call_number_id, Output_Stream_Type& log_strm)` [Static Function]

This function is called in `Category_Container::F_209A`.

26.3.3.7 Local Information Regarding Remote Access to Electronic Resources

`int F_209R (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ Code:
209R
See Section 23.11 [209R Local Information Regarding Remote Access to Electronic Resources], page 59.

Pica3 Code:
7133
See Section 24.9 [7133 Local Information Regarding Remote Access to Electronic Resources], page 66.

German: Lokale Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff

English: Local information regarding remote access to electronic resources

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7133.pdf`

Database Table:
`Remote_Access`. See Section 31.24 [Remote_Access], page 117.

`Subcategory_Container` Functions:
`Subcategory_Container::f_209R_S`. See Section 27.3.4.16 [Local information regarding remote access to electronic resources], page 90.

See also Section 23.11.1 [209R Fields], page 59, or Section 24.9.1 [7133 Fields], page 67.

```
int sub_F_209R (CDatabase* database,
        long record_id,
        Category_Container* category,
        string format_str,
        string URL_str,
        string URN_str,
        string license_str,
        string internal_str,
        string web_str,
        long* remote_access_id,
        Output_Stream_Type& log_strm)
```
[Static Function]

This function is called in `Category_Container::F_209R`.

26.3.3.8 Functions for Groups of Categories

int personal_names_category_func ([Static Function]
string *main_table_name*,
string *assoc_table_name*,
string *column_name*,
CDatabase* *database*,
long *record_id*,
Category_Container* *category*)

int sub_personal_names_category_func ([Static Function]
string *main_table_name*,
string *assoc_table_name*,
string *column_name*,
CDatabase* *database*,
long *record_id*,
Category_Container* *category*,
string *surname*,
string *given_name*,
string *prefix*,
unsigned long *id_number_ppn*,
long *table_ctr*,
Output_Stream_Type& *log_strm*)

int titles_category_func (string *main_table_name*, [Static Function]
string *assoc_table_name*, string *column_name*, CDatabase*
database, long *record_id*, Category_Container* *category*)

int sub_titles_category_func (string *main_table_name*, [Static Function]
string *assoc_table_name*,
string *column_name*,
CDatabase* *database*,
long *record_id*,
Category_Container* *category*,
long *table_ctr*,
Output_Stream_Type& *log_strm*,
[string* *main_canonical_title* = 0,
[string* *continuation_main_canonical_title* = 0,
[string* *additions_main* = 0,
[string* *continuation_additions_main* = 0,
[string* *authorship* = 0,
[string* *standard_text* = 0,
[string* *additional_creator_main* = 0,
[string* *parallel_canonical_title* = 0,
[string* *additions_parallel* = 0,
[string* *additional_creator_parallel* = 0]]]]]]]]]])

27 Subcategory_Container

Class Subcategory_Container is declared in 'sbctgcnt.web'. The classes Pica_Record and ZClient are friends of Subcategory_Container.

27.1 Type Definitions

Subcategory_Map_Type [Typedef]

Subcategory_Map_Type is a synonym for map<char, Subcategory_Container*>

Subcategory_Vector_Type [Typedef]

Subcategory_Vector_Type is a synonym for vector<pair<char, Subcategory_Container*> >.

27.2 Data Members

char pica_plus_field_id [Protected variable]

string content_description_german [Protected variable]

string content_description_english [Protected variable]

string field_value [Protected variable]

vector<Database_Command*> database_commands [Protected variable]

27.3 Functions

27.3.1 Assignment

void operator= (const Subcategory_Container& *s*) [Assignment Operator]

27.3.2 Showing

int show (stringstream& *local_strm*, [string *prefix* = ""]) [Function]

27.3.3 String Processing

int fix_string (const string& *in_string*, string& *out_string*) [Static Function]

27.3.4 Subcategory Functions

27.3.4.1 Identifier and Date of the Original Catalogue Entry

Pica+ Code:

001A

See Section 23.1 [001A Pica+ Codes], page 51.

Pica3 Code:

0200

See Section 24.2 [0200 Pica3 Codes], page 62.

German: Kennung und Datum der Ersterfassung

English: Identifier and Date of the Original Catalogue Entry

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/0200.pdf`

Database Tables:
`Records`. See Section 31.5 [ZTest; Database Tables; Records], page 111.

No `Category_Container` Functions.

See also Section 23.1.1 [001A Fields], page 51, or Section 24.2.1 [0200 Fields], page 62.

`int f_001A_0 (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ field:
0

Pica3 field:
No identifier.

English: Identifier and Date of the Original Catalogue Entry
Position 01-04: Cataloguing identifier,
Position 05-13: Date (:DD-MM-YY)

German: Kennung und Datum der Ersterfassung
Position 01-04: Erfassungskennung,
Position 05-13: Datum (:TT-MM-JJ)

27.3.4.2 Other Subcategory_Container Functions

`int f_001B_0 (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

`int f_001B_t (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

`int f_002_AT_0 (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

`int f_003_AT_0 (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

27.3.4.3 Code(s) for Languages

Pica+ Code:
: 010@
: See Section 23.3 [010@ Code(s) for Languages], page 52.

Pica3 Code:
: 1500
: See Section 24.4 [1500 Code(s) for Languages], page 63.

German: Code(s) für Sprachen

English: Code(s) for Languages

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/1500.pdf`

Database Tables:
: `Languages`, `Records_Languages`. See Section 31.20 [Languages], page 116, and See Section 31.21 [Records_Languages], page 116.

See also Section 23.3.1 [010@ Fields], page 52, or Section 24.4.1 [1500 Fields], page 63.

`int f_010_AT_a (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ field:
a

English: Language Code for the Present Text

German: Sprachencode für den vorliegenden Text

27.3.4.4 Year of Appearance

Pica+ Code:
011@
See ⟨undefined⟩ [011@ Year of Appearance], page ⟨undefined⟩.

Pica3 Code:
1100
See ⟨undefined⟩ [1100 Year of Appearance], page ⟨undefined⟩.

German: Erscheinungsjahr

English: Year of Appearance

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/1100.pdf`

Database Tables:
`Records`. See Section 31.5 [Records], page 111.

See also Section 23.4.1 [011@ Fields], page 53, or Section 24.3.1 [1100 Fields], page 63.

`int f_011_AT_a (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ field:
a

English: Year of Appearance (Beginning), Form for Sorting

German: Erscheinugsjahr (Beginn), Sortierform

`int f_011_AT_b (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ field:
b

English: Year of Appearance (End), Form for Sorting

German: Erscheinugsjahr (Ende), Sortierform

`int f_011_AT_e (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ field:
e

English: Original Year of Appearance

German: Ursprüngliches Erscheinugsjahr

`int f_011_AT_n (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ field:
e

English: Year of Appearance (according to RAK-WB)

German: Erscheinugsjahr (nach RAK-WB)

`int year_appearance_func (string column, CDatabase* database, long record_id, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

This function is called by the other `Subcategory_Container` functions `f_011_AT_a`, `f_011_AT_b`, `f_011_AT_e`, and `f_011_AT_n`, which are described above.

27.3.4.5 Main Canonical Title

int f_021A_1 (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

int f_021A_a (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

int f_021A_d (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

int f_021A_e (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

int f_021A_f (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

int f_021A_h (CDatabase* *database*, [Static Function]
long *record_id*,
Category_Container* *category*,
Subcategory_Container* *subcategory*,
Output_Stream_Type& *log_strm*)

27.3.4.6 First Author

`int f_028A_9 (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

`int f_028A_a (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

`int f_028A_c (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

`int f_028A_d (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

27.3.4.7 Second and Additional Authors

`int f_028B_9 (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

`int f_028B_a (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

`int f_028B_d (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

27.3.4.8 Other Contributing Persons

```
int f_028C_9 (CDatabase* database,                    [Static Function]
        long record_id,
        Category_Container* category,
        Subcategory_Container* subcategory,
        Output_Stream_Type& log_strm)
```

```
int f_028C_a (CDatabase* database,                    [Static Function]
        long record_id,
        Category_Container* category,
        Subcategory_Container* subcategory,
        Output_Stream_Type& log_strm)
```

```
int f_028C_c (CDatabase* database,                    [Static Function]
        long record_id,
        Category_Container* category,
        Subcategory_Container* subcategory,
        Output_Stream_Type& log_strm)
```

```
int f_028C_d (CDatabase* database,                    [Static Function]
        long record_id,
        Category_Container* category,
        Subcategory_Container* subcategory,
        Output_Stream_Type& log_strm)
```

27.3.4.9 Place of Publication; Publisher

Pica+ Code:
: 033A
See Section 23.5 [033A Pica+ Codes], page 53.

Pica3 Code:
: 4030
See Section 24.5 [4030 Pica3 Codes], page 64.

German: Ort, Verlag

English: Place, Publisher

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/4030.pdf`

Database Tables:
: `Publishers`, `Records_Publishers` See Section 31.16 [Publishers], page 115, and See Section 31.17 [Records_Publishers], page 115.

`Category_Container` Functions:
: `Category_Container::F_033A`, `Category_Container::publishers_database_providers_func`. See Section 26.3.3.4 [Place of Publication; Publisher], page 71.

See also Section 23.5.1 [033A Fields], page 54, or Section 24.5.1 [4030 Fields], page 64.

`int f_033A_n (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

`int f_033A_p (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

27.3.4.10 Physical Description

Pica+ Code:
: 034D
: See Section 23.6 [034D Pica+ Codes], page 54.

Pica3 Code:
: 4060
: See Section 24.6 [4060 Pica3 Codes], page 65.

German: Umfangsangabe, spezifische Materialbenennung, technisches System

English: Size or Range, Specification of Material, Technical System

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/4060.pdf`

Database Tables:
: `Physical_Descriptions`, `Records_Physical_Descriptions`. See Section 31.22 [Physical_Descriptions], page 116, and Section 31.23 [Records_Physical_Descriptions], page 117.

See also Section 23.6.1 [034D Fields], page 54, or Section 24.6.1 [4060 Fields], page 65.

`int f_034D_a (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+ field:
: a

English: Text

German: Text

27.3.4.11 Subject

Pica+ Code:
: 041A
  See Section 23.7 [041A Pica+ Codes], page 55.

Pica3 Codes:
: 800; 5100–5199
  See Section 24.1 [800; 5100–5199 Pica3 Codes], page 60.

German: Hauptschlagwort und Unterschlagwoerter (Schlagwortansetzung) (Pica3 800)
RSWK-Ketten (Pica3 5100–5199)

English: Main Subject and Subsidiary Subjects (Subject Assignment) (Pica3 800)
RSWK Chains (Pica3 5100–5199)

Cataloguing Guidelines:
: Pica3 800: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/norm/800.pdf`
  Pica3 5100–5199: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/5100.pdf`

Database Tables:
: `Subjects`. See Section 31.31 [Subjects], page 121.
  `Records_Subjects`. See Section 31.32 [Records_Subjects], page 121.
  `Permutation_Patterns`. See Section 31.33 [Permutation_Patterns], page 121.

`Category_Container` Functions:
: `F_041A` and `sub_F_041A`.
  See Section 26.3.3.5 [Subject], page 72.

See also Section 23.7.1 [041A Fields], page 55, or Section 24.1.1 [800; 5100–5199 Fields], page 60.

`int f_041A_9 (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	Pica3	German	English
9	!. . .!	Identifikationsnummer (PPN)	Identification Number (PPN)

See also Section 23.7.1 [041A Fields], page 55.

`int f_041A_a (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Field '`a`' of Pica+ 401A is ambiguous, and can have three different meanings.

Pica+	**Pica3**	**German**	**English**
a	(Not Present)	Schlagwort	Subject
a	(Not Present)	Zweites und weiteres Permutationsmuster	Second and additional permutation pattern
a	(Not Present)	Angaben zur Schlagwortkette	Subject Chain Information

See also Section 23.7.1 [041A Fields], page 55.

int f_041A_f (CDatabase* *database*, long *record_id*, Category_Container* *category*, Subcategory_Container* *subcategory*, Output_Stream_Type& *log_strm*) [Static Function]

Pica+	**Pica3**	**German**	**English**
f	⟨...⟩	Permutationsmuster	Permutation Pattern

See also Section 23.7.1 [041A Fields], page 55.

int f_041A_S (CDatabase* *database*, long *record_id*, Category_Container* *category*, Subcategory_Container* *subcategory*, Output_Stream_Type& *log_strm*) [Static Function]

Pica+	**Pica3**	**German**	**English**
S	|...|¬	Indikator	Indicator

See also Section 23.7.1 [041A Fields], page 55.

27.3.4.12 Content Summary (Short)

int f_047I_a (CDatabase* *database*, long *record_id*, Category_Container* *category*, Subcategory_Container* *subcategory*, Output_Stream_Type& *log_strm*) [Static Function]

27.3.4.13 Exemplar Production Number

Pica+ Code:
: 203@
: See Section 23.8 [203@ Exemplar Production Number], page 57.

Pica3 Code:
: 7800
: See Section 24.10 [7800 Exemplar Production Number], page 67.

German: Exemplar-Produktionsnummer

English: Exemplar Production Number

Cataloguing Guidelines:
Not present.

Database Table:
`Exemplar_Production_Numbers`. See Section 31.26 [Exemplar_Production_Numbers], page 118.

See also Section 23.8.1 [203@ Fields], page 57, or Section 24.10.1 [7800 Fields], page 68.

```
int f_203_AT_0 (CDatabase* database,
        long record_id,
        Category_Container* category,
        Subcategory_Container* subcategory,
        Output_Stream_Type& log_strm)
```
[Static Function]

Pica+	**Pica3**	**German**	**English**
0		Exemplar-Produktionnummer	Exemplar Production Number

See also Section 23.8.1 [203@ Fields], page 57.

27.3.4.14 Call Number

Pica+ Code:
209A
See Section 23.9 [209A Call Number], page 57.

Pica3 Codes:
7100–7109
See Section 24.8 [7100–7109 Call Number], page 65.

German: Signatur

English: Call Number

Cataloguing Guidelines:
`http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7100.pdf`

Database Table:
`Call_Numbers`. See Section 31.27 [Call_Numbers], page 119.

`Category_Container` Functions:
`Category_Container::F_209A` and `Category_Container::sub_F_209A`. See Section 26.3.3.6 [Call Number], page 73.

See also Section 23.9.1 [209A Fields], page 58, or Section 24.8.1 [7100–7109 Fields], page 66.

`int f_209A_a (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	Pica3	German	English
a	(Not Present)	Signatur	Call Number

See also Section 23.9.1 [209A Fields], page 58.

`int f_209A_b (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	Pica3	German	English
b	...	Bibliotheksnummer	Library Number

In Pica3, '`...`' may be followed directly by '`/...#`', which corresponds to Pica+ '`j`'. See the description of `Subcategory_Container::f_209A_j` below.
See also Section 23.9.1 [209A Fields], page 58.

`int f_209A_f (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	Pica3	German	English
f	!...!	Sonderstandort	Special Location

See also Section 23.9.1 [209A Fields], page 58.

`int f_209A_j (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	Pica3	German	English
j	/...#	Abteilung der Bibliothek	Library Department

In Pica3, '`/...#`' may directly follow '`...`', which corresponds to Pica+ '`b`'. See the description of `Subcategory_Container::f_209A_b` above.
See also Section 23.9.1 [209A Fields], page 58.

27.3.4.15 Access Number

Pica+ Code:
: 209C
  See Section 23.10 [209C Access Number], page 58.

Pica3 Codes:
: 8100
  See Section 24.11 [8100 Access Number], page 68.

German: Zugangsnummer

English: Access Number

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/8100.pdf`

Database Table:
: `Access_Numbers`. See Section 31.29 [Access_Numbers], page 119.

See also Section 23.10.1 [209C Fields], page 58, or Section 24.11.1 [8100 Fields], page 68.

```
int f_209C_a (CDatabase* database,
        long record_id,
        Category_Container* category,
        Subcategory_Container* subcategory,
        Output_Stream_Type& log_strm)
```
[Static Function]

Pica+	**Pica3**	**German**	**English**
a	(Not Present)	Zugangsnummer	Access Number

See also Section 23.10.1 [209C Fields], page 58.

27.3.4.16 Local information regarding remote access to electronic resources

Pica+ Code:
: 209R
  See Section 23.11 [209R Local Information Regarding Remote Access to Electronic Resources], page 59.

Pica3 Code:
: 7133
  See Section 24.9 [7133 Local Information Regarding Remote Access to Electronic Resources], page 66.

German: Lokale Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff

English: Local information regarding remote access to electronic resources

Cataloguing Guidelines:
: `http://www.gbv.de/vgm/info/mitglieder/02Verbund/01Erschliessung/02Richtlinien/01KatRicht/7133.pdf`

Database Table:
: `Remote_Access`. See Section 31.24 [Remote_Access], page 117.

`Category_Container` Functions:
: `Category_Container::F_209R` and `Category_Container::sub_F_209R`. See ⟨undefined⟩ [Functions for Specific Categories], page ⟨undefined⟩.

See also Section 23.11.1 [209R Fields], page 59, or Section 24.9.1 [7133 Fields], page 67.

`int f_209R_0 (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	**Pica3**	**German**	**English**
0		Format	Format

See also Section 23.11.1 [209R Fields], page 59.

`int f_209R_a (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	**Pica3**	**German**	**English**
a	¬ =A¬	URL (Universal Resource Locator)	URL (Universal Resource Locator)

See also Section 23.11.1 [209R Fields], page 59.

`int f_209R_g (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	**Pica3**	**German**	**English**
g	¬ =G¬	URN (Universal Resource Name)	URN (Universal Resource Name)

See also Section 23.11.1 [209R Fields], page 59.

`int f_209R_S (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	Pica3	German	English
S	⟨ ⟩	Lizenzindikator	License indicator

See also Section 23.11.1 [209R Fields], page 59.

`int f_209R_x (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	Pica3	German	English
x	¬ =X¬	Interne Bemerkungen	Internal Remarks

See also Section 23.11.1 [209R Fields], page 59.

`int f_209R_y (CDatabase* database, long record_id, Category_Container* category, Subcategory_Container* subcategory, Output_Stream_Type& log_strm)` [Static Function]

Pica+	Pica3	German	English
y	¬[. . .]	Text für die Web-Anzeige	Text for Web Display

See also Section 23.11.1 [209R Fields], page 59.

28 Database_Command

`struct Database_Command` is declared in ‘`dbcmmnd.web`’.

28.1 Data Members

(Pointer to function) **`function`** [Public variable]

`function` is a pointer to a function returning `int` and taking the following arguments:
`CDatabase*` *database*
`long` *record_id*
`Category_Container*` *category*
`Subcategory_Container*` *subcategory*
`Output_Stream_Type&` *log_strm*

28.2 Functions

28.2.1 Constructor

`void Database_Command (void)` [Default Constructor]

28.2.2 Destructor

`void ~Database_Command (void)` [Destructor]

29 Displaying Database Contents (DB_Display)

class DB_Display is declared in 'dbdspl.web'.

29.1 Data Members

`ofstream html_strm` [Private variable]

29.2 Functions

29.2.1 Constructor

`void DB_Display (void)` [Default Constructor]

29.2.2 Destructor

`void ~DB_Display (void)` [Destructor]

29.2.3 Counting Records

`int count_records (unsigned int` *start*`, unsigned int` *end*`,` [Public Function]
`[vector<long>*` *record_id_vector* = *0*`])`

29.2.4 Output

`int open_html_file (const char*` *html_filename*`)` [Public Function]

`int display_single_record (` [Public Function]
`const unsigned int` *record_number*`,`
`const unsigned int` *record_ctr*`)`

Some of the field values in a Pica+ record contain information that should be unique for that record. For example, a record should have only one Pica Production Number, only one Year of Appearance, etc. The Records table in the PICA database contains columns for this information.

However, many other categories and fields can occur in a Pica+ record multiple times. It is therefore not possible to store all of the information for a ZTest record in a single table in the PICA database. For example, a resource may have one, two, three, or any number of authors. On the other hand, a resource may have no author at all. This is frequently the case for audio-visual media, where other categories, such as "contributor" are used for the names of the creators of the resource. If the Records table contained a single 'author_name' column, only one author's name could be stored in a line of the Records table. If the resource referred to by this record had no author, the column would have to contain NULL or a default value. On the other hand, if it had two authors, one would have to be left out. Nor would redefining the Records table with multiple columns for authors, e.g., 'author_name_1', 'author_name_2', etc., solve the problem. With this approach, there would always a fixed limit for the number of authors' names that could be accommodated, and space would always be wasted, if fewer authors were present in a record.

For categories and fields where a variable number of entries are possible, the PICA database uses a combination of *foreign key constraints* and *association tables*, as explained in Section 3.1 [Association Tables], page 4.

This implies that the data for a given record will be distributed over a number of different tables in the PICA database. Therefore, `display_single_record` must first find all of this information in order to display it.

Once the information is gathered, there are many possible ways of displaying it. The form of display performed by `display_single_record` is designed to be simple and complete, and reflect the structure of the PICA database. The code in `display_single_record` could be used as the basis for fancier and more complicated displays.

`display_single_record` writes HTML code to the output file. Each table which contains entries for the current record is represented by a ruled HTML table with fields for the names of the columns and their contents. If the database table has too many columns to be displayed comfortably across the breadth of a browser window, the names of the database columns may appear in a column on the left of the HTML table, with the values in fields to the right. If a database table contains more than one row for the current record, an additional HTML table will be created for each database row. Otherwise, `display_single_record` will create a single HTML table with the names of the database columns names in a row of the HTML table with the corresponding values in one or more rows beneath them.

The first table displayed is `Records` table. There is no difficulty in extracting this information from the database, since there will only ever be one row in this table for a record by definition. For the subsequent HTML tables, the entries that correspond to the current record must be extracted from the corresponding database table. This is done in one of two ways, depending on whether the connection between the database table and the entry for the record in the `Records` table is created by means of an association table or a foreign key constraint in the current database table.

The database table `Content_Summaries` uses a foreign key constraint. For example, Record 23 may have a content summary, which is stored in the `Contents_Summaries` table. The value in the '`content_summary_id`' column for the entry in the `Contents_Summaries` table may have the value 92. The `Contents_Summaries` table also has a column '`record_id`', which references the column '`record_id`' in the `Records` table. Its value will therefore be 23 in this entry. The following Transact-SQL query will return this row from the `Contents_Summaries` table:

```
select * from Content_Summaries where record_id = 23 order by content_summary_id
```

The `Authors` database, on the other hand, doesn't have a '`record_id`' column with an associated foreign key constraint, but uses an association table, `Records_Authors`, instead. For example, Record 48 may have three authors, and the entries for these authors in the `Authors` table may the values of 104, 105, and 210 in their '`author_id`' columns. The `Records_Authors` table has only two columns, '`record_id`' and '`author_id`', whereby the former references the '`record_id`' column in the `Records` table, and the latter references the '`author_id`' column in the `Authors` table. The following Transact-SQL query will return the data from all the rows in the `Authors` table that correspond to authors of the resource referenced by Record 48. Please note that '`select *`' cannot be used in this query, and that the columns of the `Authors` table must be named explicitly, because the `Records_Authors` and '`Records`' are both named in the "`from`" clause of the "`select`" query.

```
select A.given_name, A.surname, A.prefix, A.id_number_ppn
from Authors as A, Records_Authors as RA, Records as R
where
A.author_id = RA.author_id
and
R.record_id = RA.record_id
and
R.record_id = 48
order by A.author_id
```

Whether a column with a foreign key constraint or a separate association table is used, Transact-SQL queries similar to the ones described above are used to extract the information for a given record from the various database tables in the PICA database where this information is stored. The Transact-SQL query is passed to the `Open` function of an object whose type is an class derived from `CRecordset` (an ODBC class) that corresponds to one of the database tables. See Chapter 30 [ODBC Classes for ZTest], page 98.

ZTest contains an ODBC class corresponding to each of the database tables in the PICA database. For example, `class Records` corresponds to the `Records` table, `class Authors` corresponds to the `Authors` table, etc. `display_single_record` declares an object of each of these types. When its '`Open`' function is called with a Transact-SQL query, and this query is successful, i.e., it returns one or more rows of data from the database table, the data members of the ODBC class object are "filled" with the data from the first row from the set of rows returned by the Transact-SQL query. That is, the data members of the ODBC class object are assigned the values from the corresponding columns of the first row of data. The '`order by`' clause in the Transact-SQl queries ensures that the rows are ordered correctly. `display_single_record` now uses the "navigation functions" `CRecordset` to iterate through the rows. Each time `CRecordset::MoveNext` is invoked on an ODBC class object, the data members are assigned values from the next row of data from the database, until there are no more rows. `display_single_record` then writes this data to the output file, along with the appropriate "boilerplate" HTML code to format it.

`int display_records (` [Public Function]
  [`CString` *`search_command_str`* = `""`,
  [`const unsigned int` *`start`* = 1,
  [`const unsigned int` *`end`* = 0]]])

`display_records` displays a range of records from record number *start* to record number *end*. Both of these arguments have default values. The default for *start* is 1 and the default for *end* is 0.

If *end* = 0, either because it was passed by the caller explicitly, or because the default is being used, then `display_records` starts with record *start* and continues until `display_single_record` returns unsuccessfully (return value 1).

Otherwise, if *end* > 0 and *end* < *start*, then a warning is issued, *end* is set to 0, and `display_records` proceeds as above.

Otherwise, `display_records` starts displaying records from record number *start*, and continues up to and including record number *end*. It will continue to call `display_single_record`, even if the latter returns unsuccessfully for some record number. This is because there may be gaps in the sequence of record numbers because of deletions, or for some other reason. This isn't likely, but it is possible.

These rules imply that a call to `display_records` with no arguments will display all of the records in the `PICA_DB` database.

`int close_html_file (void)` [Public Function]

30 ODBC Classes for ZTest

30.1 PICA_Categories

Class `PICA_Categories` is declared in '`picacats.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `PICA_Categories` table in the `PICA` database. See Section 31.2 [PICA_Categories], page 110.

Data Members

`long m_pica_category_id` [Variables]
`CStringA m_pica_plus_category_code`
`int m_pica_3_category_code`
`CStringA m_description_german`
`CStringA m_description_english`

These variables reference the corresponding columns in the `PICA_Categories` table in the `PICA` database. See Section 31.2 [PICA_Categories], page 110.

30.2 PICA_Fields

Class `PICA_Fields` is declared in '`picaflds.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `PICA_Fields` table in the `PICA` database. See Section 31.3 [PICA_Fields], page 110.

Data Members

`long m_pica_field_id` [Variables]
`CStringA m_pica_plus_field_code`
`int` *m_pica_3_field_code*
`CStringA m_description_german`
`CStringA m_description_english`

These variables reference the corresponding columns in the `PICA_Fields` table in the `PICA` database. See Section 31.3 [PICA_Fields], page 110.

30.3 PICA_Categories_PICA_Fields

Class `PICA_Categories_PICA_Fields` is declared in '`pccatfld.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the association table `PICA_Categories_PICA_Fields` in the `PICA` database. See Section 31.4 [PICA_Categories_PICA_Fields], page 111.

Data Members

`long m_pica_category_id` [Variables]
`long m_pica_field_id`

These variables reference the corresponding columns in the `PICA_Categories_PICA_Fields` table in the `PICA` database. See Section 31.4 [PICA_Categories_PICA_Fields], page 111.

30.4 Sources

Class `Sources` is declared in '`sources.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Sources` table in the `PICA` database. See Section 31.1 [Sources], page 110.

Data Members

`long m_source_id` [Variables]
`CStringA m_source_name`
`CStringA m_source_abbrev`
`CStringA m_source_address`

These variables reference the corresponding columns in the `Sources` table in the `PICA` database. See Section 31.1 [Sources], page 110.

30.5 Records

Class `Records` is derived from the MFC class `CRecordset` using public derivation. It is declared in '`records.web`'.

!! TODO: Note the fact that the member functions of classes derived from `CRecordset` are all generated by Visual Studio .NET and are not documented here. LDF 2006.08.28.

Class `Records` references the `Records` table in the `PICA` database. See Section 31.5 [Records], page 111.

Data Members

`long m_record_id` [Variables]
`int m_eln_original_entry`
`int m_eln_most_recent_change`
`int m_eln_status_change`
`CStringA m_identification_number`
`CTime m_date_original_entry`
`CTime m_date_most_recent_change`
`CTime m_date_status_change`
`long m_source_id`
`int m_year_appearance_begin`
`int m_year_appearance_end`
`int m_year_appearance_rak_wb`
`int m_year_appearance_original`

These variables reference the corresponding columns in the `Records` table in the `PICA` database. See Section 31.5 [Records], page 111.

30.6 Access_Numbers

Class `Access_Numbers` is declared in '`accnums.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Access_Numbers` table in the `PICA` database. See Section 31.29 [Access_Numbers], page 119.

Data Members

long m_access_number_id [Variables]
CStringA m_access_number
long m_record_id

These variables reference the corresponding columns in the Access_Numbers table in the PICA database. See Section 31.29 [Access_Numbers], page 119.

30.7 Call_Numbers

Class Call_Numbers is declared in 'callnums.web'. It is derived from the MFC class CRecordset using public derivation. It references the Call_Numbers table in the PICA database. See Section 31.27 [Call_Numbers], page 119.

Data Members

long m_call_number_id [Variables]
CStringA m_call_number
int m_library_number
CStringA m_library_department
CStringA m_special_location

These variables reference the corresponding columns in the Call_Numbers table in the PICA database. See Section 31.27 [Call_Numbers], page 119.

30.8 Records_Call_Numbers

Class Records_Call_Numbers is declared in 'rccllnms.web'. It is derived from the MFC class CRecordset using public derivation. It references the Records_Call_Numbers table in the PICA database. See Section 31.28 [Records_Call_Numbers], page 119.

Data Members

long m_record_id [Variables]
long m_call_number_id

These variables reference the corresponding columns in the Records_Call_Numbers table in the PICA database. See Section 31.28 [Records_Call_Numbers], page 119.

30.9 Exemplar_Production_Numbers

Class Exemplar_Production_Numbers is declared in 'exprnums.web'. It is derived from the MFC class CRecordset using public derivation. It references the Exemplar_Production_Numbers table in the PICA database. See Section 31.26 [Exemplar_Production_Numbers], page 118.

Data Members

long m_exemplar_production_number_id [Variables]
LONGLONG m_exemplar_production_number_numeric
CStringA m_exemplar_production_number_text

`long m_record_id`

These variables reference the corresponding columns in the `Exemplar_Production_Numbers` table in the `PICA` database. See Section 31.26 [Exemplar_Production_Numbers], page 118.

30.10 Bibliographic_Type_Codes

Class `Bibliographic_Type_Codes` is declared in '`bbtpcds.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Bibliographic_Type_Codes` table in the `PICA` database. See Section 31.6 [Bibliographic_Type_Codes], page 112.

Data Members

```
long m_bibliographic_type_code_id                                [Variables]
CStringA m_physical_form_code
CStringA m_physical_form_material_name_english
CStringA m_physical_form_material_name_german
CStringA m_bibliographic_representation_code
CStringA m_bibliographic_representation_description_english
CStringA m_bibliographic_representation_description_german
CStringA m_description_status_code
CStringA m_description_status_description_english
CStringA m_description_status_description_german
```

These variables reference the corresponding columns in the `Bibliographic_Type_Codes` table in the `PICA` database. See Section 31.6 [Bibliographic_Type_Codes], page 112.

30.11 Bibliographic_Types

Class `Bibliographic_Types` is declared in '`bibtyps.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Bibliographic_Types` table in the `PICA` database. See Section 31.7 [Bibliographic_Types], page 112.

Data Members

```
long m_bibliographic_type_id                                     [Variables]
CStringA m_physical_form;
CStringA m_bibliographic_representation;
CStringA m_description_status;
CStringA m_miscellaneous;
CStringA m_bibliographic_representation_refinement;
CStringA m_transliteration_code;
```

These variables reference the corresponding columns in the `Bibliographic_Types` table in the `PICA` database. See Section 31.7 [Bibliographic_Types], page 112.

30.12 Records_Bibliographic_Types

Class `Records_Bibliographic_Types` is declared in '`rcbbtyps.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Records_Bibliographic_`

Types table in the PICA database. See Section 31.8 [Records_Bibliographic_Types], page 112.

Data Members

long m_record_id [Variables]
long m_bibliographic_type_id

These variables reference the corresponding columns in the Records_Bibliographic_Types table in the PICA database. See Section 31.8 [Records_Bibliographic_Types], page 112.

30.13 Physical_Descriptions

Class Physical_Descriptions is declared in 'physdesc.web'. It is derived from the MFC class CRecordset using public derivation. It references the Physical_Descriptions table in the PICA database. See Section 31.22 [Physical_Descriptions], page 116.

Data Members

long m_physical_description_id [Variables]
CStringA m_text
long m_pica_category_id
long m_pica_field_id

These variables reference the corresponding columns in the Physical_Descriptions table in the PICA database. See Section 31.22 [Physical_Descriptions], page 116.

30.14 Records_Physical_Descriptions

Class Records_Physical_Descriptions is declared in 'rcphsdsc.web'. It is derived from the MFC class CRecordset using public derivation. It references the association table Records_Physical_Descriptions in the PICA database. See Section 31.23 [Records_Physical_Descriptions], page 117.

Data Members

long m_record_id [Variables]
long m_physical_description_id

These variables reference the corresponding columns in the Records_Physical_Descriptions table in the PICA database. See Section 31.23 [Records_Physical_Descriptions], page 117.

30.15 Authors

Class Authors is declared in 'authors.web'. It is derived from the MFC class CRecordset using public derivation. It references the Authors table in the PICA database. See Section 31.12 [Authors], page 114.

Data Members

long m_author_id [Variables]

`CStringA m_given_name`
`CStringA m_surname`
`CStringA m_prefix`

These variables reference the corresponding columns in the `Authors` table in the `PICA` database. See Section 31.12 [Authors], page 114.

30.16 Records_Authors

Class `Records_Authors` is declared in '`recathrs.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the association table `Records_Authors` in the `PICA` database. See Section 31.13 [Records_Authors], page 114.

Data Members

`long m_record_id` [Variables]
`long m_author_id`

These variables reference the corresponding columns in the `Records_Authors` table in the `PICA` database. See Section 31.13 [Records_Authors], page 114.

30.17 Contributors

Class `Contributors` is declared in '`cntrbtrs.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Contributors` table in the `PICA` database. See Section 31.14 [Contributors], page 114.

Data Members

`long m_contributor_id` [Variables]
`CStringA m_given_name`
`CStringA m_surname`
`CStringA m_prefix`
`LONGLONG m_id_number_ppn`

These variables reference the corresponding columns in the `Contributors` table in the `PICA` database. See Section 31.14 [Contributors], page 114.

30.18 Records_Contributors

Class `Records_Contributors` is declared in '`rccntrbt.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the association table `Records_Contributors` in the `PICA` database. See Section 31.15 [Records_Contributors], page 115.

Data Members

`long m_record_id` [Variables]
`long m_contributor_id`

These variables reference the corresponding columns in the `Records_Contributors` table in the `PICA` database. See Section 31.15 [Records_Contributors], page 115.

30.19 Main_Titles

Class Main_Titles is declared in 'mnttls.web'. It is derived from the MFC class CRecordset using public derivation. It references the Main_Titles table in the PICA database. See Section 31.9 [Main_Titles], page 113.

Data Members

long m_main_title_id [Variables]
CStringA m_standard_text
CStringA m_main_canonical_title
long m_continuation_main_canonical_title
CStringA m_additions_main
long m_continuation_additions_main
CStringA m_additional_creator_main
CStringA m_parallel_canonical_title
CStringA m_additions_parallel
CStringA m_additional_creator_parallel
CStringA m_authorship

These variables reference the corresponding columns in the Main_Titles table in the PICA database. See Section 31.9 [Main_Titles], page 113.

30.20 Records_Main_Titles

Class Records_Main_Titles is declared in 'rcmnttls.web'. It is derived from the MFC class CRecordset using public derivation. It references the association table Records_Main_Titles in the PICA database. See Section 31.10 [Records_Main_Titles], page 113.

Data Members

long m_record_id [Variables]
long m_main_title_id

These variables reference the corresponding columns in the Records_Main_Titles table in the PICA database. See Section 31.10 [Records_Main_Titles], page 113.

30.21 Content_Summaries

Class Content_Summaries is declared in 'contsums.web'. It is derived from the MFC class CRecordset using public derivation. It references the Content_Summaries table in the PICA database. See Section 31.11 [Content_Summaries], page 114.

Data Members

long m_content_summary_id [Variables]
long m_record_id
long m_continuation
CStringA m_content_summary

These variables reference the corresponding columns in the Content_Summaries table in the PICA database. See Section 31.11 [Content_Summaries], page 114.

30.22 Languages

Class `Languages` is declared in '`language.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Languages` table in the `PICA` database. See Section 31.20 [Languages], page 116.

Data Members

`long m_language_id` [Variables]
`CStringA m_language_name_english`
`CStringA m_language_name_german`
`CStringA m_language_abbrev`

These variables reference the corresponding columns in the `Languages` table in the `PICA` database. See Section 31.20 [Languages], page 116.

30.23 Records_Languages

Class `Records_Languages` is declared in '`reclang.web`'. It is derived from the MFC class `CRecordset` using public derivation. Its references the association table `Records_Languages` in the `PICA` database. See Section 31.21 [Records_Languages], page 116.

Data Members

`long m_record_id` [Variables]
`long m_language_id`
`CStringA m_association_type`
`CStringA m_association_type_name`

These variables reference the corresponding columns in the `Records_Languages` table in the `PICA` database. See Section 31.21 [Records_Languages], page 116.

30.24 Subject_Types

Class `Subject_Types` is declared in '`subjtyps.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Subject_Types` table in the `PICA` database. See Section 31.30 [Subject_Types], page 120.

Data Members

`long m_subject_type_id` [Variables]
`CStringA m_indicator`
`CStringA m_description_german`
`CStringA m_description_english`
`BOOL m_pica3_800`
`BOOL m_pica3_51xx`

These variables reference the corresponding columns in the `Subject_Types` table in the `PICA` database. See Section 31.30 [Subject_Types], page 120.

30.25 Subjects

Class `Subjects` is declared in '`subjects.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Subjects` table in the `PICA` database. See Section 31.31 [Subjects], page 121.

Data Members

`long m_subject_id` [Variables]
`long m_subject_type_id`
`CStringA m_subject`
`LONGLONG m_id_number_ppn`
`int m_chain_number`
`int m_chain_link_number`
`CStringA m_chain_info`

These variables reference the corresponding columns in the `Subjects` table in the `PICA` database. See Section 31.31 [Subjects], page 121.

30.26 Records_Subjects

Class `Records_Subjects` is declared in '`recsubjs.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the association table `Records_Subjects` in the `PICA` database. See Section 31.32 [Records_Subjects], page 121.

Data Members

`long m_record_id` [Variables]
`long m_subject_id`

These variables reference the corresponding columns in the `Records_Subjects` table in the `PICA` database. See Section 31.32 [Records_Subjects], page 121.

30.27 Permutation_Patterns

Class `Permutation_Patterns` is declared in '`prmpttrn.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Permutation_Patterns` table in the `PICA` database. See Section 31.33 [Permutation_Patterns], page 121.

Data Members

`long m_permutation_pattern_id` [Variables]
`long m_record_id`
`long m_subject_id_start`
`long m_subject_id_end`
`long m_chain_number`
`CStringA m_permutation_pattern`

These variables reference the corresponding columns in the `Permutation_Patterns` table in the `PICA` database. See Section 31.33 [Permutation_Patterns], page 121.

30.28 Remote_Access

Class `Remote_Access` is declared in '`rmaccess.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Remote_Access` table in the `PICA` database. See Section 31.24 [Remote_Access], page 117.

Data Members

`long m_remote_access_id` [Variables]
`long m_remote_access_id`
`BOOL m_license_indicator`
`CStringA m_format_type`
`CStringA m_web_display_text`
`CStringA m_URL`
`CStringA m_URN`
`CStringA m_internal_remarks`

These variables reference the corresponding columns in the `Remote_Access` table in the `PICA` database. See Section 31.24 [Remote_Access], page 117.

30.29 Records_Remote_Access

Class `Records_Remote_Access` is declared in '`rcrmaccs.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the association table `Records_Remote_Access` in the `PICA` database. See Section 31.25 [Records_Remote_Access], page 118.

Data Members

`long m_record_id` [Variables]
`long m_remote_access_id`

These variables reference the corresponding columns in the `Records_Remote_Access` table in the `PICA` database. See Section 31.25 [Records_Remote_Access], page 118.

30.30 Publishers

Class `Publishers` is declared in '`publshrs.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Publishers` table in the `PICA` database. See Section 31.16 [Publishers], page 115.

Data Members

`long m_publisher_id` [Variables]
`CStringA m_publisher_name`
`CStringA m_place`
`BOOL m_primary_info_source`

These variables reference the corresponding columns in the `Publishers` table in the `PICA` database. See Section 31.16 [Publishers], page 115.

30.31 Records_Publishers

Class `Records_Publishers` is declared in '`recpubs.web`'. It is derived from the MFC class `CRecordset` using public derivation. Its references the association table `Records_Publishers` in the `PICA` database. See Section 31.17 [Records_Publishers], page 115.

Data Members

`long m_record_id` [Variables]
`long m_publisher_id`

These variables reference the corresponding columns in the `Records_Publishers` table in the `PICA` database. See Section 31.17 [Records_Publishers], page 115.

30.32 Database_Providers

Class `Database_Providers` is declared in '`dbprovs.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Database_Providers` table in the `PICA` database. See Section 31.18 [Database_Providers], page 115.

Data Members

`long m_database_provider_id` [Variables]
`CStringA m_database_provider_name`
`CStringA m_place`

These variables reference the corresponding columns in the `Database_Providers` table in the `PICA` database. See Section 31.18 [Database_Providers], page 115.

30.33 Records_Database_Providers

Class `Records_Database_Providers` is declared in '`rcdbprov.web`'. It is derived from the MFC class `CRecordset` using public derivation. Its references the association table `Records_Database_Providers` in the `PICA` database. See Section 31.19 [Records_Database_Providers], page 116.

Data Members

`long m_record_id` [Variables]
`long m_database_provider_id`

These variables reference the corresponding columns in the `Records_Database_Providers` table in the `PICA` database. See Section 31.19 [Records_Database_Providers], page 116.

30.34 Temp_IDs

Class `Temp_IDs` is declared in '`tempids.web`'. It is derived from the MFC class `CRecordset` using public derivation. It references the `Temp_IDs` table in the `PICA` database. See Section 31.34 [Temp_IDs], page 122.

Data Members

`long m_temp_id` [Variable]

This variable references the corresponding column in the `Temp_IDs` table in the `PICA` database. See Section 31.34 [Temp_IDs], page 122.

31 Database Tables for ZTest

31.1 Sources

`int identity (0, 1) not null source_id` [Column]
`primary key PK_Sources` [Constraint]

`varchar(128) not null source_name` [Column]
The default value is 'N/A'.

`varchar(32) not null source_abbrev` [Column]
The default value is 'N/A'.

`varchar (1024) not null source_address` [Column]
The default value is 'N/A'.

31.2 PICA_Categories

`int identity (0, 1) not null pica_category_id` [Column]
`primary key PK_PICA_Categories` [Constraint]

`char(7) not null pica_plus_category_code` [Column]
The default value is 'N/A'.

`smallint null pica_3_category_code` [Column]

`varchar(128) not null description_german` [Column]
The default value is 'N/A'.

`varchar(128) not null description_english` [Column]
The default value is 'N/A'.

31.3 PICA_Fields

`int identity (0, 1) not null pica_field_id` [Column]
`primary key PK_PICA_Fields` [Constraint]

`char(7) not null pica_plus_field_code` [Column]
The default value is 'N/A'.

`char(4) not null pica_3_field_code` [Column]
The default value is 'N/A'.

`varchar(128) not null description_german` [Column]
The default value is 'N/A'.

`varchar(128) not null description_english` [Column]
The default value is 'N/A'.

31.4 PICA_Categories_PICA_Fields

Association table.

`int not null pica_category_id` [Column]

References `PICA_Categories(pica_category_id)`.
See Section 31.2 [PICA_Categories], page 110.
The default is 0.

`int not null pica_field_id` [Column]

References `PICA_Fields(pica_field_id)`.
See Section 31.3 [PICA_Fields], page 110.
The default is 0.

31.5 Records

`int identity(0, 1) not null record_id` [Column]
`primary key PK_Records` [Constraint]

`varchar(512) not null identification_number` [Column]

Can contain non-numerals! The default value is `'N/A'`.

`datetime date_original_entry` [Column]

TO DO: Change to “`not null`” after testing!

`datetime date_most_recent_change` [Column]

TO DO: Change to “`not null`” after testing!

`datetime date_status_change` [Column]

TO DO: Change to “`not null`” after testing!

`smallint eln_original_entry` [Column]

TO DO: Change to “`not null`” after testing!

`smallint eln_most_recent_change` [Column]

TO DO: Change to “`not null`” after testing!

`smallint eln_status_change` [Column]

TO DO: Change to “`not null`” after testing!

`smallint null year_appearance_begin` [Column]

`smallint null year_appearance_end` [Column]

`smallint null year_appearance_rak_wb` [Column]

`smallint null year_appearance_original` [Column]

`int not null source_id` [Column]

References `Sources(source_id)`. The default value is 0.

31.6 Bibliographic_Type_Codes

This table is filled by the stored procedure `regenerate_tables`. See Chapter 32 [Database Stored Procedures], page 123. It is not changed when ZTest is run.

`int identity(0, 1) not null bibliographic_type_code_id` [Column]
`primary key PK_Bibliographic_Type_Codes` [Constraint]

`char(1) null physical_form_code` [Column]

`varchar(128) null physical_form_material_name_english` [Column]

`varchar(128) null physical_form_material_name_german` [Column]

`char(1) null bibliographic_representation_code` [Column]

`varchar(128) null bibliographic_representation_description_english` [Column]

`varchar(128) null bibliographic_representation_description_german` [Column]

`char(1) null description_status_code` [Column]

`varchar(64) null description_status_description_english` [Column]

`varchar(64) null description_status_description_german` [Column]

31.7 Bibliographic_Types

`int identity not null bibliographic_type_id` [Column]
`primary key PK_Bibliographic_Types` [Constraint]

`char(1) not null physical_form` [Column]

`char(1) not null bibliographic_representation` [Column]

`char(1) not null description_status` [Column]

`char(1) null miscellaneous` [Column]

`char(1) null bibliographic_representation_refinement` [Column]

`char(1) null transliteration_code` [Column]

31.8 Records_Bibliographic_Types

Association table.

`int not null record_id` [Column]

References `Records(record_id)`. The default value is 0.

`int not null bibliographic_type_id` [Column]

References `Bibliographic_Types(bibliographic_type_id)`. The default value is 0.

31.9 Main_Titles

`int identity not null main_title_id` [Column]
`primary key PK_Main_Titles` [Constraint]

`varchar(512) not null standard_text` [Column]
The default value is `'N/A'`.

`varchar(512) not null main_canonical_title` [Column]
The default value is `'N/A'`.

`int not null continuation_main_canonical_title` [Column]
The default value is 0.

`varchar(512) not null additions_main` [Column]
The default value is `'N/A'`.

`int not null continuation_additions_main` [Column]
The default value is 0.

`varchar(512) not null additional_creator_main` [Column]
The default value is `'N/A'`.

`varchar(512) not null parallel_canonical_title` [Column]
The default value is `'N/A'`.

`varchar(512) not null additions_parallel` [Column]
The default value is `'N/A'`.

`varchar(512) not null additional_creator_parallel` [Column]
The default value is `'N/A'`.

`varchar(512) not null authorship` [Column]
The default value is `'N/A'`.

31.10 Records_Main_Titles

Association table.

`int not null record_id` [Column]
References `Records(record_id)`. The default value is 0.

`int not null main_title_id` [Column]
References `Main_Titles(main_title_id)`.

31.11 Content_Summaries

The `Content_Summaries` table is referenced by the ODBC class `Content_Summaries`. See Section 30.21 [Content_Summaries], page 104.

`int identity(0, 1) not null content_summary_id` [Column]
`primary key PK_Content_Summaries` [Constraint]

`int not null record_id` [Column]

References `Records(record_id)`. See Section 31.5 [Records], page 111.

`int not null continuation` [Column]

The default value is 0.

`varchar(1024) not null content_summary` [Column]

31.12 Authors

`int identity not null author_id` [Column]
`primary key PK_Authors` [Constraint]

`varchar(128) not null given_name` [Column]

The default value is `'N/A'`.

`varchar(128) not null surname` [Column]

The default value is `'N/A'`.

`varchar(16) not null prefix` [Column]

The default value is `'N/A'`.

`bigint not null id_number_ppn` [Column]

The default value is 0.

31.13 Records_Authors

Association table.

`int not null record_id` [Column]

References `Records(record_id)`. The default value is 0.

`int not null author_id` [Column]

References `Authors(author_id)`.

31.14 Contributors

`int identity not null contributor_id` [Column]
`primary key PK_Contributors` [Constraint]

`varchar(128) not null given_name` [Column]

The default value is `'N/A'`.

`varchar(128) not null surname` [Column]

The default value is `'N/A'`.

`varchar(16) not null prefix` [Column]

The default value is `'N/A'`.

`bigint not null id_number_ppn` [Column]

The default value is 0.

31.15 Records_Contributors

Association table.

`int not null record_id` [Column]

References `Records(record_id)`. The default value is 0.

`int not null contributor_id` [Column]

References `Contributors(contributor_id)`.

31.16 Publishers

`int identity not null publisher_id` [Column]
`primary key PK_Publishers` [Constraint]

`varchar(256) not null publisher_name` [Columns]

The default value is `'N/A'`.

`varchar(256) not null place` [Columns]

The default value is `'N/A'`.

`bit not null primary_info_source` [Columns]

The default value is `1`.

31.17 Records_Publishers

Association table.

`int not null record_id` [Column]

References `Records(record_id)`. The default value is 0.

`int not null publisher_id` [Column]

References `Publishers(publisher_id)`. The default value is 0.

31.18 Database_Providers

`int identity not null database_provider_id` [Column]
`primary key PK_Database_Providers` [Constraint]

`varchar(256) not null database_provider_name` [Columns]

The default value is `'N/A'`.

`varchar(256) not null place` [Columns]

The default value is `'N/A'`.

31.19 Records_Database_Providers

Association table.

`int not null record_id` [Column]
References `Records(record_id)`. The default value is 0.

`int not null database_provider_id` [Column]
References `Database_Providers(database_provider_id)`. The default value is 0.

31.20 Languages

`int identity not null language_id` [Column]
`primary key PK_Languages` [Constraint]

`varchar(64) not null language_name_english` [Columns]
`varchar(64) not null language_name_german`
The default value is `'N/A'`.

`char(3) not null language_abbrev` [Column]
The default value is `'N/A'`.

31.21 Records_Languages

Association table.

`int not null record_id` [Column]
References `Records(record_id)`.

`int not null language_id` [Column]
References `Languages(language_id)`.

`char(1) null association_type` [Column]

`varchar(32) not null association_type_name` [Column]
The default value is `'N/A'`.

31.22 Physical_Descriptions

The following PICA categories used all used for the physical descriptions of resources. Unfortunately, some are used for more than one physical feature, so they are ambiguous. For example, the English description of the PICA3 4060 category is “Size or Range, Specification of Material, Technical System”. In addition, these categories all have only a single field, namely “Text”, so that unlike other categories, the ambiguity cannot be resolved by the use of fields.

PICA3	PICA+
4060	034D
4061	034M
4062	034I
4063	034K

!! TODO: Add cross-references and index entries!

The number of different features that can be described using these classifications is large enough to make it impracticable to have a column for each one in the Physical_Descriptions table. Due to the above-mentioned ambiguity, it would also be impossible for a program to determine which column would be appropriate for a given entry. The Physical_Descriptions table therefore includes a pica_category_id column, so that users may at least see how the data was categorized in the PICA source.

With respect to these categories, the pica_field_id in the Physical_Descriptions table is redundant. I have added it in case other categories are used for physical descriptions.

`int identity not null physical_description_id` [Column]
`primary key PK_Physical_Descriptions` [Constraint]

`varchar(256) not null text` [Column]
  The default value is 'N/A'.

`int not null pica_category_id` [Column]
  References PICA_Categories(pica_category_id).
  See Section 31.2 [PICA_Categories], page 110.

`int not null pica_field_id` [Column]
  References PICA_Fields(pica_field_id).
  See Section 31.3 [PICA_Fields], page 110.

31.23 Records_Physical_Descriptions

Association table.

`int not null record_id` [Column]
  References Records(record_id).
  See Section 31.5 [Records], page 111.

`int not null physical_description_id` [Column]
  References Physical_Descriptions(physical_description_id).
  See Section 31.22 [Physical_Descriptions], page 116.

31.24 Remote_Access

The Remote_Access table stores information from entries of the category Pica+ 209R/Pica3 7133. See Section 23.11 [209R Local information regarding remote access to electronic resources], page 59, and Section 24.9 [7133 Local information regarding remote access to electronic resources], page 66.

The function Category_Container::sub_F_209R, which is called by Category_Container::F_209R, writes to this table. See ⟨undefined⟩ [Functions for Specific Categories], page ⟨undefined⟩.

`int identity(0, 1) not null remote_access_id` [Column]
`primary key PK_Remote_Access` [Constraint]

`bit not null license_indicator` [Column]

`varchar(64) not null format_type` [Column]
  The default value is 'N/A'.

`varchar(512) not null web_display_text` [Column]
The default value is `'N/A'`.

`varchar(512) not null URL` [Column]
The default value is `'N/A'`.

`varchar(512) not null URN` [Column]
The default value is `'N/A'`.

`varchar(512) not null internal_remarks` [Column]
The default value is `'N/A'`.

31.25 Records_Remote_Access

Association table. See Section 31.5 [Records], page 111, and Section 31.24 [Remote_Access], page 117.

The function `Category_Container::sub_F_209R`, which is called by `Category_Container::F_209R`, writes to this table. See ⟨undefined⟩ [Functions for Specific Categories], page ⟨undefined⟩.

`int not null record_id` [Column]
References `Records(record_id)`.
See Section 31.5 [Records], page 111.

`int not null remote_access_id` [Column]
References `Remote_Access(remote_access_id)`.
See Section 31.24 [Remote_Access], page 117.

31.26 Exemplar_Production_Numbers

The `Exemplar_Production_Numbers` table stores information from entries of the category Pica+ 203@/Pica3 7800. See Section 23.8 [203@ Exemplar Production Number], page 57, and Section 24.10 [7800 Exemplar Production Number], page 67.

Since so little information is stored in this table, it includes a column `record_id` which references `Records::record_id`. This makes it possible to do without an association table.

The function `Subcategory_Container::f_203_AT_0` writes to this table. See Section 27.3.4.13 [Exemplar Production Number], page 87.

`int identity(0, 1) not null exemplar_production_number_id` [Column]
`primary key PK_Exemplar_Production_Number` [Constraint]

`bigint null exemplar_production_number_numeric` [Column]

`varchar(64) exemplar_production_number_text` [Column]
The default value is 'N/A'.

`int not null record_id` [Column]
References `Records::record_id`. See Section 31.5 [Records], page 111.

31.27 Call_Numbers

The `Call_Numbers` table stores information from entries of the categories Pica+ 209A/Pica3 7100–7109. See Section 23.9 [209A Call Number], page 57, and Section 24.8 [7100–7109 Call Number], page 65.

The functions `Category_Container::F_209A`, `Subcategory_Container::f_209A_a`, `Subcategory_Container::f_209A_b`, `Subcategory_Container::f_209A_f`, and `Subcategory_Container::f_209A_j` are used to write to this table. See Section 26.3.3.6 [Call Number], page 73, and Section 27.3.4.14 [Call Number], page 88.

`int identity(0, 1) not null call_number_id` [Column]
`primary key PK_Call_Numbers` [Constraint]

`varchar(128) call_number` [Column]
The default value is 'N/A'.

`smallint not null library_number` [Column]
The default value is 0.

`varchar(64) library_department` [Column]
The default value is 'N/A'.

`varchar(128) special_location` [Column]
The default value is 'N/A'.

31.28 Records_Call_Numbers

Association table. See Section 31.5 [Records], page 111, and Section 31.27 [Call_Numbers], page 119.

The function `Category_Container::F_209A` writes to this table. See ⟨undefined⟩ [Functions for Specific Categories], page ⟨undefined⟩.

`int not null record_id` [Column]
References `Records(record_id)`.
See Section 31.5 [Records], page 111.

`int not null call_number_id` [Column]
References `Call_Numbers(call_number_id)`.
See Section 31.27 [Call_Numbers], page 119.

31.29 Access_Numbers

The `Access_Numbers` table stores information from entries of the category Pica+ 209C/Pica3 8100. See Section 23.10 [209C Access Number], page 58, and Section 24.11 [8100 Access Number], page 68.

Since so little information is stored in this table, it includes a column `record_id` which references `Records::record_id`. This makes it possible to do without an association table.

The function `Subcategory_Container::f_209C_a` writes to this table. See Section 27.3.4.15 [Access Number], page 90.

`int identity(0, 1) not null access_number_id` [Column]
`primary key PK_Access_Numbers` [Constraint]

`varchar(128) not null access_number` [Column]
The default value is 'N/A'.

`int not null record_id` [Column]
References `Records::record_id`.

31.30 Subject_Types

This table stores information about the classes of subjects specified in the "Indikator/Indicator" field of Pica+ 041A/Pica3 800; 5100–5199. This is the 'S' field in Pica+ 041A. See Section 23.7 [041A Subject], page 55, Section 23.7.1 [041A Fields], page 55, and Section 24.1 [800; 5100–5199 Subject], page 60.

The `Subject_Types` table is referenced by the ODBC class `Subject_Types`. See Section 30.24 [Subject_Types (ODBC Class)], page 105.

The `indicator` column (see below) must contain one of the characters used as values in this field.

The following table shows these values. Some are only used with Pica3 5100–5199.

Value	German	English	Pica3 800
c	Körperschaftsschlagwort (Ort)	Entity Subject (Location)	Yes
f	Formschlagwort	Form Subject	No
g	Geografisches/ethnografisches Schlagwort	Geographical/Ethnographical Subject	Yes
p	Personenschlagwort	Personal Subject	Yes
s	Sachschlagwort	Material Subject	Yes
t	Titelschlagwort (800)/Werktitel (51xx)	Title Subject (800)/Work Title (51xx)	Yes
k	Körperschaftsschlagwort (Name)	Entity Subject (Name)	Yes
z	Zeitschlagwort	Temporal Subject	No

The `Subject_Types` table is completely filled by the stored procedure `regenerate_tables`. No entries are added when ZTest is run. Therefore, no default values are needed. See Chapter 32 [Database Stored Procedures], page 123.

`int identity(0, 1) not null subject_type_id` [Column] `primary key PK_Subject_Types` [Constraint]

`char(1) null indicator` [Column]

`varchar(64) not null description` [Column]

`bit not null pica3_800` [Column]

`bit not null pica3_51xx` [Column]

31.31 Subjects

This table stores information from entries of the categories Pica+ 041A/Pica3 800; 5100–5199. See Section 23.7 [041A Subject], page 55, and Section 24.1 [800; 5100–5199 Subject], page 60.

`int identity(0, 1) not null subject_type_id` [Column]
`primary key PK_Subjects` [Constraint]

`int not null subject_type_id` [Column]
: References `Subject_Types::subject_type_id`. See Section 31.30 [Subject_Types], page 120.

`varchar(128) not null default 'N/A' subject` [Column]

`bigint not null id_number_ppn` [Column]
: The default value is 0.

`smallint null chain_number` [Column]

`smallint null chain_link_number` [Column]

`varchar(256) not null default 'N/A' chain_info` [Column]

31.32 Records_Subjects

Association table.

`int not null record_id` [Column]
: References `Records::record_id`. The default is 0.

`int not null subject_id` [Column]
: References `Subjects::subject_id`. The default is 0.

31.33 Permutation_Patterns

This table stores information from entries of the categories Pica+ 041A/Pica3 5100–5199. Permutation patterns (German *Permutationsmuster*) aren't used in Pica3 800. See Section 23.7 [041A Subject], page 55, and Section 24.1 [800; 5100–5199 Subject], page 60.

This table has a column `record_id`, which references the `record_id` column in the `Records` table, and columns `subject_id_start` and `subject_id_end`, which reference the `subject_id` column in the `Subjects` table (see below). It is therefore an association table. See Section 31.5 [Records], page 111, and Section 31.31 [Subjects], page 121.

`int identity(0, 1) not null permutation_pattern_id` [Column]
`primary key PK_Permutation_Patterns` [Constraint]

`int not null record_id` [Column]

References `Records::record_id`. See Section 31.5 [Records], page 111.

`int not null subject_id_start` [Column]

References `Subjects::subject_id`. See Section 31.31 [Subjects], page 121.

`int not null subject_id_end` [Column]

References `Subjects::subject_id`. See Section 31.31 [Subjects], page 121.

`int not null chain_number` [Column]

`varchar(64) not null permutation_pattern` [Column]

31.34 Temp_IDs

`int temp_id` [Column]

32 Database Stored Procedures

`create_tables` [Stored Procedure]
    Creates the database tables.

`regenerate_tables` [Stored Procedure]
    Initializes the database tables. Some of them contain entries that must be present before data is read from the input file.

`delete_tables` [Stored Procedure]
    Deletes the entries in the database tables, without dropping the tables.

`drop_tables` [Stored Procedure]
    Drops the database tables.

`create_catalogs` [Stored Procedure]
    Creates the fulltext catalogs.

`fill_catalogs` [Stored Procedure]
    Fills the fulltext catalogs.

Glossary

B

Bib-1. “The Bib-1 attribute set is part of the Z39.50 client server protocol.” (http://en.wikipedia.org/wiki/Bib-1).

E

ELN: See “External Library Number”.

External Library Number: Abbreviation: ELN. See [Abbreviations E], page 126.

M

MFC: See “Microsoft Foundation Classes”.

Microsoft Foundation Classes: Abbreviation: MFC. See [Abbreviations M], page 126.

O

ODBC: See “Open Database Connectivity”.

Open Database Connectivity.

P

PPN: See “Pica Production Number”.

PQF: See “Prefix Query Format”.

Pica Production Number: Abbreviation: PPN. See [Abbreviations P], page 126.
Prefix Query Format.
Pica: See “Project for Integrated Catalogue Automation”.
Project for Integrated Catalogue Automation: Abbreviation: Pica.

R

RPN: See “Reverse Polish Notation”.

RSWK: See “Rules for the Subject Catalogue” (“Regeln fuer den Schlagwortkatalog”).

Regeln fuer den Schlagwortkatalog (RSWK): Rules for the Subject Catalogue. See Section 23.7 [041A Subject], page 55, and Section 24.1 [800; 5100–5199 Subject], page 60.

Reverse Polish Notation (RPN).

Rules for the Subject Catalogue: Regeln fuer den Schlagwortkatalog (RSWK) See Section 23.7 [041A Subject], page 55, and Section 24.1 [800; 5100–5199 Subject], page 60.

Y

YAZ.

Z

Z39.50.

Abbreviations

E

ELN: External Library Number
See [Glossary E], page 124.

EOF: end-of-file character

engl: English

G

germ: German

M

MFC: Microsoft Foundation Classes
See [Glossary M], page 124.

O

ODBC: Open Database Connectivity
See [Glossary O], page 124.

P

Pica: Project for Integrated Catalogue Automation See [Glossary P], page 124.

PPN: Pica Production Number
See [Glossary P], page 124.

PQF: Prefix Query Format
See [Glossary P], page 124.

R

RPN: Reverse Polish Notation
See [Glossary R], page 124.

RSWK: Regeln fuer den Schlagwortkatalog (Rules for the Subject Catalogue)
See Section 23.7 [041A Subject], page 55, Section 24.1 [800; 5100–5199 Subject], page 60, and [Glossary R], page 124.

Bibliography

Michael, James J. and Mark Hinnebusch. *From A to Z39.50.* A Networking Primer. Mecklermedia Corporation. Westport, London 1995. ISBN 0-88736-766-6.

Concept Index

Variable Index

A

B

C

D

E

Q

R

S

T

Data Type Index

Function Index

~

C

D

O

P

R

S

T

U

W

Y

Z

Database Table Index

A

B

C

D

E

F

G

H

I

L

M

P

R

S

T

U

W

Y

Database Stored Procedure Index

Filename Index

Pica+ Index

0

2

A

C

E

F

H

I

K

L

M

O

P

R

S

U

Y

Z

Pica3 Index

0

1

4

5

7

8

A

C

E

F

H

I

K

L

M

O

P

R

S

U

Y

Z

Appendix A GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

   The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

   Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.0.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix B GNU General Public License

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software

which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy  name of author

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.